Securing Medical Devices and Protecting Patient Privacy in the Technological Age of Healthcare

by

Paul D. Martin

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

May, 2016

Abstract

The healthcare industry has been adopting technology at an astonishing rate. This technology has served to increase the efficiency and decrease the cost of healthcare around the country. While technological adoption has undoubtedly improved the quality of healthcare, it also has brought new security and privacy challenges to the industry that healthcare IT manufacturers are not necessarily fully prepared to address.

This dissertation explores some of these challenges in detail and proposes solutions that will make medical devices more secure and medical data more private. Compared to other industries the medical space has some unique challenges that add significant constraints on possible solutions to problems. For example, medical devices must operate reliably even in the face of attack. Similarly, due to the need to access patient records in an emergency, strict enforcement of access controls cannot be used to prevent unauthorized access to patient data. Throughout this work we will explore particular problems in depth and introduce novel technologies to address them.

Each chapter in this dissertation explores some aspect of security or privacy in the

ABSTRACT

medical space. We present tools to automatically audit accesses in electronic medical record systems in order to proactively detect privacy violations; to automatically fingerprint network-facing protocols in order to non-invasively determine if particular devices are vulnerable to known attacks; and to authenticate healthcare providers to medical devices without a need for a password in a way that protects against all known attacks present in radio-based authentication technologies. We also present an extension to the widely-used beacon protocol in order to add security in the face of active attackers; and we demonstrate an overhead-free solution to protect embedded medical devices against previously unpreventable attacks that evade existing control-flow integrity enforcement techniques by leveraging insecure built-in features in order to maliciously exploit configuration vulnerabilities in devices.

Primary Reader: Dr. Aviel D. Rubin

Secondary Reader: Dr. Anton Dahbura and Dr. Malek Ben Salem

Acknowledgments

I would like to thank many people for their help, support, collaboration and guidance along this journey. First, I would like to thank my advisor, Avi Rubin, for mentoring me since I started Johns Hopkins as an undergraduate in 2007. Avi is responsible for much of my personal and professional development and without him I likely never would have been introduced to this field at all.

My thesis committee, Avi Rubin, Malek Ben Salem and Tony Dahbura, have worked hard to read and evaluate this thesis and I am grateful to them for their time, effort and advice.

Several other people have also provided me mentorship and research support over the years that I am immensely thankful for. They include Stephen Checkoway, Sayeed Chowdhury, Matthew Green, Michael Hylkema, Seth Nielson and Stanley Pietrowicz. I would additionally like to extend my thanks to the computer science faculty of Johns Hopkins University for making my experience over the past eight years fulfilling and for their countless hours of dedicated work with me as a student. In addition to the computer science faculty, I would like to thank the administrators in the computer

ACKNOWLEDGMENTS

science office

Versions of the chapters in this dissertation have appeared in various publications or have been submitted to various venues. I would like to take this opportunity to thank my co-authors on each publication. They include Malek Ben Salem,[1] Rafae Bhatti,[2] Joseph Carrigan,[3] Stephen Checkoway,[1,4] Matthew D. Green,[4] Avi Rubin,[1,2,4,5] Michael A. Rushanan,[3–5] David Russel,[1] Tom Tantillo[5] and Yifan Yu.[1] These publications and submissions may have been reproduced in whole or in part in this dissertation. Versions of these publications may also appear in the dissertations of any of my coauthors.

My work on this thesis was supported under the Health and Human Services-funded Strategic Healthcare IT Advanced Research Projects on Security (SHARPS) grant, the NSF-funded Trustworthy Health and Wellness (THaW) grant, and by Accenture Technology labs. I am thankful to these organizations for their financial support of my work.

During my time in the Ph.D. program I had many wonderful lab mates, past and present. I would like to thank Ayo Akinyele, Christina Garman, Gabe Kaptchuk, Ian Miers, Matt Pagano, Mike Rushanan and Sam Small for making this experience fun.

I would like to express my gratitude to Christine Nichols for her constant encouragement while writing this thesis and for spending hours proofreading this very long and technical document.

Finally, I would like to give special thanks to my parents, Marsha and James, my

sister, Krysta, my grandmother Maria, my uncle David and my aunt Linda for their lifetime of support, guidance and inspiration.

Dedication

This thesis is dedicated to the memory of my grandfather, Anthony Sarro.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The healthcare industry has been adopting technology at an astonishing rate. This technology has served to increase the efficiency and decrease the cost of healthcare around the country. However, while technological adoption has undoubtedly improved the quality of healthcare, it also has created new security and privacy challenges that healthcare IT manufacturers are not necessarily fully prepared to address in the industry.

This dissertation explores some of these challenges in detail, and proposes solutions that will make medical devices more secure and medical data more private. Unlike many other industries, the medical space must address unique challenges and requirements that place significant constraints on possible solutions to problems. For example, it is imperative that medical devices continue to operate reliably even in the face of attack. Similarly, due to the need to access patient records in an emergency,

strict enforcement of access controls cannot be used to prevent unauthorized access to patient data.

Throughout this work, we will explore particular problems in depth and introduce novel technologies that provide solutions. Our work culminates in a vision for a more secure and private technologically-enabled healthcare environment centered around the integration of the technologies proposed in this work.

1.1 Security and Privacy Issues in Modern Healthcare

In order to focus our research, we identified the most pertinent security and privacy issues in modern healthcare today. As technology plays an increasing role in healthcare, numerous security and privacy problems are beginning to reveal themselves. In many cases we need new technologies to address the issues.

One particularly prevalent trend in healthcare IT today is the move from paper charts to electronic medical records (EMRs), as mandated by the Health Insurance Portability and Accountability Act (HIPAA).[6] In addition to mandating a transition to EMRs, HIPAA contains numerous privacy provisions that healthcare providers must follow in order to fully comply with the law. These provisions have created a booming industry around technologies that help healthcare providers protect patient privacy in a HIPAA-compliant environment.

However, properly configuring and deploying these necessary types of privacy-protecting technologies requires extensive expertise. To make matters worse, unlike in other fields, privacy protections in healthcare cannot be enforced in a rigid fashion: a healthcare worker may need emergency access to patient information, even when he or she has not obtained prior authorization.

Currently, Healthcare providers must often rely on audits to determine whether privacy violations have occurred, but such audits are complex and difficult to carry out. EMR log audits require auditors to search through massive amounts of data in order to identify a violation that may or may not have occurred. Administrators with which we spoke told us that they don't even try to perform an audit unless they have a pre-existing indication of what they should examine. As a result, privacy violations are often only detected in audit logs after they have already been reported to a healthcare provider. Thus, this highly reactive approach to audits is only useful for evidence gathering after a report has been made. Audits have only a very limited range of utility for immediate privacy protection when emergency situations intersect with patient privacy concerns.

Unlike corporate or home networks where physical access to the network is restricted, healthcare practices need to treat a wide variety of patients, and thus their networks are more vulnerable to physical attacks by malicious parties. In fact, in most healthcare environments, potential attackers are left with completely unsupervised access to computer on wheel (COW) carts containing both workstations which

store private patient data, and safety-critical therapy devices. As a result of this pervasive and constant threat, authorization in hospitals is critical in order to prevent malicious parties from improperly accessing patient resources or tampering with medical equipment.

Furthermore, as healthcare practices become increasingly wired and complex, many more medical devices are becoming network-connected; some devices receive firmware updates over the Internet, and some even send data to remote servers or cloud storage so that doctors can retrieve test results from a mobile device when they are away from the office. Since many medical devices implement widely-deployed technologies, even if an attacker does not have direct physical access to a medical device, the attacker can often still find remotely exploitable vulnerabilities that would allow him or her to interfere with the operation of the device.

Although health networks today contain many embedded medical devices responsible for directly administering therapy, device manufacturers often focus on reliability and usability rather than on security. In addition, even if some devices possess the underlying options to be configured securely, many of these devices are not actually configured securely by manufacturer default. Instead, healthcare IT staff are expected to properly secure the device before it is put into use at their specific location.

Unfortunately, many healthcare practices do not possess the skilled IT workers who are able to determine how to properly configure and secure all of the different medical devices, both to protect privacy and to prevent attacks. This secure config-

uration problem is compounded by the fact that medical device manufacturers often charge for device updates, which include security fixes. These updates are often expensive, so healthcare providers are hesitant to purchase them unless they add useful new features to the devices.

There are many negative consequences of an attacker compromising an embedded medical device connected to a healthcare network. The attacker could degrade the performance of the device, tamper with measurements that the device may take, disable the device, or in some cases cause the devices to directly harm patients. (One doctor we spoke with mentioned that in the case of radiation therapy machines an attacker could disable safety controls and potentially utilize the device to kill a patient.)

1.2 Our Vision

We envision the solutions proposed in this thesis being combined into a cohesive system that modernizes security and privacy in the healthcare environment of the future. In such an environment, our proposed technologies will dramatically decrease the cost and increase the effectiveness of securing medical devices and protecting confidential patient information. We believe that the technologies proposed in this work naturally support one another and that the whole is greater than the sum of the parts.

Let us envision the workflow in our imagined future medical environment. Dr. Smith is a doctor in cardiology. He begins his shift by putting on his KBID bracelet. Dr. Smith signs into a terminal to activate the bracelet. He then begins making his rounds and visits his first patient. Using a location-based tracking system implemented on the Beacon+ platform, Dr. Smith is able to pull up the patient's chart directly on his smartphone, without having to enter any information into the phone at all. Dr. Smith sees that the patient needs an EKG. He moves to the EKG machine in the corner of the room and taps his KBID bracelet to the contact pad in order to authenticate himself to the computer terminal that controls the procedure.

Today Dr. Smith is on double-duty; his friend in geriatrics has taken a personal day and Dr. Smith has agreed to handle any emergencies in geriatrics while he is simultaneously on-call in cardiology. Suddenly, Dr. Smith receives a phone call. One of the patients in geriatrics has suffered a seizure. Dr. Smith pulls out his smartphone and enters the patient's name into the EMR system's mobile application in order to pull up the patient's medical record. The EMR software prompts the doctor that since this patient is in a geriatrics and Dr. Smith is registered in cardiology, the access will be allowed but it will be logged in the audit system. Dr. Smith proceeds and discovers that the geriatrics patient's infusion pump is administering too high a dose.

Although these infusion pumps used by the hospital are ten years old and have previously been subjected to malicious re-configuration attacks, thanks to the retrofitted

Sentinels present on each device, the insecure web configuration and telnet modes of the pumps are disabled when the pumps are deployed in the hospital. The healthcare IT staff was able to quickly determine which pumps to retrofit by running a machine learning-based fingerprinting tool to identify which pumps were running vulnerable versions of non-upgradable ROM-based firmware.

Dr. Smith enters an immediate-infusion drug order directly from his phone in order to correct the patient's dosage, and pages a nurse in the department to check on the patient. Later, the doctor's EMR access is detected in an automated audit. The hospital auditor investigates the access and sees that Dr. Smith has accessed the data of a patient in geriatrics. The auditor checks the shift records and discovers that Dr. Smith had a temporary one-day assignment in geriatrics during the period the access had occurred. The auditor moves on to the next anomalous access.

All of the technologies just described are fully detailed in this work. They have served to improve the quality of care that Dr. Smith is able to provide while making the technology easy to use. The technology has further served to help ensure that patient privacy is being upheld in a way that is transparent to Dr. Smith. Our future healthcare environment leverages technology to protect patient privacy, increase the speed of care, and protect devices against previously unknown attacks, all while enhancing the workflow of healthcare providers responsible for caring for others.

1.3 Our Approach

Clearly many of the issues described above are complex and multifaceted. This dissertation looks at each issue and introduces techniques that can be deployed in real-world healthcare settings in order to work towards addressing each of these unique challenges in the healthcare IT landscape.

1.3.1 Enforcing Minimum Necessary Access in Healthcare Through Integrated Audit and Access Control

One of the most important requirements of HIPAA is the "minimum-necessary" access requirement, which states that healthcare personnel must be granted no more access to electronic healthcare data than is necessary in order to work effectively. Due to the complexity of constructing such a policy, many hospitals do not comply with the regulation and instead manually audit the logs when they suspect that abuse has occurred. This audit-only approach is error-prone, reactive, and difficult to carry out due to the volume of data contained in the logs.

To address this problem, we have built a *policy engine* capable of automatically auditing logs and separating normal accesses from abnormal accesses. Our policy engine implicitly constructs role-based policies from the audit data in order to produce

a workable policy that can be used to enforce minimum-necessary access. The policy engine can also audit an existing role-based access policy by comparing it to observed accesses in order to determine whether the existing policy is over-permissive compared to actual usage patterns.

1.3.2 Classifying Network Protocol Implementation Versions: An OpenSSL Case Study

We present a new technique for identifying the implementation version number of software that is used for Internet communications. While many programs may exchange version numbers, oftentimes only a small subset of them send any information at all. Furthermore, they usually do not provide accurate details about which implementation is used. Using machine learning techniques to build a feature database, we then apply this database to network traffic to identify specific implementations on servers. We apply our technique to OpenSSL and report our results. This work is particularly useful for extracting information about protocol implementations on Internet-facing embedded devices and may actually be the only non-invasive way to determine whether such devices are likely to be vulnerable to known attacks. An administrator of a hospital network could run this tool against the entire network in order to determine whether deployed devices are likely to be attacked by malware.

1.3.3 Sentinel: Secure Mode Profiling and Enforcement for Embedded Systems

Embedded devices are employed in a variety of mission-critical environments. Although many of these devices contain a wide variety of modes and features to support all possible use cases, in practice only a small subset of these modes may be used in a given deployment. In many embedded devices, some of these unused modes represent a security risk.

We address this problem by creating Sentinel, a secure mode profiler for embedded systems. Sentinel uses a bus tapping interface to derive a partial control flow graph during device execution. This graph represents the subset of device modes actually observed during use. The control flow graph is generated without any prior knowledge of the device or its software and constitutes a security profile which can be used to audit device execution in order to detect attacks. The profile can be easily enforced by existing bus monitors with minor modifications.

1.3.4 Beacon+: Applications of Short-Range Authenticated Unidirectional Advertisements

We present Beacon+, a Bluetooth Low Energy (BLE) device that extends the design of the popular Beacon specification with unspoofable, temporal and authen-

ticated advertisements. We use Beacon+ to implement a secure real-time tracking system that also serves as a foundation for other novel location-based applications. We describe two such applications, namely the location-based restrictions on access control we implement, and real-time indoor navigation.

1.3.5 KBID: Kerberos Bracelet Identification

The most common method for a user to gain access to a system, service, or resource is to provide a secret, often a password, that verifies his identity and thus authenticates him. Password-based authentication is considered strong only when the password meets certain length and complexity requirements, or when it is combined with other methods in multi-factor authentication. Unfortunately, many authentication systems do not enforce strong passwords due to a number of limitations; for example, the greater amount of time taken to enter complex passwords. We present an authentication system that addresses these limitations by prompting a user for credentials once and then storing an authentication ticket in a wearable device that we call *Kerberos Bracelet Identification* (KBID).

1.4 Outline of This Work

Each chapter in this dissertation explores one unsolved problem related to security or privacy in the health medical space.

Chapter 2 examines current security and privacy research trends in the health and medical space.

Chapter 3 looks at automatically auditing accesses in electronic medical record systems in order to proactively detect privacy violations.

Chapter 4 introduces a tool for fingerprinting network-facing protocols in order to non-invasively determine they prevalence of protocol implementation deployments that are vulnerable to known attacks. We envision that such a tool would be useful to healthcare IT staff in determining whether or not devices on their network support protocol or library versions that are vulnerable to known attacks.

Chapter 5 demonstrates a zero-overhead solution to protect embedded medical devices against previously unpreventable attacks that leverage built-in device features in order to allow an attacker to compromise the security or privacy of an embedded device.

Chapter 6 presents an extension to the widely-used beacon protocol in order to add security in the face of active attackers. We use these extensions to add a second factor of authentication to electronic medical record lookups on mobile devices.

Chapter 7 describes an authentication bracelet that allows for fast authentication in healthcare settings while protecting the wearer against known attacks

present in radio-based authentication technologies without compromising speed of authentication.

f

Although security and privacy in the health and medical space (henceforth health and medical security) is an extraordinarily broad topic, there are currently several distinct subtopics in this field that allow us to divide the research space up into a finite number of clusters. These clusters represent the four loci of the vast majority of academic research in this space today.

Health and medical security has become a critical topic, as electronic medical records have become more widely distributed and medical devices are increasingly connected to the "Internet of Things." Healthcare providers have moved from storing medical records in vast paper systems to storing medical records in local computer systems, to storing medical records in cloud-based health information exchanges (HIE). Naturally, as the storage of sensitive patient information transitions to cloud-based systems, healthcare providers must emphasize patient privacy protection. In order to support these efforts, the Office of the National Coordinator and the National Science Foundation have both awarded large research grants to consortiums of universities to study both the security and privacy aspects of modern healthcare with the goal of producing the technological capability necessary to adequately secure these technologies.[7,8] Much of the privacy research in these spaces has come out of these inter-university collaborative efforts.

In addition to protecting patient privacy, healthcare providers must also protect medical devices from malicious exploitation. Today, medical devices from insulin

pumps to radiology machines all have network-connected components. If an attacker successfully compromises such a device, he or she may be able to affect device performance, and potentially be able to kill the patient attached. Such an attack may even be undetectable. Thus medical device security is of paramount importance. Unfortunately, this issue, particularly in the non-implantable medical device space, has not been as widely studied in the academic space as the privacy issues described above. Although general embedded system security has been extensively studied, as described in Section 1 healthcare IT environments have dramatically different constraints than other types of industrial environments, so the problem deserves specialized attention. Luckily, device manufacturers are increasingly aware of these concerns and are working to resolve these security issues before exploitation of medical devices becomes a pervasive threat.

Most existing research in medical data fits into one of four clusters. These clusters are briefly defined as follows:

Controlling and Auditing Access to Medical Data Access control, especially with respect to electronic medical record data, is a particularly fertile area of research in health and medical security. Projects have looked at various aspects of this problem, including structuring appropriate policies and enforcing access control constraints without interfering with patient care.

Healthcare providers leave an audit trail when they access patient data. This audit trail is often composed of thousands or millions of individual accesses; as

a result, the shear volume of this data makes auditing the accesses difficult. A large number of recent research in health and medical security attempts to solve this problem through the use of various automated techniques.

Protecting Patient Privacy Numerous research projects have looked at securing private patient data through various means such as self-encrypting cloud-stored medical records and outsourcing computation on encrypted data. Such research has focused on how to improve the usefulness of medical data by allowing it to be shared with authorized third parties, while simultaneously preventing it from being accessed by unauthorized parties.

Authenticating Users in Medical Settings Authentication can be a time consuming process, particularly if complex policies are used. Given the time-critical nature of resource access in emergency situations, numerous researchers have looked at the problem of authentication in healthcare and have proposed various solutions that are designed to make authentication more secure and seamless.

Securing Medical Devices The security of medical devices against remote adversaries is of critical importance for patient safety. Medical device security is an emerging research area that extends traditional embedded systems security research by considering it within the context of the additional challenges of healthcare environments.

In this chapter we describe some of the current research in each of these areas of

health and medical security.

1.5 Controlling and Auditing Access to Medical Data

In healthcare settings, audit and access control go hand-in-hand. Access control in the medical field is complicated by the need for healthcare providers to access information during emergencies, regardless of whether they have been granted access to the information prior to the time of the emergency occurrence. This scenario is called "breaking the glass" and it allows a previously unauthorized user access to information. This problem has been studied in some existing literature.[9] As a result of these requirements, oftentimes an audit log is necessary to help an administrator understand whether or not the set of access control policies being enforced are the same as the set of access control policies being followed. The difference between these two sets has been referred to in some work as an "empathy gap" between the two models.[10]

1.5.1 Building Better Policies

There has been much research looking at building better policies in healthcare environments.[11–13,13–16] One common approach to the problem is to build better

models.[11,13,17–19] Work on Experience-based Access Management[11] sought to improve access control models in healthcare by viewing them as dynamic and ethereal as opposed to static and fixed. EBAM, as a concept, proposes a flexible approach to healthcare access control, one that is constantly being re-evaluated. In section 2 we build on this concept by using actual audit logs combined with data mining techniques to actually facilitate this re-evaluation. Related work[17–19] has also looked at using audit logs in order to better understand how to model roles in the policy.

1.5.2 Formal Analysis

Other work has sought to formalize the language[20,21] used to describe medical record accesses in order to allow policy rules to be analyzed using formal logic.[22] This is particularly useful within the context of trying to accurately model policy interactions across healthcare environments[15] rather than just within them.

1.5.3 Improved and Automated Audit

Yet another aspect of this work addressing issues in current medical security technology looks at making better use of audit logs. In general this is accomplished in one of two ways: by making it easier for administrators to manually audit logs[23,24] or by building machine-learning based systems to perform automated log audits.[25–27] Most research has focused on automated log analysis, and therefore, has proposed

numerous techniques for performing automated log audits. One technique enforces formalized dataflow models on the audit logs in order to determine where the accesses branch from the predetermined dataflow pathways.[25,28–30]

Several other techniques use statistical analysis and machine learning to detect potentially unauthorized accesses.[?,26,31] Other research has used such techniques in order to determine how closely an audit log fits a predefined access control model in realtime[32] or to audit logs that are missing data points.[33] Additionally work has focused on purpose instead of access. This work imposes restrictions on how accessed data may be used. It builds a model using audit logs to determine when the purpose policy may have been violated.[34,35]

1.5.4 Game Theory Models for Audit

Finally, a small body of research exists that looks at audit in game theoretic terms.[36–38] These projects holistically reconcile the audit and enforcement in order to determine how to best provide incentives and deterrents[39] when enforcing privacy rules on healthcare data. This work models policy enforcement as a game and looks to refine audit algorithms to improve their efficiency.[38]

1.6 Protecting Patient Privacy

A large body of research in health and medical security focuses on protecting patient privacy. Protecting patient privacy is a broad mandate with different meanings to different people. Research has focused on policies for sharing medical records, or parts of medical records, with third parties, as well as encryption-enforced access control, rights management, and outsourced computations on encrypted medical data.

1.6.1 Third Party Access to Medical Data

One popular area that current research has explored is how patients can best manage and share sections of their medical data with third parties.[40–43] One realization of this idea is the concept of the personally controlled health record[40]–the idea that a medical record is fully owned by and travels with the patient rather than the healthcare provider. Additional work in this area looks at differentiating between *s*ensitive data in patient health records and other types of data.[43] This work looks to segment medical records in a way that allows patients to share parts of medical records with different parties under different conditions. This is similar to how a smartphone permissions system allows different apps to access different features on the device.

Much of the work in this area looks at health information exchanges (HIE) and how

different institutions in an HIE may have different policies for securing and sharing patient data. One interesting problem on which some research has focused is how to search through data stored at different healthcare sites[44] in a privacy preserving fashion. Another interesting researh focus has been on how to limit the sharing of data between different sites to only relevant information in order to approximate how data has been shared in the past.[42] For example, previously, if a patient needed to share information from a primary care physician with a specialist, the primary care physician's office would only fax over relevant information to the specialist and would omit the non-relevant parts of the patient's medical record. Some work in this area has even gone as far as to propose a cryptographic model for an entirely anonymous healthcare system.[45]

1.6.2 Cryptographic Enforcement of Sharing Policies

Additional work in this area looks at enforcing medical record sharing policies using cryptography.[46] Some of this work considers using techniques such as identity-based encryption (IBE) to enforce sharing constraints on records shared in health-information exchanges.[46] similarly, other work looks at using attribute-based encryption (ABE) to enforce these constraints.[12] In the IBE case, medical records can be encrypted in such a way that they can only be accessed by certain entities. In the

ABE case, medical records can be encrypted, in whole or in part, under boolean policies. If the accessor possesses the necessary attributes to satisfy the policies, then the accessor is able to decrypt the record. Additional work has looked at storing encrypted medical data with a third-party cloud-provider and using a client-side decryption in order to access the data.[47] Other work along these lines has examined the possibility of using DRM-technologies to enforce access control.[48,49]

1.6.3 Outsourced Computation on Encrypted Data

Lastly, an increasingly important area of research that has been gaining popularity as of late, focuses on allowing researchers to run experiments on sensitive patient data without actually accessing the data. Due to the highly sensitive nature of some data, such as sequenced genomes, researchers have begun trying to devise schemes that allow all operations to be performed on data that is still encrypted by using homomorphic encryption.[50–56] This area of research is still in its infancy, but we believe that it will exponentially increase in popularity in the coming years as large-scale homomorphic encryption schemes become more practical and we learn more about the extent of the private patient data that the human genome contains.

1.7 Authenticating Users in Medical Settings

Authentication has always been of particular concern in medical settings because it is the necessary first step in the healthcare provider's process of engaging with any subsequent technological solution such as those outlined in this work. Unfortunately, healthcare staff often see authentication mechanisms as obstacles to accessing the patient data necessary to provide life-saving care in an emergency situation. In some time-critical situations, a patient's health may depend on how quickly staff can authenticate to a computer system to obtain critical information. Similarly, healthcare staff may not always remember to log out of a terminal before they are called off to deal with another emergency situation. This leaves computer systems that contain incredibly sensitive information potentially exposed to attacks. Thus, authentication has become a somewhat popular area of study in health and medical security.

Similarly, as medical care is increasingly moving out of healthcare facilities and into the home, researchers have been investigating new techniques that would allow a user to authenticate to their personal health devices. In many cases these devices are used as part of a body area network and thus they must all be authenticated to the same user simultaneously. Therefore, the biometric research relevant to our work focuses primarily on new biometrics that are suitable for use in such health environments. We will therefore not discuss biometrics that are used for other purposes in

this section.

Authentication-related research projects can be split into two classes: biometrics and authentication methods.

1.7.1 Biometrics

Researchers are always looking for new biometrics that can be used as part of authentication protocols. Recent research has discovered some novel biometrics that differ significantly from the traditional, more widely-known biometrics such as retinal patterns and fingerprints. These new biometrics are being investigated for their applications to mobile healthcare and body-area networks.

One area of research has been on using bioimpedance, which measures the human body's natural resistance to electrical currents, as a biometric. Research has shown that each person's body responds to different frequency alternating currents in slightly different ways.[57] The uniqueness of this response can be utilized as a biometric when authenticating a user.

Other research has examined using vocal resonance as a biometric. Vocal resonance is the resonance of a voice as measured from a device placed on someone's body–essentially it is the internal resonance of their voice inside of their body. Research in this area produced a device[58] that was able to differentiate between different speakers; the device was also able to determine whether or not it was actually on the subject's body.

Both of the above mentioned biometrics could be integrated into wearable medical devices in order to authenticate a user to the device outside of a traditional healthcare setting, without requiring any action whatsoever on the user's part.

1.7.2 Authentication Methods

Recent research has proposed several novel authentication methods meant to improve the speed and efficiency of authentication in healthcare.

One emerging research focus in authentication methods uses body-coupled communication. Body-coupled communication uses skin conductivity as a data transmission medium. "Existing work has introduced a system for key exchange over a body area network.[59] By applying a very small voltage to the tissue of a dead mouse, the researchers were able to communicate at a rate of 5Hz or 5 bits per second."[3]

Additional work in authentication methods as applied to healthcare, examines what researchers call the "one body problem".[60] This research is concerned with the problem of whether multiple medical devices are resident on the same body. The solution[61] works by coordinating readings accelerometers present on multiple sensors with the readings from an accelerometer on a smartphone in order to determine whether the sensors are on the same moving body.

There has also been some research on the problem of de-authentication, a particularly pertinent problem in modern healthcare.[62] This work resulted in a working prototype of a deauthentication bracelet called ZEBRA. ZEBRA's function is to de-

tect when a user is done using the computer terminal and to then deauthenticate the user from the terminal even if the user forgets to do so. "In ZEBRA, a user wears a bracelet that encapsulates a wireless radio, accelerometer, and gyroscope; these components record and transmit wrist movements to a computer system that is currently being used. The computer system continually compares received movement measurements to input it receives from its keyboard and mouse. If these two measurements are not correlated, the current session is de-authenticated."[3]

1.8 Securing Medical Devices

Medical devices are one of the most critical types of cyber-physical systems in existence today. Although most medical devices share many commonalities with other types of industrial embedded devices, medical devices also pose unique security challenges not found in most other types of embedded systems. These challenges were described in 1. Note that the medical devices relevant to the work in this dissertation are non-implantable in nature. Thus, we will limit this discussion of medical device security to non-implantable medical devices, particularly since a recent survey of the state of implantable medical device security already exists.[63]

Despite these serious concerns about medical device safety and security, medical device-specific embedded systems security research is a relatively new field. In fact, as little as five years ago there had been almost no academic research exploring the

problem.[64] Since then, several high profile vulnerabilities were discovered in infusion devices, which significantly increased public awareness regarding the issue.

Unfortunately, there does not appear to have been significantly more work on securing non-implantable medical devices. It is likely that as vulnerabilities in these devices continue to be discovered, this situation will change in the near future.

1.8.1 Detecting Malware in Medical Devices

Malware detection on medical devices is an interesting problem because medical device software typically cannot be directly modified. Thus, any host-resident malware detection scheme is utterly infeasible in a medical setting. There has been some work focusing on using side-channels to detect malware in medical devices.[65] This work uses power as a side channel to determine if a medical device is behaving differently than its normal profile, possibly indicating that the device has been compromised.

1.8.2 Exploiting Vulnerabilities in Medical Devices

Medical device vulnerabilities are becoming an increasingly alarming topic. Recently, the discovery of remotely exploitable vulnerabilities in a popular model of infusion pump triggered an FDA warning about the security of these devices.[66] This warning significantly raised industry-awareness of the security problems resident in

embedded medical devices. There has been some previous research performed on the safety of embedded medical devices, which explored threats across insulin pump systems, a specific type of medical infusion system.[67] It explored the attack surface of these infusion pumps and detailed specific security concerns that the industry should address. These types of attacks were later implemented and discussed in a Black Hat talk in 2001.[68]

Chapter 2

Enforcing Minimum Necessary Access in Healthcare Through Integrated Audit and Access Control

2.1 Introduction

The Health Insurance Portability and Accountability Act (HIPAA),[6] enacted in 1996, sets strict privacy requirements that healthcare providers must abide by in the handling of sensitive electronic health care data. One such requirement is the "minimum necessary access" requirement,[69] which states that healthcare personnel

must be granted no more access to electronic healthcare data than the minimum necessary in order to work effectively.

Historically, Role-Based Access Controls (RBACs) with fine-grained policies have been the norm in granting and limiting access to information to groups of users with well-defined roles.[70–74] At first glance, this model may seem an appropriate fit for tightly controlling access to data in a healthcare setting. However, the RBAC model implicitly depends on the fact that access policies are well defined and understood by involved parties before accesses occur.[75] Since the purpose of an RBAC policy is to prevent unauthorized accesses to data in real-time, the current systems are inflexible.

In contrast, the emergency-based nature of today's healthcare environment has more fluid and temporal-based access control requirements. In emergency situations, employees may need access to healthcare data in ways that are unknown (and unknowable) to the policy creator in advance.[76] In these situations, denying the employee access in a rigid and inflexible fashion may actually cause serious injury or death to the patient. Healthcare providers must perform a balancing act between giving employees the minimum necessary access to healthcare data to perform their duties while simultaneously giving enough access to prevent injury to the patient in an emergency situation.

Due to the delicate nature of this balancing act, and in an attempt to err on the side of caution in time-sensitive life or death situations, many healthcare providers ignore the HIPAA mandate and fail to implement policies that grant only minimum

access.[77,78] Instead these healthcare providers rely on detailed audit logs, which contain records of every access made across the entire system, in order to detect access abuse.[79] When an abuse is suspected, IT staff must search through the logs manually in order to piece together what happened. At a large hospital, such as the one that produced the data that we worked with, there are on the order of millions of separate accesses totaling gigabytes of data per year. In the best of cases, when the IT staff has an idea of what types of abuse may have taken place, the problem is time-consuming and error prone.[80] In the worst of cases, when IT staff does not know that access abuses have occurred, the problem quickly becomes intractable and abuse may go unnoticed.[81]

In contrast to the above process, our system eliminates the need to make a trade-off between giving employees the minimum necessary access to healthcare data and allowing for unfettered access to data in emergency situations. We do so by dynamically generating flexible RBAC policies based upon typical access patterns observed from real audit logs. These policies fit the specific use case of the environment for which they are generated, which will have its own set of slightly different requirements compared to any other similar environment. While we consider our setting for this research to be the healthcare field, there is no reason that our tool cannot be used in any other field that generates detailed audit logs. In fact, there are likely many different applications where generating an RBAC policy based on real accesses is useful.

CHAPTER 2. ENFORCING MINIMUM NECESSARY ACCESS IN HEALTHCARE THROUGH INTEGRATED AUDIT AND ACCESS CONTROL

Our solution frames the policy generation problem as embarrassingly parallel[82] and is highly scalable. It can not only generate derived policies from actual audit logs but can also compare these derived policies to an existing policy in order to generate an overall assessment of how compliant the derived policy is with the stated policy. Finally, our tool also separates exception-based accesses from normal accesses in order to make manual auditing feasible and straightforward.

Our Contribution. This work contains four significant and novel contributions:

1. We have constructed a tool that can look at access logs and derive an implicit role-based access control (RBAC) policy.

2. This tool can be used in a different mode to check accesses for compliance with an intended policy even when no RBAC is actually implemented. This mode provides a method for auditing the accesses that are not explicitly defined in the stated policy.

3. This tool can compute a risk level to be used in determining compliance with the "minimum necessary access" requirement of HIPAA.

4. Using our tool we have analyzed data collected from a large hospital, have audited accesses that are not in its policy, and have generated a HIPAA minimum necessary access compliance risk level.

2.2 Role-Based Access to EMR Data

In this section we construct a theoretical framework for discussing role-based access control policies derived from electronic medical record audit logs and we define the concept of exception-based accesses.

2.2.1 Policy Definitions

Before we can compute and audit policies we must first define a theoretical framework in which to discuss them in the abstract. A policy is a set of Boolean equations defining valid accesses in the EMR system. We are concerned with four key pieces of data when formulating our policies: Department, Role, Screen and Action. Departments and roles are assigned to individual users. A department is a logical subdivision of a healthcare provider specializing in a particular area of healthcare. A role defines a user's particular function within a given department. Healthcare providers can assign employees to multiple departments and to multiple roles within each department. A screen is a logically grouped view for some subset of patient data. An action is some function computed over a screen. Actions include viewing, editing, printing and copying.

For the purposes of this discussion, we can view a department-role mapping as a single *role mapping* in our overall policy (hereinafter: role) and we can view a screen-action pair as a single *access type* (hereinafter: action) in our overall policy. A policy

is defined as a function $\mathcal{F}$: $Roles \mapsto A$: $A \subset Actions$. As previously discussed, many healthcare providers do not have a policy to enforce the minimum necessary access requirements of HIPAA. We consider this situation to be the default in the absence of any other policy definition and we define the corresponding policy to be the Cartesian product of all roles and actions: $Roles \times Actions$.

To provide a basic example, consider the following departments: $Oncology, Radiology$, roles: $Doctor, Nurse$, screens: $Prescription$ and actions: $View, Edit$. Our default policy will be:

$Oncology, Doctor \rightarrow Prescription, View$

$Oncology, Doctor \rightarrow Prescription, Edit$

$Oncology, Nurse \rightarrow Prescription, View$

$Oncology, Nurse \rightarrow Prescription, Edit$

$Radiology, Doctor \rightarrow Prescription, View$

$Radiology, Doctor \rightarrow Prescription, Edit$

$Radiology, Nurse \rightarrow Prescription, View$

$Radiology, Nurse \rightarrow Prescription, Edit$

Note that in some instances, a single user may be assigned several roles spanning multiple departments. Our system provides two methods for handling these cases.

Method 1. We consider the user's accesses within each role and department separately. For example, if a user is a doctor in both radiology and in oncology, the user's accesses would first be compared to the accesses of other users in oncology and then

be compared to the accesses of other users in radiology.

Method 2. Alternatively, our system can compare the user's accesses to the accesses of other users with the exact same group of role assignments. In this case we essentially consider multiple role assignments as a single compound role. With this method, the accesses of our doctor of both radiology and oncology would be compared to the accesses of other doctors also assigned to both radiology and oncology.

For the purposes of our analysis in this chapter, we consider each role assignment independently as in method 1.

2.2.2 Exception-Based Accesses

As a side effect of determining our minimum necessary policy we will implicitly be determining exception-based accesses as well. For our purposes, exception-based accesses are accesses to EMR fields that differ significantly from the normal access patterns of other users in the same role. Often times these accesses occur in emergency situations with no time to spare and with a patient on the verge of serious injury or death. Healthcare providers refer to the need to access medical record data in an emergency situation without regards to any existing access control systems as "breaking the glass." While breaking the glass may be one legitimate reason to step outside the confines and an access control policy, in other cases a user may access restricted information for nefarious purposes (such as to snoop on a major celebrity that is staying in the hospital). Since it is impossible to distinguish "breaking the

glass" situations from illegitimate accesses with perfect accuracy, our policy engine will notify auditors about all exception-based accesses so that the auditor can determine how to best proceed. In the next section we will explore the log analysis functions that our policy engine offers while paying special attention to which functions point out exception-based accesses, which functions point out violations of the minimum necessary policy and which functions point out both.

Many administrators may not want to ultimately remove exception-based accesses from their local policies after examining them on a case-by-case basis. We spoke with administrators who often felt that it was necessary to have a mapping in the policy in case of an emergency even if it was accessed extremely infrequently. In such situations it is especially important to note exception-based accesses because the only way an administrator will be able to detect abuse here is by performing periodic audits on the access logs. Most evidence of abuse will appear as exception-based accesses, which should therefore be audited more carefully than regular accesses.

2.3 Policy Engine

Our policy engine performs a variety of statistical tests in parallel on an intermediate data representation derived from EMR audit logs.

2.3.1 Intermediate Representation

Given that our tool must be able to work with audit logs from any number of different systems, each using different and incompatible log formats,[83,84] we construct an intermediate representation in order to convert all logs to a standard format that can be worked with by our tool. The basic format is straightforward: we simply parse out the Department, Role, Screen, Action and UserID fields (which should be supported by any tool which claims to keep detailed HIPAA-compliant audit logs) and output them in exactly that order. With this intermediate representation in place we can extend our tool to parse any log formats as necessary by writing a new mapper to parse a given log format into intermediate representation format.

2.3.2 Motivations for Parallelism

The data sets that we analyzed were small enough for our tool to process in serial rather than in parallel. However, we opted for a parallel architecture because we believe that in the near future, as healthcare providers increasingly shift to storing all of their medical records electronically, health information exchanges will store medical records for several healthcare providers in a region. This trend has already begun. For example, the Chesapeake Regional Information System for Our Patient[85] in Maryland already works as a centralized repository for medical records for many healthcare providers in the region. As health information exchanges deal with ever-

increasing numbers of medical records from many different sources, the audit logs generated by EMR systems will dramatically increase in size, eventually necessitating the use of parallelism and big-data analysis.

Our computations can be constructed in such a way that the log analysis problem becomes embarrassingly parallel.[82] Our problem requires us to analyze audit logs and compare within roles. Our universe has many roles and statistics are only ever computed within a given role–never across roles. Thus, we actually split our data into groups where all users with a given role are said to be in the same group. Once we do this, all of our computations are *within groups* rather than across the entire data set and can thus be localized.

2.3.3 Exploiting Parallelism

We are therefore using data decomposition[86] to break up and parallelize our task. We perform several different classes of computation upon each of our groups. Many of these computation classes depend on the output of previous computation classes. According to Flynn's Taxonomy[87,88] this type of problem is best suited to a multiple instruction multiple data (MIMD) architecture.[89] We use MapReduce[90–92] to fill this role. Since no data sharing occurs between individual computation classes and different computations can run in parallel, the MapReduce design methodology is well suited to our particular problem. Because MapReduce was designed for analyzing large data sets and was built to scale with the number of nodes in the cluster, our

process can be super-scaled to work on huge data sets with the use of a multi-node MapReduce cluster.

2.3.4 Log Analysis Functions

Each log analysis function is implemented as a separate set of MapReduce tasks. These tasks can be computed on the dataset in parallel, except where dependencies between the tasks exist as described in section 3.5. Note that all computations within each task are also performed in parallel on different portions of the dataset simultaneously.

2.3.4.1 Statistics

Our first MapReduce job computes simple statistics across the dataset. These statistics are used in our later analyses. The job consists of a single mapper:

$$map(rows) \rightarrow (role, access, uid) \tag{2.1}$$

and a single reducer:

$$\begin{aligned} reduce(role, access, UID) \rightarrow (role, access, avg, \\ stdDev, numAccesses, numRepresented) \end{aligned} \tag{2.2}$$

The mapper takes one or more EMR audit logs as input and parses out the role-access combination and the UID that made the access. The reducer takes the output of the mapper as input and computes the average and standard deviation. It also records how many users in the role utilized a specific $role \rightarrow access$ mapping and the total number of accesses overall. Note that this computation simply ignores users who do not access the information at all (rather than adding them to the total users count). Our other jobs will catch cases where a few users under-utilize or over-utilize their access privileges and cases where a small percentage of users are responsible for all of the accesses for a given role.

2.3.4.2 Access Counts

The counts job is also used in later analyses. It uses the same mapper as the statistics job but rather than computing statistics its reducer outputs the number of times each user utilized a given $role \rightarrow access$ mapping.

$$map(rows) \rightarrow (role, access, uid) \tag{2.3}$$

$$reduce(role, access, uid) \rightarrow (role, access, uid, count) \tag{2.4}$$

2.3.4.3 Unutilized Mappings

After we have produced our basic statistics, the first auditing function that our minimum necessary evaluation will perform is to determine unutilized mappings. These are mappings in the default policy that are never utilized by any users.

$$map(role, access) \rightarrow roles \times access \tag{2.5}$$

$$\begin{aligned} reduce(role, access) \rightarrow (role, access) : \\ (role, access) \notin (2) \end{aligned} \tag{2.6}$$

The output of this job will be all instances where the ideal policy allows for a (role, access) mapping for which there is never a corresponding access in the audit logs. In these cases the default policy is over-expressive. Such accesses are the opposite of "minimum-necessary" and are candidates for removal from the policy.

2.3.4.4 Average Below Threshold

After we have simplified our ideal policy by removing all null cases, we turn our attention to looking at underutilized mappings. To do so, we look for mappings where the average number of accesses per user is lower than a pre-established threshold. This threshold is adjustable, but we have set to two for the purposes of our analysis. Our

job will analyze the output of the statistics job and look for any instances where the average number of accesses per represented user is lower than this threshold.

$$\begin{aligned} &reduce(role, access, avg, stdDev, numAccesses, \\ &numRepresented) \rightarrow (role, access, average) : \\ &average \leq threshold \end{aligned} \tag{2.7}$$

The purpose of this job is to point out potential exception-based accesses. We believe that the accesses with low averages represent exception-based access because they show mappings that are seldom accessed on average.

Note that the threshold value for average number of accesses is user-configurable.

2.3.4.5 User Low Percentage

It may be the case that a small number of users assigned to a particular role are responsible for 100% of the utilization of a specific access credential. This might occur when a group of users has different job requirements than other users assigned to the same role (in which case the policy should have two roles) or it might occur because a small group is colluding to commit misconduct. In either case, this is exception-based access because it shows that most people in a given role do not utilize credentials that a small number of people in the same role have been observed utilizing. For our purposes we have set the threshold for reporting this activity to 5%. The "user low

percentage" job outputs a list of mappings where fewer than 5% of the users with the role are responsible for 100% of the utilizations of the given credential.

$$map(rows) \rightarrow (role, access, id) \tag{2.8}$$

$$\begin{aligned} reduce((role, access, id) + (role, access, statistic)) \rightarrow \\ uniqueIds : \frac{count(uniqueIds)}{count(numUsers)} < probability \end{aligned} \tag{2.9}$$

Note that the threshold value for the minimum percentage of users assigned to a credential that must utilize the credential is user-configurable.

2.3.4.6 Abnormal Utilization of Credentials

Abnormally frequent utilization of a credential can be an indication that a user in a particular role may actually be performing actions that are significantly different than others in that role. This could be indicative of the need for a second role to increase the granularity of the policy. It could also be indicative of a user abusing their access privileges.

Abnormally infrequent utilization of a credential, on the other hand, could be an indication that a user may be over-assigned roles or that a role definition is too broad. It may also be an indication that a user is not performing all of its job duties and

is thus not using the privileges granted to them. In either case these results warrant further investigation.

Abnormally frequent and abnormally infrequent utilization of access privileges are by definition exception-based accesses because they represent an access pattern that significantly differs from the norm.

Abnormally Frequent Utilization

$$map(rows) \rightarrow (role, access, id) \tag{2.10}$$

$$reduce(role, access, id) \rightarrow (role, access, id, count) : \\ count > avg + 3 * stdDev \tag{2.11}$$

Abnormally Infrequent Utilization

$$map(rows) \rightarrow (role, access, id) \tag{2.12}$$

$$reduce(role, access, id) \rightarrow (role, access, id, count) : \\ count < avg - 3 * stdDev \tag{2.13}$$

Note that the number of standard deviations user to establish the threshold is user-configurable.

2.3.5 Scheduling MapReduce Jobs

To fully exploit the parallel nature of this problem we have written a dependency graph for these jobs. In this way, jobs can be run in the most efficient manner possible and scheduled simultaneously with other jobs that have the same dependencies. This allows for more efficient parallelization of tasks than would be possible if all of the jobs were chained in a linear fashion. We schedule our jobs as shown in Figure 2.1

2.4 Evaluation

We obtained access to a collection of anonymized logs used by a major hospital in the Midwest[1]. This hospital uses a default permissive policy and thus any user can access any field in any medical record.

The format of these logs is shown in Table 2.1.

[1]Although the logs are anonymized, due to patient privacy laws the dataset is still considered confidential. No part of the dataset has been shared or reproduced in this chapter.

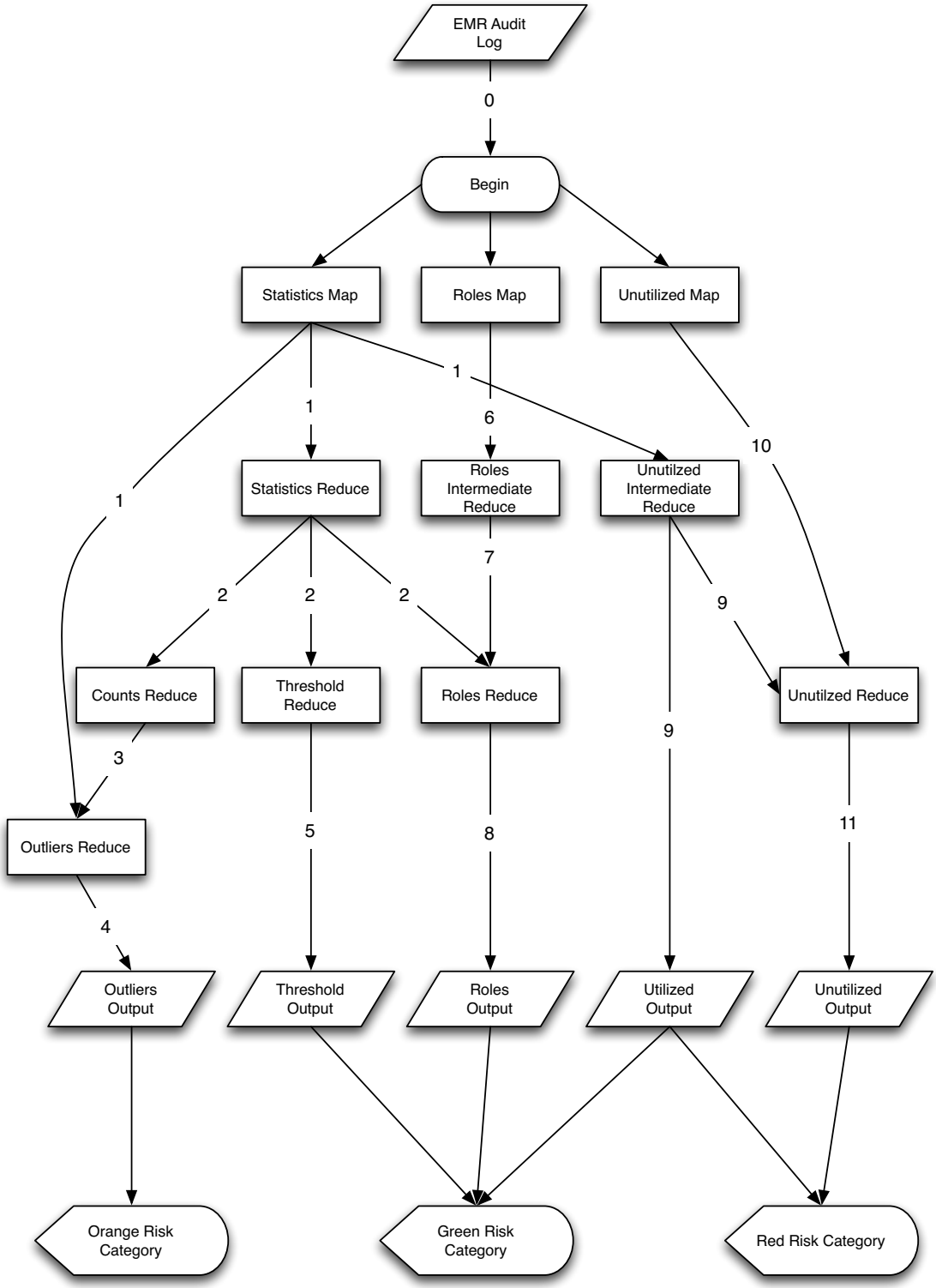


Figure 2.1: Dependency tree of MapReduce jobs.

Field
Patient Pseudonym
Patient Age
Patient Zip Code
Inpatient LOS (Days)
Inpatient Hospital Patient Location
Inpatient Encounter Pseudonym
Inpatient Encounter Start Date
Inpatient Encounter End Date
Inpatient Encounter Type
Inpatient Patient's Service
Inpatient User Reason
Ambulatory Pri. Care Phys. Pseudonym
Ambulatory User Pseudonym
Ambulatory Chart User Access Dates
Inpatient User Department Affil.
Ambulatory User Department Affil.
Inpatient User Role
Ambulatory User Role
Ambulatory Chart Access Module
Ambulatory Chart Access Action
User Inpatient Note Activity Count
User Inpatient Order Activity Count
Inpatient Chart User Access Flag
meta_load_exectn_guid
meta_orignl_load_dts

Table 2.1: Fields defined in anonymized EMR audit logs.

2.4.1 Results

All tests ran on a data set consisting of a 353.1MB EMR audit log containing 646,483 entries. This audit log contains anonymized electronic medical record data and we received IRB approval to utilize it in our analysis. Our policy engine schedules all MapReduce jobs on a specially configured Hadoop cluster running on three PowerMac G5s with OpenJDK installed. We scheduled all jobs and dependencies using the JobControl functionality offered by Hadoop.

2.4.1.1 Unutilized Mappings

The first part of our unutilized mapping test was to calculate the default policy. Since no such policy was in use at the hospital that provided us with our sample data, we calculated the ideal policy by using the Cartesian product of roles and actions: $Roles \times Actions$. We calculated a total of $1,747,200$ separate mappings, which constitutes our default policy. The second part of our test is to determine which mappings were actually utilized as observed in the audit logs. There were only $37,942$ actually utilized mappings. We determined that there are $1,709,445$ unutilized mappings. This result, shown in Figure 2.2, demonstrates that current EMR access policies are grossly over-permissive.

$$UnutilizedMappings$$
$$= PossibleMappings - ObservedMappings$$

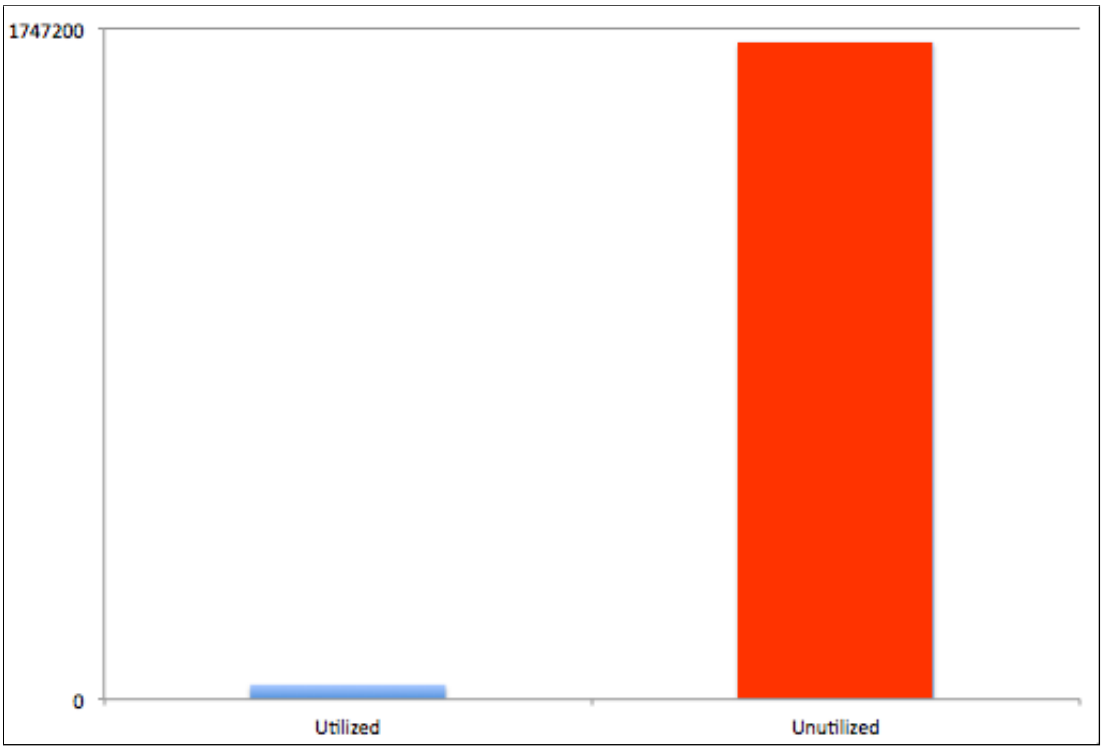


Figure 2.2: Utilized vs. Unutilized Mappings

$= 1,747,200 - 37,942$

$= 1,709,445$

2.4.1.2 Average Below Threshold

The threshold test ran over the output of our statistics job. Its output was any mapping where the average number of accesses was less than or equal to two. Through our analysis we found that there were $18,981$ such mappings out of the $37,942$ mappings that were utilized at all. From this we can see that there are actually only $18,961$ mappings that are accessed on a regular basis (i.e. not exception-based). This result is shown in Figure 2.3.

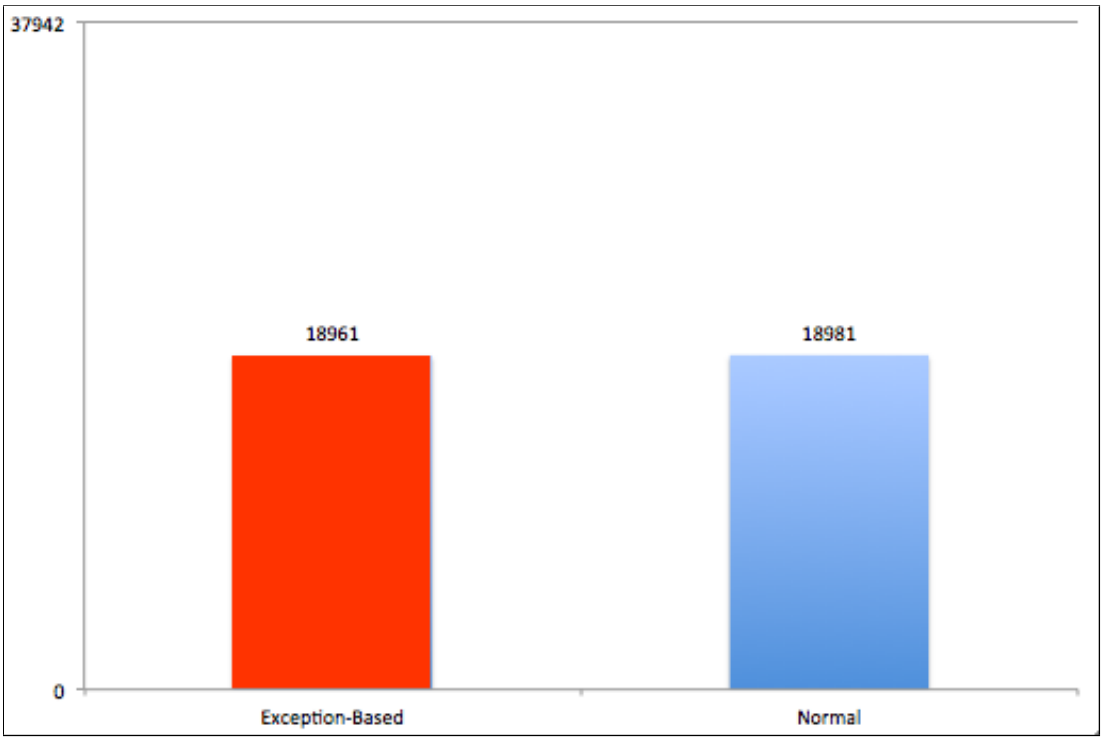


Figure 2.3: Mappings with Average Utilization Below Threshold

$NormalAccesses$

$= ObservedMappings - BelowThreshold$

$= 37,942 - 18,961$

$= 18,981$

2.4.1.3 User Low Percentage

This job computes the percentage of users with a given role that are responsible for all of the accesses that were observed for each mapping. If the total percentage

of users with a specific mapping who made an access was under 5% of all users then this was output. In order to get this information we wrote a custom mapper that computes the total number of users in each role. We aggregate this with the output of the statistics function (which indicates the number of users that made a specific access) and divide the number of users that made an access by the total number of users in a specific role. Using this we computed a total number of $6,316$ mappings where fewer than 5% of users were responsible for 100% of the accesses. These totals are shown in in Figure 2.4 Note that there is a significant overlap between the mappings output in this job and the mappings output by the threshold job–there were $5,108$ mappings that were repeated as output from both.

$$
\begin{aligned}
&NormalAccesses \\
&= ObservedMappings - LowPercentage \\
&= 37,942 - 6,316 \\
&= 31,626
\end{aligned}
$$

2.4.1.4 Abnormal Utilization of Credentials

This job is a reducer-only job. It takes the output of the statistics job and concatenates it with the output of the counts job. Using the output of the statistics job, it calculates $Average + 3 * StandardDeviation$ and $Average - 3 * StandardDeviation$.

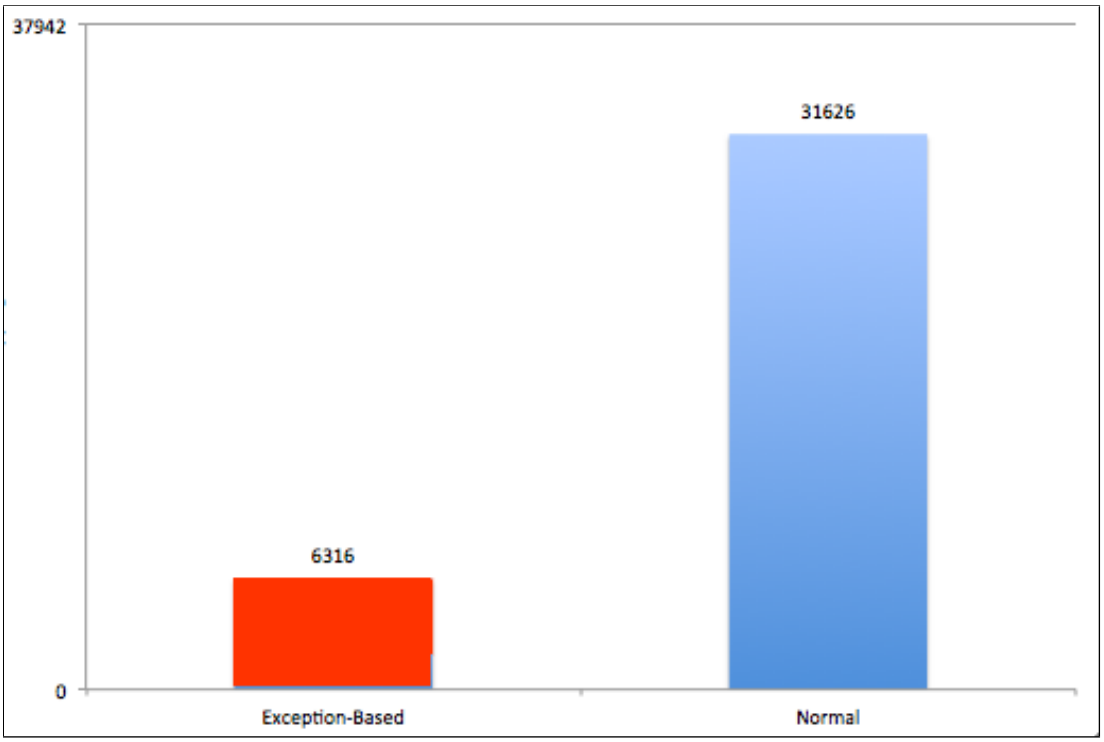


Figure 2.4: <5% of Users Responsible for 100% of Accesses

It then takes the counts from the counts job and compares them to the calculations it has made. If the count is greater than the $Average + 3 * StandardDeviation$ for the role it outputs this as an abnormal high. It uses similar means to output abnormal lows. From the dataset we have observed that there are $12,838$ individuals with a specific mapping that utilize this mapping abnormally frequently and no individuals with a specific mapping that utilize this mapping abnormally infrequently. This result is graphed in Figure 2.5. Note that there are $411,450$ unique individual accesses to the $37,942$ utilized role to access mappings in our dataset.

$NormalAccesses$

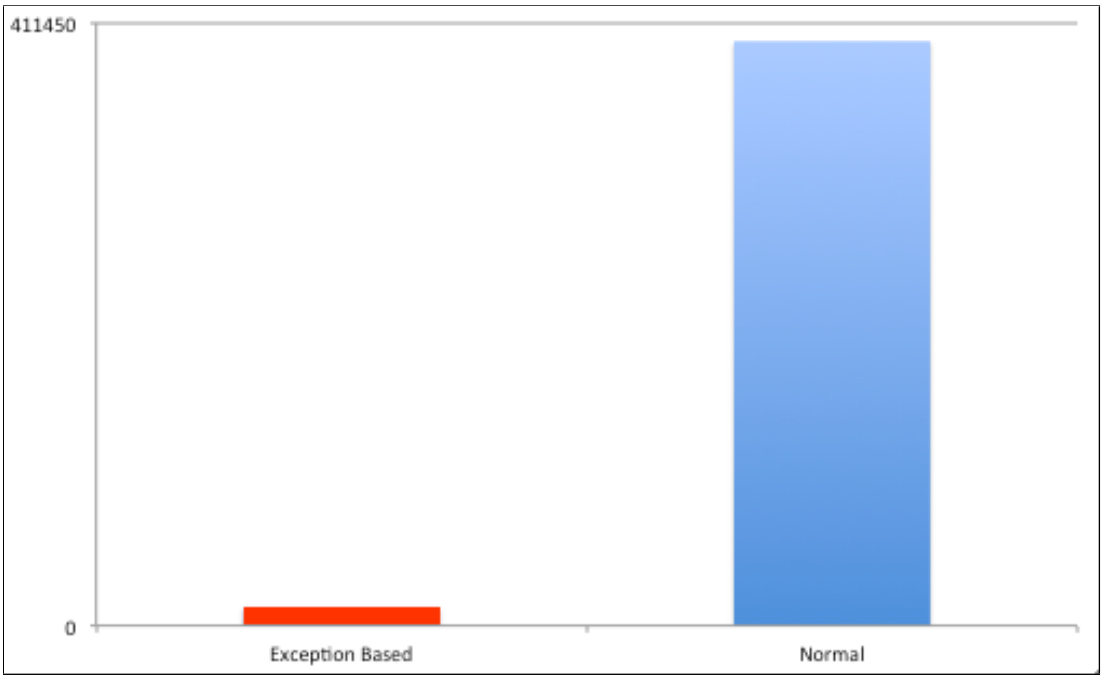


Figure 2.5: Abnormally Frequent Utilization of Access Privileges

$= UniqueObservedAccesses - Outliers$

$= 411,450 - 12,838$

$= 398,612$

2.5 Analysis

We use a table to interpret our results. We first assign colors to the various types of accesses: green for normal accesses, orange for exception-based accesses, and red for unutilized accesses. We have three risk categories for each color: high, medium and low. The risk categories are based on the percentage of accesses of a given type divided by the total number of possible mappings in the policy. For example, if 90%

of mappings were unutilized then the HIPAA compliance value would be high-risk in the red column. Note that the different colors can have different risk levels for the same data set. We have sketched out the percentage breakdown per category and risk level below:

	Green	Orange	Red
High	$< 75\%$	$> 25\%$	25%
Medium	$75 - 90\%$	$10 - 25\%$	$10 - 25\%$
Low	$> 90\%$	$< 10\%$	$< 10\%$

We compare the percentages that we yield from our previous analysis to the values in this table in order to determine our risk category for each color. This gives us three separate risk numbers. We compute define an overall risk level as the median of these three risk levels. For example, if we have:

Percentage	Risk Category
50%	Green
22%	Orange
28%	Red

We see that this corresponds to:

Percentage	Risk Category	Risk Level
50%	Green	High Risk
22%	Orange	Medium Risk
28%	Red	High Risk

Therefore, the overall risk level will be "High."

2.5.1 Risk Level for Audit Logs

In order to categorize risk we assign risk levels to each type of access. Risk levels correspond to the risk that an observed policy does not comply with the HIPAA minimum-necessary access requirement for a specific category. We also assign an overall risk level for the observed policy as a whole.

Green Category

The green category calculates mappings with an average number of accesses above our minimum threshold. We note that we have calculated $1,747,200$ possible role to access mappings in our sample data. We have determined that $18,961$ mappings had an average utilization below our minimum threshold value.

$$NormalAccesses$$

$$= UtilizedMappings - BelowThreshold$$

$- (LowPercentage \cap AboveThreshold)$

$= 37,942 - 18,961 - 1208$

$= 17,753$

Now that we have calculated our normal accesses based on our exception-based accesses we can calculate our overall percentage of normal accesses.

$\frac{17,753}{1,747,200} = .01016$

We observe that $1.016\% < 75\%$ and we therefore calculate that we are in the "High" risk category.

Orange Category

The orange category tracks all mappings which exist due to exception-based accesses compared to the total number of utilized mappings. It shows the overall percentage of all exception-based accesses.

$ExceptionBasedAccesses$

$= BelowThreshold + LowPercentage$

$- (LowPercentage \cap BelowThreshold)$

$= 18,981 + 6,316 - 5,108$

$= 20,189$

$\frac{20,189}{1,747,200} = .01156$

We therefore calculate that we are in the "Low" risk category because $1.156\% < 10\%$.

Red Category

The red category tracks the percentage of unutilized mappings. From the above data we can see that we have $1,747,200$ mappings in our ideal policy. We see that we have $1,709,258$ unutilized mappings.

$\frac{1709258}{1747200} = .97828$

We therefore calculate that we are in the "High" risk category since 97.828% of our mappings are unutilized.

Overall Risk

As stated, the overall risk is the median of our risks for the three individual categories. We see that we have risk levels of "High", "Low", and "High" and thus

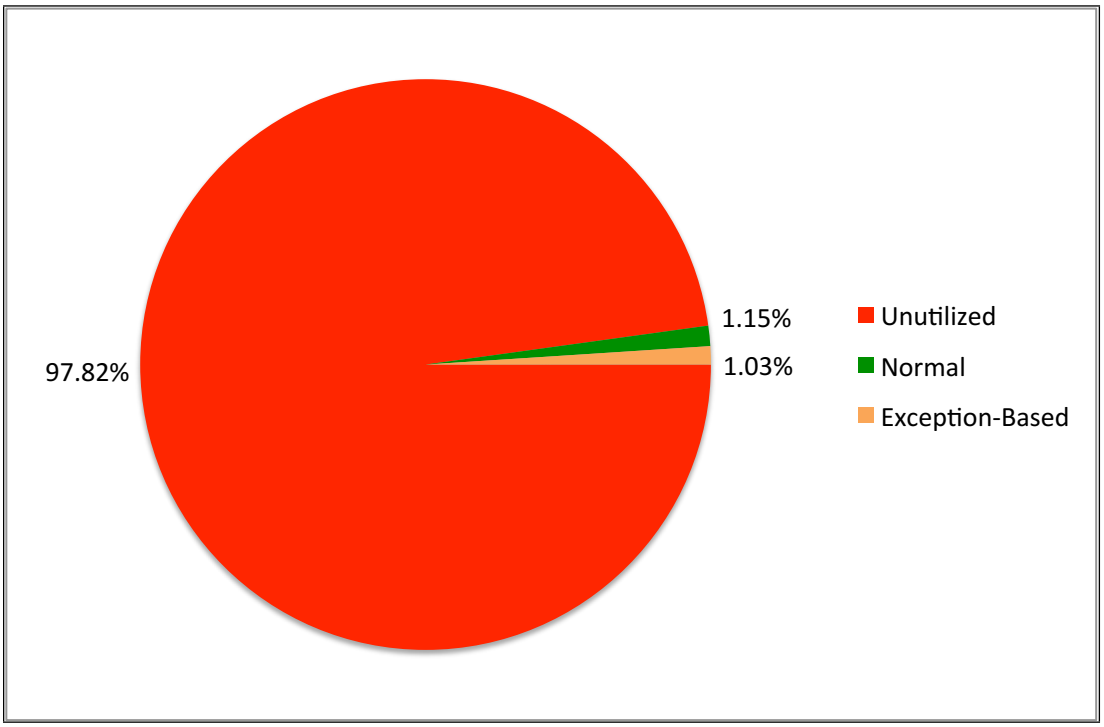


Figure 2.6: Total Access Type Breakdown

our overall risk level will be "High".

2.6 Related Work

Previous research has explored using statistical and machine learning techniques for role mining and audit.[93,94] Research has also explored anomaly detection in electronic medical record systems.[95] The work below is particularly related to our work as it has explored using audit logs to reconcile an access control policy with observed usage.

2.6.1 Privacy Management Architecture

In 2007, Bhatti et. al. propose framework called PRIvacy Management Architecture (PRIMA)[79] to compare a theoretical access control policy (P_{PS}) to the actual usage observed in EMR audit logs (P_{AL}) in order to determine theoretical policy coverage as well as to propose policy changes to P_{PS} that can help bring P_{AL} more in line with P_{PS}. This is accomplished by proposing additions to the rules in P_{PS} to account for typical accesses observed from P_{AL}. Our work is similar but builds on the previous techniques by taking the opposite approach and removing superfluous rules from P_{PS}.

2.6.2 Experience-Based Access Management

Experience-based access management (EBAM)[76] is an emerging field which separates an ideal access control model from an enforced model and uses an audit process bring the ideal model more in line with actual access patterns. Prior work[74] has successfully performed role prediction using electronic medical records. Our work extends this prior work by additionally proposing policy enhancements using the same data. Our policy engine realizes many of the goals of EBAM on a big-data scale and with a real prototype system that can audit access logs and is thus an instantiation of EBAM applied to the electronic medical record access control. By regularly using our policy engine to perform audits, an administrator can keep a policy up-to-date

based on observed, real-world usage of EMR system.

2.6.3 Explanation-Based Auditing System

The explanation-based auditing system (EBAS)[96] proposed by Fabbri et. al. audits EMR accesses based on the premise that employees in a specific role are responsible for the majority of accesses to certain types of medical records that correlate with the role. For example, the personnel within the cardiology department should logically be responsible for the majority of normal accesses to the medical records of cardiology patients. EBAS uses this principle as its basis for determining the underlying reason for an access and for categorizing the access as normal or exception-based.

EBAS is similar to our policy engine in that it uses information about role assignments to determine whether a particular access is exception-based however the mechanisms used to categorize accesses differ significantly. EBAS attempts to determine whether a particular access makes sense in the context of the underlying condition for which a patient is being treated whereas our policy engine attempts to determine if a particular access is similar to accesses made by other users assigned to similar roles.

2.7 Future Work

The work outlined here can be extended in several key ways to further increase the utility that our system provides to healthcare providers.

2.7.1 Large Scale Log Analysis

As is clear from our evaluation above, we only had access to a 300MB log with roughly $600,000$ lines of data. This log represents the number of accesses in a three month period at a large hospital. In the future we envision health information exchanges which provide electronic medical record storage and policy enforcement for all of the healthcare providers in a particular region. In such a scenario the size and scale of electronic medical record audit logs has the potential to increase dramatically. Our policy engine is built with scalability in mind. Computing the regional electronic medical record access policy based on aggregated audit logs is a useful extension to our current work.

2.7.2 Real-Time Policy Engine

This policy engine could be extended with the ability to analyze accesses in real time. In such a situation, when a user attempts to make an exception-based access an auditor can be notified when the access attempt occurs. Real-time auditing deters unauthorized accesses by allowing an auditor to closely monitor the behavior of every

employee in the hospital at all times. Such a scenario allows a user to "break the glass" when absolutely necessary while simultaneously discouraging frivolous circumventions of the policy.

The real-time policy engine would be built using a client-server architecture with an auditor having access to an administrative web application and users having access to a mobile healthcare application. As the client makes queries to the server, the queries would be evaluated and compared to the audit logs in real-time by the policy engine to determine if the accesses are exception-based. Exception based accesses will then be reported to the client and the client can determine whether or not to continue, knowing that the access will also be reported to the auditor.

2.7.3 Detecting Medical Mistakes and Prescription Fraud

Our policy engine can be modified to detect prescription fraud. The audit logs in our sample dataset contain detailed information about a patient's diagnosis and the healthcare provider actions taken as a result. This information includes a prescription and treatment history. We can look at the treatments for other patients with similar medical histories and ailments and use this information to determine whether or not a given medical treatment is exception-based. An auditor can then individually audit cases where treatment for a condition appears to be outside the norm in order to

determine whether or not a mistake has been made or fraud has occurred. We can similarly compare doctors individually to the other doctors with the same role in order to determine whether or not a doctor is prescribing treatments in a way that is atypical.

2.8 Conclusion

We can clearly see from our data taken from a real healthcare setting that in a permissive policy there are an overwhelmingly large number of mappings that are never utilized. We have developed and demonstrated a working system for separating these mappings from utilized mappings. We have furthermore developed highly scalable techniques for separating exception-based accesses from utilized mappings. We have constructed a tool to interpret and analyze these results and deployed it as a web-application capable of auditing real access logs.

Chapter 3

Classifying Network Protocol Implementation Versions: An OpenSSL Case Study

3.1 Introduction

Many Internet protocols communicate meta information about the software that is running at each end host. Included are details such as protocol version number, available parameters, software name and version, and other useful information that allows protocols to self-tune their interaction. Although the meta information included in the early messages in Internet protocols can be extremely useful, it also introduces potential security risks. Due to publicly available lists of software vulner-

abilities corresponding to early versions of nearly any well-known implementation, by announcing an implementation version number, the software is also broadcasting a set of vulnerabilities. Although exchanging protocol version number is frequently necessary to ensure interoperability, as a security precaution, many protocol implementations provide no information about the *implementation* version number. This represents a challenge to those interested in measuring and classifying Internet traffic.

Our goal in this chapter is to provide a framework and a tool for measuring the prevalence of protocol implementation versions and specific instantiations for common communication protocols on the Internet, without trusting the meta information shared by the communicating parties. Rather than believing the version numbers and implementation identifiers that are included in the traffic, we attempt to automatically infer protocol version numbers and to additionally classify network traffic according to specific implementations, based on observable implementation fingerprints.

Our approach is to use machine learning to extract features from training data for known protocol implementations and versions. Once we build a database of features for a particular protocol, X, we crawl the web, speaking X to as many protocol peers as possible. Our analysis engine then uses the database of features that was built with the training data to attempt to determine what version and implementations of X we have found. Thus, we are able to measure the prevalence of specific protocol versions and implementations on the Internet without trusting that the implementations are speaking the truth.

As a case study, we implement our system and measure the prevalence of OpenSSL versions on the Internet. We find that a small fraction (about 7%) of Apache deployments use a Linux distribution-default configuration which report the OpenSSL version used. We use this as ground truth data to train our classifiers. Our results indicate that many insecure versions of OpenSSL are in active deployment, and we believe that our research presented here has led to a much more accurate analysis of the state of SSL on the Internet than was previously possible. This case study demonstrates the utility of our general framework for measurement.

We believe that our tool would be useful in evaluating protocol library versions on closed-source devices. One such scenario where this could be the case is in hospitals. A typical hospital contains a multitude of closed-source network-connected medical devices. Administrators often have no insight into how the software on these devices is implemented. Given the current widespread security and privacy concerns in the health IT industry, administrators can use a fingerprinting tool such as the one that we propose in this chapter in order to determine how prevalent devices with security vulnerabilities are on their networks. This information can be useful in determining how to best segment the hospital network.

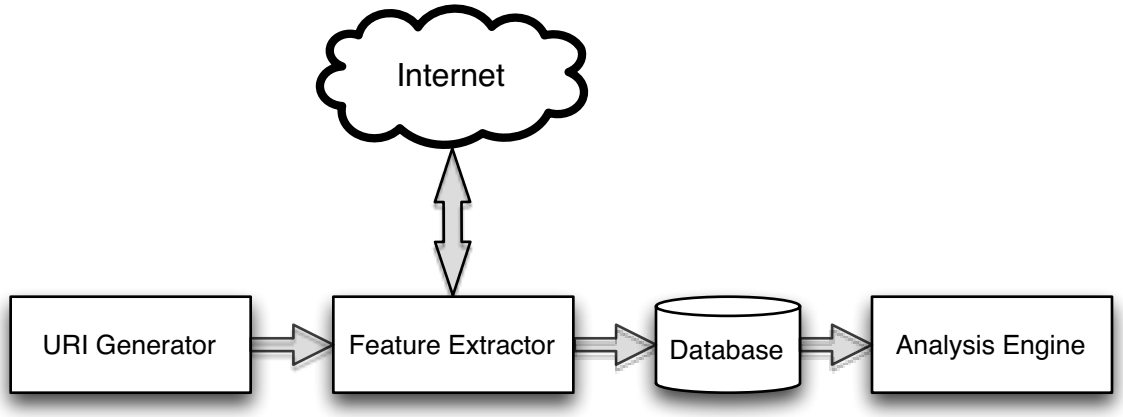
3.2 Tool design and case study methodology

In this section we give a high-level overview of our classification tool, which uses a combination of benign scanning and machine learning to identify version numbers for protocol implementations deployed on remote servers. We designed the tool using a modular architecture so that it may be used in a variety of Internet measurement studies. The end result is a tool which is useful for classifying protocol implementations for which we have some labeled data. For our SSL/TLS case study, we rely on web servers which are configured to report their version number.

3.2.1 Architecture

Our classification tool consists of four components: a URI generator, a feature extractor, a database, and an analysis engine. Figure 3.1 shows the components and the flow of information through the system. The URI generator, feature extractor and classification algorithm are modular and can be easily replaced by new modules for different tasks. We first describe each component in the abstract; then we describe our concrete instantiation for classifying OpenSSL versions.

Figure 3.1: Architectural diagram for the classification tool.



3.2.1.1 URI generator

The URI generator's job is to produce a set of URIs to be consumed by the feature extractor. This modular component can be as simple as a static set of URIs produced one at a time or as complex and dynamic as may be required for the particular use case. The same generator can be used for a variety of classification tasks. Examples of generators include a generator that produces the Alexa top 500 sites,[97] a generator that walks through the entire IPv4 address space, and a web crawler that starts at a fixed set of web sites and recursively crawls links it encounters. We use the last example in our SSL/TLS case study.

3.2.1.2 Feature extractor

The feature extractor takes as input a URI, interacts with the network server pointed to by the URI, and produces a feature vector for an arbitrary set of features.

For example, the feature extractor can fetch web pages, query the server for configuration options, or engage in aggressive probing such as that performed by standard network measurement tools like nmap. The feature extractor is necessarily specific to the classification task; however, the modular nature of our architecture makes swapping out the feature extractor for a different implementation easy.

3.2.1.3 Database

After the feature extractor has produced a feature vector, it is inserted into a generic relational database.

3.2.1.4 Analysis engine

The heart of our system is the analysis engine which runs one of a variety of machine learning classification algorithms on the data in the database. The analysis engine heavily leverages the Python Orange library,[98] an open source data visualization and analysis tool, to first cluster and then classify the data. The analysis engine assumes that some subset of the data has labels.

We used a semi-supervised learning approach to classify the remaining, unlabeled data. Since there are likely to be a large number of protocol implementation versions (e.g., for our OpenSSL study, we found 79 distinct versions), we use k-nearest neighbors (kNN) as our classifier. The downside of kNN is that its accuracy depends heavily on the similarity metric used. To mitigate this, we first perform a k-means

clustering and then run kNN for each cluster.

Although we use k-means and kNN, they are not the only possible choices. Some applications may perform better with other choices and thus this is a configurable option. We experimentally validated that k-means and kNN are good choices for our use case among the set of algorithms supported by Orange. Another option which may perform well for our use would be a multiclass logistic regression since many of our features are binary or categorical.

3.2.2 OpenSSL version instantiation

To be useful as a classification tool, the URI generator and feature extractor need concrete instantiations.

3.2.2.1 URI generator

Our URI generator is implemented as a web crawler that traverses HTTP links and recursively looks for any new HTTPS links to pass to the feature extractor. To determine where to start our crawl, we first identified several websites that contain a large number of external links to many different domains. We pointed our crawler at several of these sites, including `http://www.cnn.com` and `http://news.ycombinator.com` and ran it until we had collected a set of about 123,000 websites. We believe this set to be a representative sample of the sites frequently visited by average users because the sites are all within a few degrees of separation from popular websites.

3.2.2.2 Feature extractor

For each URI produced by the URI generator, the feature extractor initiates a secure connection to the server and interrogates it to learn a variety of features including:

- supported SSL/TLS versions and ciphersuites;
- secure/insecure renegotiation support;
- session resumption support;
- CA, validity, issue date, expiration date;
- TLS compression support;
- TCP/IP stack information;
- HTTP headers;
- web server configuration.

Some of the features we discovered had surprising characteristics. For example, we discovered that the server response HTTP header was reported by 81.7% of the servers in our data set. 7.8% of those server responses contained a number of useful features such as the version numbers of most of the Apache software stack including Apache modules like mod_perl and mod_php and, importantly for us, OpenSSL. These version numbers form the labels that the analysis engine uses. The server response is described in more detail in Section 6.6.

We construct our feature extractor by extending version 0.4 of iSECPartner's SSLyze tool.[99] SSLyze provides methods to enumerate all cipher suites for SSLv2,

SSLv3, TLSv1.0, TLSv1.1 and TLSv1.2. In addition, SSLyze provides methods to enumerate session resumption support, to scan for insecure session renegotiation, and to collect information about site certificates (e.g., expiration date, issuing certificate authority, and certificate signature validation).

We implemented the following extensions to SSLyze via new plugins: TLS compression support detection (i.e., to determine vulnerability to the CRIME attack[100]); HTTP headers collection generated by website navigation; server response string parsing and TCP/IP configuration information collection.

3.3 SSL/TLS background

Secure Socket Layer (SSL/TLS)[101] is perhaps the most important security protocol on the Internet. Although SSL/TLS has many applications—including Virtual Private Networking and inter-application communication—it is most commonly used to secure web traffic served via the HTTPS protocol. The relevance of TLS and HTTPS has increased in recent years, as many websites (including Gmail, Facebook and Twitter) have begun to deploy HTTPS by default.[102–105]

While TLS is widely deployed across the Internet, there are only a small number of popular implementations. When considering HTTPS, the most common are OpenSSL (used by a majority of Apache deployments) and Microsoft's SChannel. Results from the February 2013 Netcraft survey indicate that Apache servers dominate IIS

implementations across surveyed Internet domains,[106] indicating that OpenSSL is likely the most popular TLS implementation on the Internet.

In fact, given the widespread dependence on OpenSSL, it is reasonable to say that for many users, OpenSSL *is* TLS. This should draw attention from the security community as there are numerous versions of OpenSSL in current deployment, many of which include serious protocol vulnerabilities.[107–109] If an attacker can determine the version of OpenSSL deployed on a specific webserver, he may be able to seriously compromise the effectiveness of any TLS connection made to that site.

Surprisingly, we have very little information on the deployment of OpenSSL library versions, because most web servers do not advertise this information. Even if an Apache server version is known, this does not necessarily imply that the server is using a known version of OpenSSL: in many systems, the installed OpenSSL version is determined by various factors, including the particular Apache and OS distribution, the presence of other software on the system, and (most commonly) server misconfiguration. This can lead to the widespread deployment of old, broken versions of the library.

3.3.1 SSL/TLS (in)security

Recent attacks on SSL/TLS, such as BEAST[110] and CRIME,[100] demonstrate the sensitivity of SSL/TLS security to configuration choices made by server administrators. We briefly describe several protocol and implementation vulnerabilities below.

Many of these were identified by the SSL Pulse project.[111]

3.3.1.1 SSLv2 support

SSLv2 is known to have many vulnerabilities which make it unsuitable for use in secure communications. Although SSLv2 is considered so insecure that major Linux distributions don't even build OpenSSL with support for it anymore, the SSL Pulse data estimates that 28.4% of SSL servers on the Internet today support this version of the protocol, more than the number of sites that support TLSv1.1 and TLSv1.2 combined (9.2% and 11.4% respectively).[111]

3.3.1.2 Insecure session renegotiation

In 2009 a practical attack on SSLv3/TLSv1.0 was proposed that exploited flaws in the session renegotiation feature allowing for a man-in-the-middle attack. The vulnerability allows an attacker to queue an HTTP command to be executed by the server on behalf of the client immediately when a client makes an SSL connection.

3.3.1.3 Insecure CBC mode ciphersuites

Several recent attacks have illustrated flaws in the implementation of CBC mode encryption within SSL and TLS. These flaws stem from two basic causes: the improper use of initialization vectors[110,112] and flaws related to TLS's insecure MAC-then-encrypt approach to authenticated encryption.[108,113] The latest of these vul-

nerabilities[108] was patched in February, 2013, hence these attacks remain an active concern. Similar flaws are also present in Datagram TLS, and were recently exploited by AlFardan and Paterson.[114]

3.3.1.4 TLS compression support

A general class of attacks exploiting protocol vulnerabilities due to TLS compression and SPDY. While this mode of operation has been recognized as insecure for a number of years,[115] the recent CRIME attack demonstrated the feasibility of using this vector to attack critical information such as session cookies.[100]

3.3.1.5 Software vulnerabilities

The OpenSSL implementation of TLS and SSL has been subject to numerous software vulnerabilities. These include dozens of implementation flaws with potential consequences ranging from denial of service to remote code execution.[116]

One of the results of our research is the ability to identify versions of OpenSSL in use on the Internet. Since many versions of OpenSSL contain known, severe vulnerabilities, such as information leakage and remote code execution,[117–119] by examining a large, representative sample of servers, we get an indication of what fraction of servers in use today are vulnerable.

3.4 Results

All tests described below were run on m2.4xlarge Amazon EC2 instances with 64 GB of RAM and four cores running Red Hat Enterprise Linux 6, PostgreSQL 8.4.12, Python 2.6.6 and Orange 2.0b. The tests were extremely memory and CPU intensive and even with these sizable resources our tests took on the order of hours to days to run on a test data set containing 123,345 scanned sites.

3.4.1 Prediction accuracy

As described in Section 4.4, our analysis engine takes a semi-supervised learning approach to version identification. To determine how accurate our classification is, we perform a 10-fold cross validation on our training data which consists of the self-reported OpenSSL versions. Interestingly, we find that if we exclude *only* the OpenSSL version from the Apache server response HTML header, we can identify the OpenSSL version with overwhelming probability by using the reported version numbers of Apache modules like mod_php and mod_perl; however, this result is misleading. Among all 9,653 of the scanned web sites which report module versions numbers, only 615 do not report an OpenSSL version. We hypothesize that this is due to the use of a different SSL/TLS library such as GNU TLS. Thus, when testing accuracy, we omit the entire server string from each 10% test set.

Because the accuracy of kNN is so dependent on the similarity metric, we first

Version Component	$k = 10$	$k = 20$	$k = 30$
major.minor	91.6	94.3	93.4
fix	75.5	74.0	75.7
patch	50.2	49.6	54.7
distro-build	49.6	49.2	54.7

Table 3.1: Percentage of OpenSSL version components correctly predicted for k clusters. The percentages in each row are the percentages of correctly predicting the version components up to the given component.

cluster similar points and run kNN on each cluster (see Section 4.4). The accuracy of our classification (weakly) depends on the number of clusters we use. We test with $k = 10$, 20, and 30 clusters. For each cluster, we use $k = 10$ neighbors to classify each test point. OpenSSL versions are split into: major, minor, fix, patch and (optionally) distro-build versions.[120] For example, an OpenSSL version running on Fedora Linux might be openssl-1.0.0a-fedora1. The major version would be 1, the minor 0, the fix 0, the patch a and the distro-build fedora1. For OpenSSL, there are essentially two major.minor version combinations in use, 0.9 and 1.0. Therefore, we tread major and minor versions as a single version component. Table 3.1 and Figure 3.2 show the percentage of versions that can be identified correctly.

3.4.2 Observed OpenSSL versions

Since we are focused on classifying OpenSSL versions, we need to remove data points which do not correspond to OpenSSL versions but come from Microsoft SChannel, GNU TLS, or other SSL/TLS implementations. The first step of the analysis

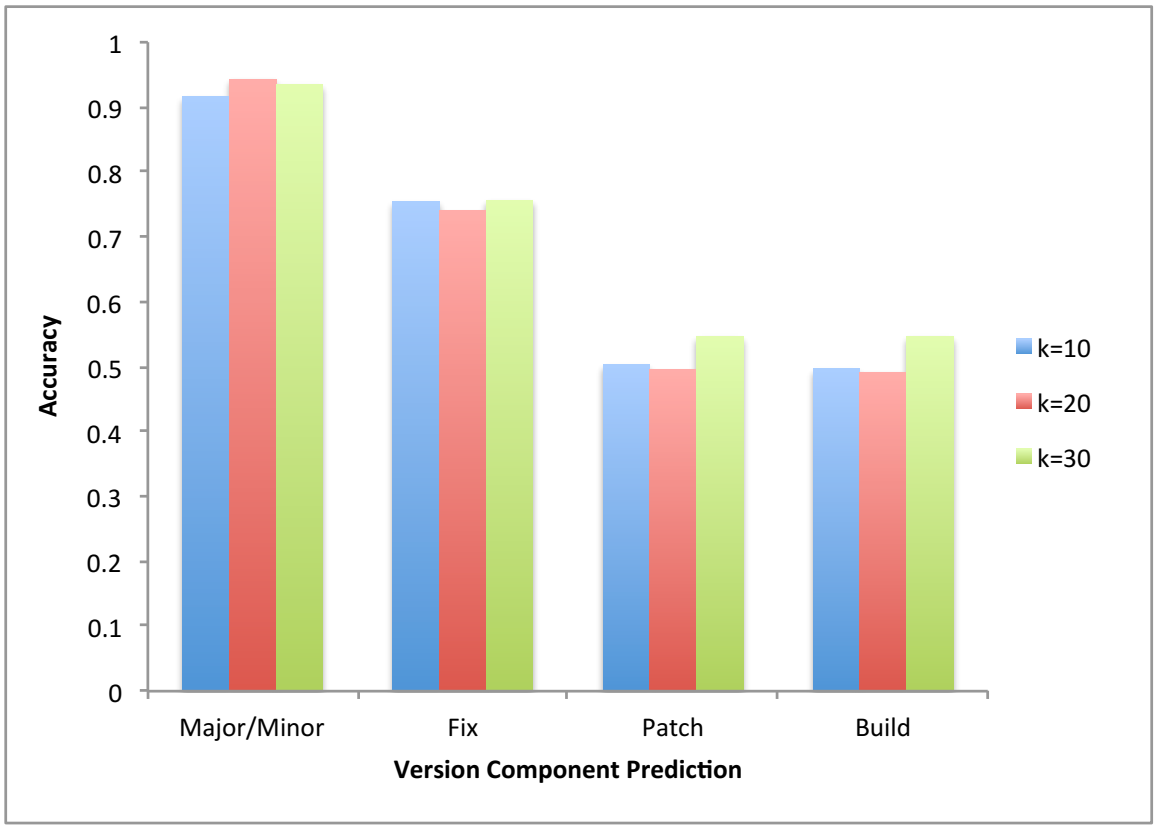


Figure 3.2: Percentage of OpenSSL version components correctly predicted for k clusters.

engine is to perform k-means clustering. This produces clusters of data whose feature vectors lie close together in the feature space. We hypothesize that OpenSSL data points are likely to cluster together (although not necessarily in a single cluster) and that other implementations are not likely to lie in clusters with (many) OpenSSL data points. To that end, for each cluster, we examine the number of data points that are labeled and throw out any cluster which does not contain at least 10% labeled data.

After removing clusters with too few labeled data points, we have 61,832 unlabeled data points. From this we can give a rough breakdown of popularity of OpenSSL

Version	Percentage
0.9.8e-fips-rhel5	37.25
0.9.8g	14.50
0.9.7a	7.02
0.9.8o	4.76
1.0.0-fips	4.36
0.9.7d	2.91
0.9.8n	2.75
0.9.7e	1.94
0.9.8c	1.80
0.9.8m	1.74
0.9.8e	1.72
0.9.8r	1.71

Table 3.2: Most popular OpenSSL versions on the Internet.

versions on the Internet, Table 3.2. In the following section, we discuss the implications of these results in the context of vulnerabilities present in different versions of OpenSSL.

3.4.3 Vulnerabilities

From the OpenSSL website's known vulnerability report section[116] we see that 75.9% of known vulnerabilities apply to five or more OpenSSL versions within the same patch family. Thus if the OpenSSL implementation can be accurately fingerprinted to within a patch family then there is a high probability that at least some of the vulnerabilities reported for the guessed OpenSSL version will apply even if the guess is incorrect.

Based on the results in Table 3.2 and data from the OpenSSL vulnerability re-

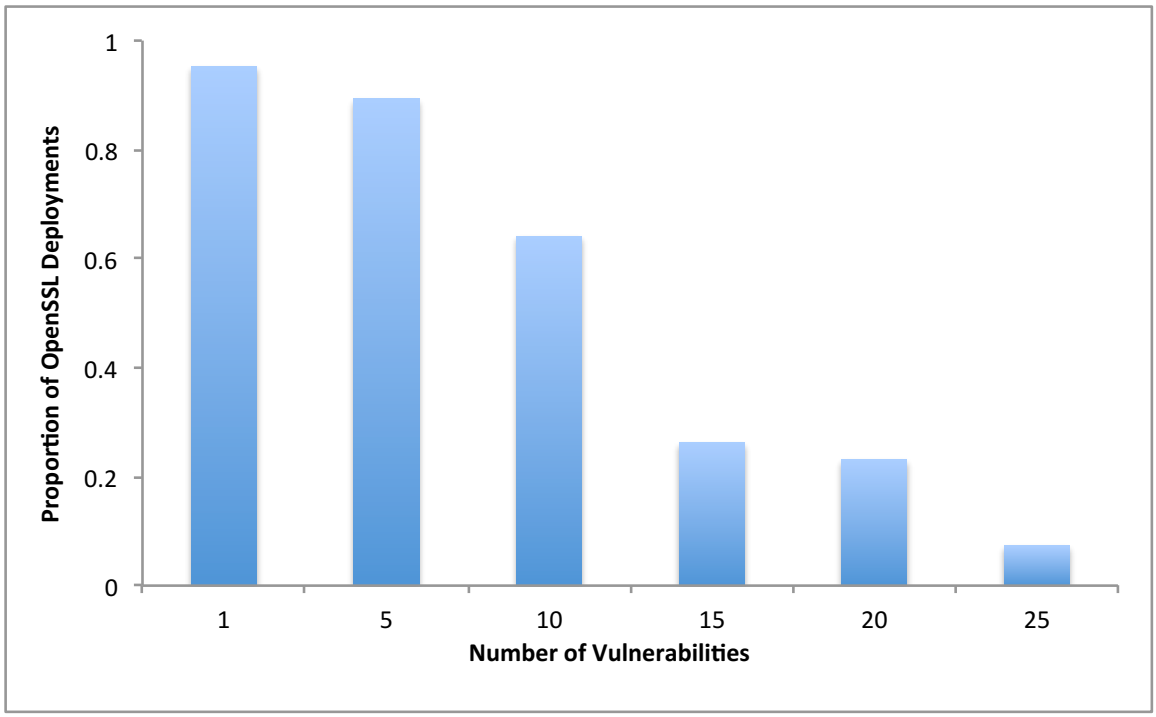


Figure 3.3: Percentage of OpenSSL deployments with at least n unpatched vulnerabilities

Distribution	OSSL Version	CVEs
Debian Squeeze (6.0)	0.9.8o	11
Debian Lenny (5.0)	0.9.8g	24
Debian Etch (4.0)	0.9.8c	26
RHEL 6	0.9.8e/1.0.0-fips	0/14
RHEL 5	0.9.7a/0.9.8e-fips	14/0
RHEL 4	0.9.6b/0.9.7a	9/14
Fedora 18	1.0.1c	3
Fedora 17	1.0.0i	3
Fedora 16	1.0.0e	9

Table 3.3: Default OpenSSL versions shipping with popular Linux distributions.

port,[116] we have computed the number of reported vulnerabilities for each OpenSSL version that we have predicted using our analysis engine. Our results show that 95% of all deployed OpenSSL versions have at least one known CVE that hasn't been patched by OpenSSL (but that may have been patched by an individual distribution vendor). 64.12% of all deployed OpenSSL versions have more than 10 CVEs. The complementary CDF of vulnerabilities is presented in Figure 3.3.

One of the least surprising findings of our survey is that most users are running the OpenSSL version included in their Linux distribution and, in many cases, users do not keep their Linux installations up to date. We will discuss potential reasons for this in the next section. Table 3.3 includes a list of three of the most popular Linux distributions that we found in our crawl as well as the default OpenSSL version that each shipped with.

We note that many of the most popular OpenSSL versions in Table 3.2 are versions that shipped in the popular Linux distributions in Table 3.3. This table indicates that

most of the default OpenSSL versions included in shipping Linux distributions have several known vulnerabilities. It is important to note that Linux distributions may patch some of these vulnerabilities on their own, though manufacturers usually do not patch all vulnerabilities. This is because many vulnerabilities are discovered after support for a given version is no longer being supported. Our training data is unable to take these patch levels into account for many distributions (such as Debian). This is a fundamental limitation of our data set and of the training data available. As a result, the number of CVEs filed against an OpenSSL version form an upper bound of how many known vulnerabilities might exist, but in practice the number of outstanding vulnerabilities could be lower. However, simply discovering the base version of a particular OpenSSL server raises the possibility of it being vulnerable to a known attack.

3.5 Discussion

The results of our SSL/TLS case study validate our modular framework and general approach for identifying and classifying versions of network protocol implementations. Because our framework treats the modular components (the URI generator, feature extractor, and classification algorithms) as black boxes, we can plug in different modules to study other protocol implementations or a different swath of the Internet with a minimum of effort. In the future, we plan to do exactly this (see

Section 3.9).

In the rest of this section, we discuss the significance of these results and address some of the questions that they raise.

3.5.1 Severity of vulnerabilities

Our results indicate that many of the OpenSSL servers on the Internet are vulnerable to implementation-specific exploits described in CVEs cataloged by the OpenSSL project. Our first consideration is the severity of the vulnerabilities. Of the 54 vulnerabilities, four can lead to remote code execution, fourteen to a DoS and seven to information leakage. We observe that types of vulnerabilities range in severity from moderate to catastrophic. We therefore assert that not only are these OpenSSL servers vulnerable to many of the protocol attacks as described by the SSL Pulse project[111] but that they are also widely unprotected against implementation-specific attacks that could, in some cases, not only compromise the privacy of an existing SSL session, but also the security of the underlying server.

3.5.2 Distribution-specific patches

We have also observed that many Linux distributions release package updates in fixed-length cycles. The versions of all major libraries, including OpenSSL, are fixed at the end of distribution's release cycle. This prevents all further package updates,

including security patches, that the library vendor releases from being automatically integrated into new builds of the package. If a major vulnerability is discovered an individual distribution's package maintainer may elect to backport the patch and release it as part of a security update for that distribution, but this is not guaranteed.

This process is problematic for several reasons. First, the package maintainers responsible for backporting patches cannot necessarily provide the level of scrutiny toward patch as the library maintainers can. Also, many of the CVEs are released after a distribution has stopped releasing security patches for an outdated, but still widely deployed release. This results in mission-critical systems being left connected to the Internet without any ability to receive security patches. To illustrate this point, our analysis has revealed that Debian Lenny which officially supports OpenSSL version 0.9.8g is still widely deployed on the Internet. This particular version has 24 known vulnerabilities, 17 of which Debian patched before it discontinued support for Lenny. The remaining vulnerabilities will never be patched. We found that 0.7% of all of the servers that we crawled self-reported that they run Lenny in the server response header. An attacker viewing this information will immediately know that this server must be running unpatched versions of many libraries and daemons that may be vulnerable to remote code execution exploits.

3.5.3 TLS1.1/1.2 deployment

The only OpenSSL branch that supports TLS1.1/1.2 is the OpenSSL 1.0.1 branch,[120] which, as shown in Table 3.2 is not widely deployed. We believe that administrators are hesitant to upgrade to the 1.0.1 branch of OpenSSL because most Linux distributions have not yet released packages for it. To further complicate matters, the OpenSSL, mod_ssl, and Apache versions are interdependent making upgrading one without the others difficult. These complications prevent many administrators from upgrading their web server stacks on their own. Since most Linux distributions do not include the OpenSSL 1.0.1 branch, a large fraction of Apache servers cannot support the most recent versions of TLS, even if their administrators wanted to. This observation is corroborated by the SSL Print's observations of TLS1.2/1.2 market penetration during the same time period as our crawl[111]

3.5.4 Case study conclusions and future work

The current state of OpenSSL deployment on the public Internet is not good. By actively interrogating web servers, our tool discovered close to 62% of web servers using OpenSSL are running versions which contain known vulnerabilities that are susceptible to exploitation. This is explained, at least in part, by the fact that many web server stacks use the default version of OpenSSL provided by the operating system which can become stale if not kept up-to-date.

In the future, we plan to explore the nature of biases in our training data by leveraging additional classification techniques on additional features such as operating system version. We believe that the additional data labels will provide insight to our current semi-supervised approach. In addition, we plan to explore avenues for acquiring better training data for additional TLS implementations.

3.6 Limitations of data

The approach that we use for collecting our training data is a good approach for fingerprinting OpenSSL implementations but it has several limitations. The major limitation of our approach comes from our analysis making the assumption that the observed labels (the OpenSSL version numbers) come from web servers that accurately report them. We believe that this is likely to be the case since Apache programatically generates this information when it is run rather than being hard coded in a configuration file. A hard coded value would run the risk of not being updated when OpenSSL is updated and thus the server would report old version numbers. Another concern is that a clever administrator might try to spoof these labels to throw off would-be attackers. While this is a concern, we did not observe any cases where observed features did not match the reported OpenSSL version. Additionally, there is the possibility that our training data may be skewed towards servers that are configured with default settings or towards servers that are infrequently updated.

Both concerns can be addressed by starting with a better set of data for which we have ground truth.

We are also limited by the fact that we are only able to collect training data for OpenSSL and thus cannot identify other SSL/TLS implementations such as GNU TLS or Microsoft SChannel. As a result, we were forced to rely on clustering techniques to identify likely OpenSSL implementations from among the servers we crawled. This introduces both false positives—that is, SSL/TLS libraries we classify as being OpenSSL when they are not—and false negatives—throwing out OpenSSL data points because the clusters do not contain enough labeled instances. Since we again do not have ground truth, we cannot measure the accuracy of this binning technique. More complete training data would remove this limitation.

Finally, we were only able to scan web servers on the public Internet. Thus, our results do not generalize to all OpenSSL deployments such as those on corporate LANs or behind NAT gateways, an important problem in its own right due to the presence of malware on these networks resulting from, among other things, bring-your-own-device corporate cultures.

3.7 Generalizing the approach

We designed SSLPrint using a modular architecture in order to facilitate its usage in other studies. One particular use case for this functionality is to enable the finger-

printing of other security protocols. There are a few requirements in doing so. The primary requirement is that if the default analysis tool and techniques are to be used then the protocol must have some mechanism of reporting the actual version number for some percentage of the data. This is necessary because our analysis is contingent upon supervised learning techniques which require labeled training data. Whether or not there is a means to collect such data is wholly dependent upon the individual protocol being analyzed.

We note that the accuracy of SSLPrint may vary when used to analyze a different protocol since this information is dependent upon the number of features that are dependent upon the underlying version number of the protocol being analyzed and on how predictive these features are of this version number. We do note, however, that SSLPrint will report accuracy data when run. If a given protocol has many observable features that closely relate to version number then the protocol will be a good candidate for this sort of analysis.

3.8 Related work examining SSL/TLS

Much of the SSL security research to date considers the quality of certificates deployed across websites, as well as server-based configuration options. Community driven projects such as the EFF SSL Observatory and Netcraft SSL Survey have been erected to provide an in-depth analysis of server certificates on the web. More

concretely, the EFF SSL Observatory aims to determine the trustworthiness of Certificate Authorities by investigating multiple features of all server certificates on the web (e.g., number of certificates signed by an authority).[121] The Netcraft SSL Survey is a monthly data collection service that attempts to identify how online businesses use encryption to secure their online transactions (i.e., confirming known certificate usage and deployment).[106] In 2012, Heninger et al.[122] performed an Internet-scale analysis of all SSL certificates. The intent of the analysis was to determine how many servers had weak or insecure key generation mechanisms. The end result was the startling realization that 5.57% of deployed SSL servers shared an RSA key factor.

In the past few years there have been several high profile attempts to survey SSL servers that are vulnerable to the known protocol attacks with varying areas of focus.[99,123–125] In contrast, we take an entirely new approach to analyzing the security of SSL servers by looking at individual, implementation-specific vulnerabilities rather than looking for protocol-specific ones. In doing so we classify the OpenSSL implementations of a large sample of servers and chart these implementations against known version-specific vulnerabilities for which CVEs exist. We show that a large percentage of servers are running OpenSSL versions for which there are many unpatched vulnerabilities. In many cases, these vulnerabilities are severe and in some cases lead to total privacy breaches or remote code execution.[117–119]

Finally, recent work attempts to fingerprint SSL *library type* (e.g., SChannel vs. OpenSSL) by actively probing TLS handshake responses[126] for a list of known error

responses. While this tool, SSLAudit, can distinguish different libraries, it is not able to determine library version.

3.9 Conclusions and future work

We have shown that machine learning techniques can be effective as a means of classifying Internet communication based on the implementations that generated the traffic. Identifying specific implementations is extremely useful for measuring and analyzing security. For example, in our case study looking at SSL/TLS, we discovered that more than 62% of the installations running OpenSSL deployed versions that are known to be susceptible to published exploits.

Our case study discovered that default configurations are typically maintained, and we know that these often become stale quickly. Studies such as ours can be used to identify the prevalence of patch applications on the Internet, and we believe that our technique can be easily applied to other protocols. We plan to continue to enhance our machine learning techniques and to implement additional case studies to study the implementations of other security-sensitive protocols such as SSH or DNS. One objective of this work is to provide measurement tools that analysts can use to discover how widely patches have been applied to different software packages. We also hope our work will provide generic tools for researchers who need to learn about the use of particular implementations of protocols.

Chapter 4

Sentinel: Secure Mode Profiling and Enforcement for Embedded Systems

4.1 Introduction

With the advent of the Internet of Things, today's embedded devices have become increasingly connected.[127–129] One of the major benefits of this connectivity has been realized in industrial networks, where device measurements can now be taken remotely. Gone are the days when technicians would carefully transcribe analog readings produced by disparate equipment onto paper charts. Modern industrial networks leverage technology in order to minimize the number of times a technician has to in-

teract with paper or perform a manual reading. Many industrial networks contain a mix of general-purpose computers and mission-critical embedded devices.[130–134]

These devices are often installed directly by an agent of the manufacturer or by field technicians specifically trained by these agents. There are numerous examples of embedded devices containing special modes that are meant to be used only by these agents during setup.[135,136] Unfortunately, manufacturers still produce devices that do not have secure default configurations[137–139] and in many cases these modes can also be used by attackers to maliciously reconfigure and attack the device.[138,139]

In addition to being vulnerable to remote attacks like any other network-facing device, many embedded systems are also particularly vulnerable to physical attacks as well. Industrial equipment must often be deployed in the field. In such an environment, the equipment owner typically has little control over who interacts with the devices. Until recently, most manufacturers have focused on reliability and usability rather than on security. There are many possible negative consequences of an attacker tampering with industrial equipment–the attacker could degrade performance, tamper with measurements, disable functionality or, in some cases, even cause direct bodily harm.[140,141] Clearly physical security in industrial networks is a serious problem that does not always receive the attention that it deserves. When we consider physical attacks in conjunction with remote attacks, we see that many embedded devices have enormous attack surfaces.

To further complicate matters, many embedded devices are designed to have ex-

traordinarily long lifespans. For example, a medical infusion pump model might be supported by its manufacturer for 10–15 years.[142] This creates several problems. As security techniques constantly improve, legacy devices are often left behind. In some industries, such as healthcare, embedded devices may not even be able to receive software upgrades without the update first going through a lengthy and expensive review process.[143] Due to this review process, manufacturers sometimes recommend that a particular feature not be used without releasing a firmware update that actually removes the mode.[144] Furthermore, since industrial devices are designed with longevity in mind, they are often designed to be sold to as many customers as possible. This means that the devices often contain a wide variety of modes and features in order to meet the requirements of all possible customers.

As a result, any particular customer may only use a small subset of the available features in a given device; in many cases actual usage of the device may fit a very narrow profile. Despite the extensive work looking at securing embedded devices against exploits[145–155] there has been little work on how to build and enforce security profiles that cover a device's typical usage patterns, and how to disable insecure device features without requiring a firmware update.

This new work would be a useful complement to existing work on defense against exploits, as it would allow a manufacturer to dramatically reduce the attack surface on an embedded device by allowing the device to be locked down to a limited profile after it has been configured. For example, on devices with a configuration mode,

it would be useful to disable [1] the configuration mode while the device is deployed in the field if the device should only be reconfigured when it is offline.[68] Similarly, on devices that use interfaces for debugging purposes, it would be useful to disable these interfaces when the device is not being debugged.[156–158] On legacy devices with telnet access, it may be useful to disable telnet altogether.[159–161] Note that these three examples all come from vulnerabilities reported in actual industrial embedded devices. Existing exploit mitigation techniques would not adequately address these issues but a secure device profiler such as Sentinel could.

We address this problem by creating Sentinel, a secure device profiler for embedded systems. Sentinel significantly increases the physical security of embedded devices by using a bus tapping interface to derive a partial control flow graph representing a subset of functionality of the attached embedded system. This control flow graph is built from device execution traces taken while running the device through the desired functions. Anything in the control flow graph is considered an allowable action within the device's security profile and any action not in the security profile is considered a security violation.

The partial control flow graphs corresponding to constructed security profiles can replace a full control flow graph in existing bus snooping-based control flow integrity techniques[145–147] with only minimal modifications in order to detect attempts to access features not in the deployment profile of the device. Sentinel builds its partial control

[1] Disabling a device feature is one of several possible mitigation techniques that Sentinel can be made to support. These techniques are discussed in more detail in Section 4.4.4.

flow graph by monitoring the memory bus directly. It is therefore a passive observer to normal device operations. Because of its passive nature, Sentinel is able to accomplish this auditing without any additional runtime overhead whatsoever. Furthermore, Sentinel is robust even in the face of lossy data.

We outline a high-level design for an execution-based embedded device profiler. We prove its feasibility by using it to build a security profile of a particularly important type of embedded device–a medical infusion pump. We test the utility of the profile by using it to audit an execution trace taken during a physical attack on the device. We further outline how we solve difficult problems such as accounting for interrupt and exception handlers and anticipating instruction prefetches before looking how it handles lossiness by looking at the relationship between sample size and false positive rate. Finally, we consider additional uses for Sentinel beyond our novel security profiling technique.

Threat Model. This work is concerned with attackers that already have some access to an embedded system. The attackers may be physical or remote. Their goal is to attempt to access a feature or a mode of the device that is not within its use profile without detection.

Our Contribution. This work proposes the following significant and novel contributions:

1. Dynamically builds partial control flow graph corresponding to device features and modes without any knowledge of the underlying source code or firmware

2. Audits execution traces to ensure that device operated within a given security profile

3. Assists with disassembly of specific device features and modes

4.2 Related Work

There has been much previous work looking at memory bus monitoring in embedded devices.[145–147,162–165] Some of this work has used bus monitoring in conjunction with other techniques to add security to insecure embedded devices[146,162,163] including by using a modified compiler to output a full control flow graph to be used in conjunction with a bus monitor to enforce control flow integrity.[145–147]

All control-flow related bus monitoring work that we know of has looked at using the full control flow graph to detect and prevent remote exploits rather than to limit device functionality. In contrast, our work dynamically builds partial control flow graphs that encapsulate specific device functionality. These partial control flow graphs can be enforced as security profiles at runtime or they can be used to audit device execution in order to ensure that an attached device is operating according to its expected run-time profile.

4.2.1 Hardware-Assisted Run-Time Monitoring for Secure Program Execution on Embedded Processors

This work[145] is a simulated design for a bus monitoring circuit that can enforce control-flow integrity on embedded devices in real-time. The design enforces inter-procedural control flow, intra-procedural control flow, and instruction stream integrity. Inter-procedural control flow is accomplished by storing all procedures in function call graph that is translated into a finite-state machine. Whenever the monitor observes a function call it checks to see if the call corresponds to a valid state transition. Intra-procedural control flow is accomplished through the use of a basic block table. For each basic block, its two possible successor addresses addresses are stored (the address of the next basic block depends on whether or not the branch was taken). Instruction stream integrity is accomplished by also storing the hash of each basic block in the basic block table, hashing the instruction stream for the basic block at run-time, and ensuring that the observed hash matches the hash stored in the table. The described data structures are stored in an *enhanced executable* and are loaded into a hardware monitor at run-time.

4.2.2 A Watchdog Processor to Detect Data and Control Flow Errors

This work[147] proposes a design for a hardware watchdog processor for Motorola M68040-based embedded devices. The watchdog is used for integrity and reliability purposes rather than for security. The watchdog is implemented as a custom circuit that is capable of detecting faults caused by radiations and electromagnetic interferences. These sorts of faults can cause two types of errors: data errors and control flow errors. The proposed watchdog processor can thus detect both types of errors. The watchdog protects against data errors by implementing a bus protection strategy based on Automatic Repeat Request. The watchdog protects against control flow errors by calculating a signature for each *branch free block* in the entire binary and storing the signatures for all blocks offline. At runtime the signature is recomputed and compared to the stored signature.

4.2.3 Vigilare: Toward Snoop-based Kernel Integrity Monitor

Vigilare[163] is a hardware kernel integrity monitor designed to facilitate integrity checking on operating system kernels (and hypervisors in particular). Existing hardware-based integrity monitors used sampling-based bus traffic monitoring schemes due to

implementation-level difficulties that this work purported to solve. In contrast to the snapshot-based approaches, Vigilare implements a real-time approach capable of monitoring all bus-traffic rather than only a limited amount. Vigilare loads the addresses of important kernel symbols within static regions from the System.map file and verifies the integrity of the data at these addresses during device runtime by capturing write operations to these addresses.

4.3 Background

Background knowledge of several key concepts will aid in understanding our solution to the problem of building partial control flow graphs and auditing execution traces.

4.3.1 Address Space Layout in Embedded Systems

The memory bus is used to transfer data between the CPU and a memory device. It is composed of three smaller busses: an address bus, a data bus, and a control bus. The address bus specifies the logical address of the memory to access. The data bus is used to communicate data to or from the memory device. The control bus is used to tell the memory controller what type of bus operation is to occur, as well as when an address or data is latched on the address or data bus, respectively.

Many embedded systems store their firmware on a ROM chip that is directly

mapped into the address space rather than on a secondary storage such as a hard drive or solid state drive.[166] In such a system the CPU can execute code out of the ROM by directly loading instructions from its memory addresses. Additionally, these systems typically contain a RAM that is also mapped to the address space. Typically RAM and ROM devices use an internal addressing scheme of linear addresses starting at address 0x0. Thus, linear addresses seen by the CPU must be translated into physical addresses used to access the data in the physical memory device. A memory controller translates linear addresses into physical addresses.

4.3.2 Instruction Prefeteching

Almost all modern CPU architectures implement some form of instruction prefetching[167] in order to keep the CPU pipeline filled with instructions. When the CPU pipeline is empty the CPU must wait for its next instruction to be fetched from main memory or from cache. This may be a relatively time consuming process. Instruction prefetching helps to minimize how often the CPU must wait on instructions to be fetched. One of the earliest architectures to implement instruction prefetching was the Intel 8086, which could linearly prefetch up to six bytes of instructions.[168] More modern architectures implement significantly more advanced prefetching algorithms to provide branch prediction[169] and branch target prediction.[170]

4.3.3 Interrupt Handling

Most CPU architectures support facilities for handling exceptions as well as environment-triggered inputs, which may be triggered in an unpredictable fashion.[171] Such constructs are signaled to the CPU externally through an interrupt controller or internally through a software interrupt. The CPU, upon detecting an exception or an interrupt, will execute a corresponding handler for the event. The list of all possible handlers is typically stored as a jump table somewhere in system memory. The precise location of this interrupt table is architecture–dependent. When interrupts are triggered the system will choose what to do based on a combination of architectural specification and system state. The system may ignore certain interrupts or delay their handling, or it may immediately halt execution and jump to the interrupt handler. In many cases the interrupt handler will return to the previously-executing code once it has finished executing. In some cases the interrupt handler may not return at all.

4.3.4 Control Flow Integrity

Control flow integrity is a defense technique that can protect a system against control flow hijacking attacks such a buffer overflows[172] and return-oriented programming.[173] Control flow integrity typically protects against such attacks with the use of a control flow graph generated by the compiler at compile time. The control flow graph is a graph of all basic blocks and transitions in a program. This graph is used

to generate a state machine. The program execution is then monitored and all control flow transitions are checked in accordance with the generated state machine. If any observed control flow transition does not correspond to a transition in the control flow graph, then the monitor will trigger a security violation and halt the attack.

4.4 Design

Sentinel has been realized as a compound design consisting of two orthogonal components: a bus tap and a device profiler. The Sentinel bus tap is connected to an embedded system's address and control buses. The bus tap forwards the start and end addresses of all basic blocks fetched by the CPU to the device profiler. It accomplishes this by monitoring all instruction fetches. If it detects a jump between basic blocks it forwards the jump source and the jump target to the device profiler. The jump target represents the first address in a basic block and the jump source represents the last.

The device profiler receives the list of basic block addresses starting with the address of the first basic block fetched by the CPU when the device initially boots. It uses these basic block addresses to construct a full control flow graph of the observed device execution. Thus, the device profiler receives device execution traces and constructs a control flow graph from them. This control flow graph is, by definition, sparse because any basic block start address is associated with a single basic block

end address, and any basic block has exactly two possible successors based on whether the jump at the end of the block is taken. Thus, the device profiler stores this control flow graph as an adjacency list.

This partial control flow graph represents a *security profile*. Any device modes or features that were accessed during the execution trace will be a part of the security profile, and therefore considered allowable accesses during subsequent audits or enforcement. Conversely, any features or modes not in the security profile will be considered anomalous accesses. The Sentinel prototype that we have constructed can audit arbitrary execution traces in order to determine whether or not the execution trace violates a given security profile. Combined with the control flow based real-time bus monitors proposed in the prior work[145–147] one could enforce Sentinel's security profiles on embedded devices in real-time.

Sentinel can either be integrated into existing designs internally or it can be attached to existing hardware externally. Because Sentinel is an open design it can be easily modified to fit a variety of use cases. In our embodiment, Sentinel is designed to be easily integrated into embedded systems that meet certain architectural requirements. While it may be possible to modify the Sentinel design to fit into other types of embedded systems, we will restrict our discussion to cover only the types of systems into which Sentinel can currently be integrated with no modifications.

4.4.1 Architectural Requirements

Sentinel can be integrated into any system that meets its architectural requirements. The Sentinel device profiler is architecture agnostic as it relies only on the abstraction provided by the bus tap in order to build the control flow graph. The Sentinel bus tap, on the other hand, can only be integrated into embedded devices that meet several basic architectural requirements. In practice these requirements are minor and most devices that are not system on chip (SoC) should meet them. Even SoC devices with additional external memory could possibly be made to work with the Sentinel bus tap in a more limited capacity.

4.4.1.1 External Memory Bus

The bus tap must translate the raw electrical signals that are sent across the memory bus into an infinite stream of addresses. Therefore, in order to be a candidate for Sentinel integration, an embedded system must fetch its instructions from external memory banks such as ROM and/or RAM. If an embedded system were to fetch its instructions from internal memory banks instead (as would be the case with a system on chip), Sentinel would have no way to monitor the address and control buses, so it could not be integrated into the target system. Note that this limitation does not extend to caches. Sentinel supports caches as it monitors the data path from the CPU to the system memory. Thus, data will be observed by the Sentinel bus tap as it fills the cache. Sentinel need not be aware that data is subsequently accessed from

the cache.

Although this external memory requirement appears to prevent Sentinel from being used with SoC devices with internal memory, this is not entirely true. In many types of embedded devices, the internal memory on the SoC is limited and is only used to store a bootloader or some limited subset of the code. The rest of the code will still be stored on external memory. In these cases the Sentinel bus tap is likely to work without issue. Since the device we have used for our demo does not have such an architecture, testing the veracity of this theory is left to future work.

4.4.1.2 CPU Address Bus Control Pins

The bus tap must also be able to differentiate between bus operations in order to determine when a bus operation corresponds to an instruction fetch. In many architectures[174–177] the CPU signals the type of bus operation requested to the memory bus controller using a control bus. The bus tap uses this control bus to differentiate an instruction fetch from any other type of memory bus operation. While Sentinel's bus tap implementation is configured to interface with the x86 architecture for the purposes of our demo, it could be trivially modified to interface with any other architecture that exposes memory bus control operations through external signals.

4.4.2 Integrating Sentinel into Embedded Devices

There are numerous ways in which one could realize the Sentinel architecture in an actual product design. In one embodiment, we envision the Sentinel bus tap and device profiler being packaged together on a custom ASIC. This ASIC would sit between the CPU and memory bus controller. The manufacturer would include a *profiler mode* switch inside the device. When set to *profile mode* the ASIC would automatically build a new security profile with the desired functionality and it would store the security profile on an external flash. When set to *enforce mode* the ASIC would enforce the stored profile on instruction fetches in real-time. This would protect the device against physical attacks (such as accessing "locked" modes) as well as against exploits (such as return-oriented programming attacks).

4.4.3 Methods of Integration

Sentinel can either be externally wired to existing devices or it can be integrated into device designs as a custom ASIC. To connect Sentinel to an existing device it must be wired to the address bus. Alternately Sentinel is designed to be easy to integrate into new revisions of the printed circuit board (PCB) layout of devices already in production or to be incorporated into the initial revision of new device designs. This flexibility allows a design to be retrofitted with Sentinel at minimal cost. Furthermore if a device design already uses a custom ASIC as a memory bus

controller, as was the case in several devices at which we looked, then Sentinel can be integrated directly into this ASIC. Otherwise, Sentinel can be placed between a CPU and a memory bus controller in order to monitor and audit bus operations.

4.4.4 Failure Modes

If a real-time enforcer were configured to use the Sentinel-generated security profiles, the enforcer could be easily designed to take a number of corrective actions in response to a detected security violation based upon the needs of the specific application. In an audit-based implementation, Sentinel would be configured to report all security violations, while allowing the device to continue executing. This is usually the safest option to take if we do not know in advance what effects halting execution might have on a device. However, in some cases halting execution might be superior to allowing execution to continue. The Sentinel device profiler can be trivially modified to cause the CPU to immediately stop executing by connecting a security status output to the reset pin of the attached CPU. In more advanced designs, a corrective action could additionally be implemented. For example, an administrator might want to reset the device, disable network interfaces, and restore it to known-good settings so that it can continue executing in an offline-mode. For many embedded devices, Sentinel enforcers could easily be constructed to fit any of these use cases.

4.5 Implementation

In our prototype embodiment we have externally connected our Sentinel to a popular Intel 80C188-based[177] embedded device. We prototyped our bus tap on an FPGA and our security enforcer on a Linux workstation. The Sentinel bus tap is designed to be directly wired to the address pins and bus control pins of the 80C188. The bus tap observes all instruction fetches and captures the start and end addresses for each basic block that is executed. The bus tap then forwards these addresses to the security enforcer running on the Linux workstation over USB 2.0. On the workstation our device profiler prototype builds a control flow graph taken from concatenated execution traces and it enforces this profile on a captured execution trace. The device profiler algorithm runs in sub-real-time due to bandwidth constraints of our FPGA's development libraries. In a full-hardware ASIC realization this sub-real-time limitation would not exist.

4.5.1 Intel 80C188 Architecture

The Intel 80C188 is an 80186-based 16-bit x86 CPU with an 8-bit wide data bus. The 80C188XL contains 20 address pins, an address latch and three bus cycle status information pins. The bus cycle status information pins shown in Table 4.1[177] are used to announce the type of bus operation currently in progress. The real-time bus monitor is wired directly to these bus cycle status information pins. Using this

information combined with the Address Latch Enable (ALE#) pin, the bus monitor is able to decode address bus operations.

4.5.2 Bus Tap

We prototyped the Sentinel bus tap on an Opal Kelly XEM3010-1500 FPGA. This FPGA contains a USB 2.0 interface and a high-speed I/O bus which we connect to the target device through an attached breakout header. The bus tap is wired to the address and control pins and forwards all bus traffic of a chosen type over its USB 2.0 interface to a computer for analysis. The bus tap is wired to the address, bus control and address latch enable pins on the 80C188. The FPGA samples the values of these pins continuously at 133Mhz. When the address latch is active (low) and the bus control pins denote that an instruction fetch is occurring on the data bus the bus tap saves the corresponding address from which the current instruction is being fetched. The bus tap requires that an address is valid for three sample-cycles in order to minimize the number of potential errors caused by electrical noise, crosstalk and loose wires.

If the address is stable for three cycles the bus tap checks to see if the address is contiguous to the last valid address that it captured (e.g. if $currentaddress = previousaddress + 1$). If the addresses are contiguous it increments the previous address and continues to the next address. If the addresses are not contiguous then the bus tap will have captured the end of a basic block (e.g. the previous address)

and the beginning of a new basic block (e.g.) the next address. This implies a control flow change in the program from the previous basic block to the new basic block. The bus tap stores both of these addresses in a FIFO queue to await transfer to the security enforcer.

The fact that we are not storing contiguous blocks in the FIFO is an optimization to significantly increase throughput. Rather than transmitting contiguous addresses over the USB 2.0 interface we can instead transmit block boundaries. Since the x86 architecture always begins executing at address `0xFFFF0` the receiving computer can always determine whether it is receiving the start address or the end address for a basic block. This is important for reasons that will be described in our discussion of the implementation of the device profiler.

4.5.3 Device Profiler

The Sentinel device profiler receives a list basic block boundaries corresponding to a device execution and uses this execution trace to build a control flow graph of the associated functionality of the attached embedded system. The device profiler enforces both inter-procedural and intra-procedural control-flow constraints. Our implementation is complicated by two issues: instruction prefetching and interrupt handling. We have solved both of these issues in our x86-specific reference implementation and our solutions are generic; they can be applied to most architectures. We have also considered architectures that utilize branch-prediction and out-of-order

execution even though our 80C188 CPU did not have these features.

The Sentinel device profiler is fed a concatenated set of execution traces taken from the embedded device as it is run through its normal operations. The profiler tracks the start and end addresses for all basic blocks in each execution trace. For each address, it records the immediately preceding address as a valid possible predecessor address. That is, $(previous\_block\_end, current\_block\_start)$ and $(current\_block\_start, current\_block\_end)$ are recorded as valid jumps in the table, capturing both inter-procedural and intra-procedural control-flow. Thus the jump table forms an adjacency list representing the full control-flow graph of addresses of the instructions that fall within the security profile. Basic block boundaries are shown in more detail in Figure 4.1

In addition to the aggregated execution traces corresponding to the security profile, our device profiler is also fed a target execution trace on which to enforce the security profile. The device profiler checks to see if every control flow change in the target execution trace is also in the jump table corresponding to the security profile. In the event that the control-flow change is not in the security profile the Sentinel security enforcer outputs a security violation. The Sentinel security profiler will detect any unexpected control flow changes including control-flow changes caused by device errors, by a user accessing modes that are not in the security profile or by control-flow hijacking malware attacks (such as buffer overflows or many types of return-oriented programming attacks).

4.5.3.1 Instruction Prefetching

The Intel 80C188 CPU is capable of prefetching up to four bytes to help keep the CPU's pipeline full. This prefetching would cause problems with our security algorithms described above. In particular, the end address that we observe for a given basic block may not be the actual end of the block–it may actually be up to four bytes past the block boundary. Thus prefetching introduces a small amount of non-determinism into the system. In order to account for this nondeterminism in the system, we must allow for nondeterminism in our enforcement. Thus, to account for the prefetching we allowed for a margin of error of +/- 4 bytes in a block boundary.

In a system meant to enforce strict control-flow requirements such as a CFI-based exploit mitigation method, this margin of error could possibly have a slight effect on the overall security of the system [2]. However, since the purpose of Sentinel is instead to profile and enforce mode constraints on an embedded device, this nondeterminism does not significantly weaken the underlying security of our system [3].

4.5.3.2 Interrupt Handling

Upon considering the security profiling algorithm above it should be clear that an obvious problem is how to account for control flow jumps generated by interrupts

[2]Though this has not been established and indeed would have have to be evaluated in order to make a determination one way or the other

[3]In order to understand why this is true consider the fact that to use prefetching-based nondeterminism to properly profile an invalid device mode as it profiles a valid device mode, the improper mode would need to be split into chunks that each fall within a maximum of n bytes of the end of a basic block where n is the maximum number of bytes that the CPU architecture can prefetch.

and exceptions.

When an interrupt is triggered on the target system the bus tap will detect a jump to the interrupt. This means that the bus tap will insert a basic block boundary immediately before the interrupt start address in the instruction stream. This boundary does not correspond to an actual block boundary in the software's basic block graph but instead is caused by the interrupt splitting the block. Similarly, an interrupt return is followed by another block boundary where the profiler begins executing again. This is illustrated in Figure 4.2. From an enforcement perspective, this means that jumping to or returning from an interrupt or exception should generate two consecutive violations. Thus, we can relax the restrictions on our enforcement algorithm to allow for this particular case without significantly compromising our goals. Since we are trying to disable manufacturer-implemented features and device modes rather than enforce control-flow integrity (which is an already-solved problem) we can assume that in almost all cases any useful device mode or feature that we might be interested in restricting access to would contain more than two basic blocks. Thus we can handle arbitrary interrupts without any knowledge of the underlying system and without compromising security under our threat model.

In some cases we may have access to the interrupt table (e.g. through a firmware update image). In this case we can restrict the above algorithm to only allow for jumps to known interrupts. This will make false negatives in the system even less likely than in the general case. In an x86 binary image this table is located at

addresses `0x00000`—`0x00400`. From this table we can obtain the base address of all interrupt and exception handlers. We can then search the binary for the opcode of all iret instructions. Once we have lists of all interrupts and interrupt returns we can combine these lists to form our list of interrupt boundaries [4]. Every time we detect a control flow change we first check if it's an interrupt boundary (an interrupt start or end address). If so we skip enforcement on the previous, current and next addresses. We then continue executing normally so that we may profile the code within the interrupts. In summary, if we know the interrupt table we can restrict our algorithm to only allow for unprofiled jumps to or from known interrupt-boundary addresses.

4.5.3.3 Branch Prediction and Branch Target Prediction

Some advanced architectures utilize branch prediction and branch target prediction in order to improve performance. Although Sentinel has not been tested on such architectures, we believe that it should support them with extremely minor modifications. Because Sentinel, in its current implementation, allows two consecutive invalid jumps before it flags a mode as suspicious, a missed branch prediction should not trigger an alert. However, we have not tested this and it is conceivable an alert may be triggered if an interrupt occurs after the CPU has fetched instructions for a predicted branch that isn't in the profile that Sentinel is enforcing. This could be remedied by allowing for one additional level of indirection before triggering an alert

[4] This combination is a convenience for ease of implementation. There is no technical reason why interrupt start and end addresses could not be handled separately for slightly greater accuracy.

Bus Cycle Initiated	S2#	S1#	S0#
Interrupt Acknowledge	0	0	0
Read I/O	0	0	1
Write I/O	0	1	0
Halt	0	1	1
Instruction Fetch	1	0	0
Read Data from Memory	1	0	1
Write Data to Memory	1	1	0
Passive (no bus cycle)	1	1	1

Table 4.1: Bus Cycle Status Information (Reproduced from Intel 80C188XL Datasheet).

(e.g. by allowing three consecutive invalid jumps before triggering an alert). Testing this has been left for future work.

4.5.3.4 Out-of-Order Execution

Many modern CPU architectures contain advanced performance optimizations such as out-of-order execution. Although the 80C188 that we tested our Sentinel prototype on did not contain these features, we believe that Sentinel already supports most out-of-order execution implementations as-is. This is a corollary to the already-existing cache support included in Sentinel. In out-of-order execution, the CPU fetches instructions in program order and stores them in a cache.[178] The CPU then pulls instructions from the cache in an optimized order. Thus the out-of-order step occurs after the instruction fetch from memory. Since Sentinel concerns itself only with instruction fetches that occur outside of the CPU, out-of-order execution should have no impact on Sentinel at all.

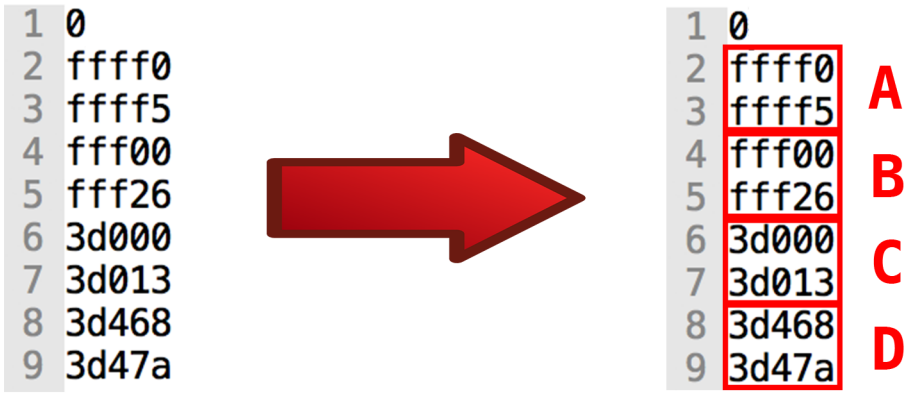


Figure 4.1: Example of how a raw bus capture (left) is split into basic blocks (right).

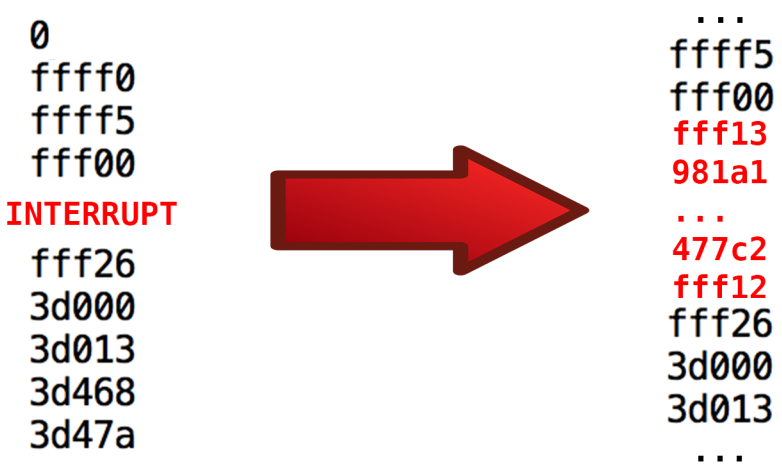


Figure 4.2: Example of how interrupts split a basic block.

4.6 Evaluation

We evaluate the Sentinel architecture by proving its real-world efficacy in detecting a physical attack on an embedded medical device. The process for this evaluation is as follows:

1. Create security profile
2. Enforce security profile
3. Access mode not defined in profile
4. Detect attack in execution trace

4.6.1 Alaris SE Infusion Pump

We evaluate the Sentinel architecture by connecting our working prototype to a popular embedded medical device. For our test implementation, we have selected the Alaris Signature Edition (SE) Infusion Pump due to its popularity and its x86-based architecture. We looked at two different board layouts for the Alaris SE. The Alaris SE contains an Intel 80C188-based[177] processor (or, in some revisions, the AMD equivalent), a custom ASIC for address encoding/decoding and data bus multiplexing, and either an Intel E28F800-CVT70[179] flash memory or an STM M27C801 UV EPROM. In board revisions that contain the Intel E28F800 the manufacturer added a header to emulate the pin layout of the STM M27C801 as shown in Figure 4.3. When de-

scribing information that is pertinent to both chips we will subsequently use the term ROM. The Intel EF2800-based Alaris pump contains an undocumented programming mode (discovered through reverse engineering) that can be entered by applying $+5V$ to the VCC pin on the programming header and connecting the GND pin to the ground source. The STM M27C801-based pumps contain a socketed EPROM that can be read using a standard universal flash programmer. We dumped the firmware from the STM M27C801 by using a flash programmer to receive a binary image of firmware version 2.79.

The Alaris SE logic board also contains a 128K battery-backed SRAM to which some code is copied at boot and in which persistent settings are stored. The Intel 80C188 is capable of addressing up to 1MB of memory. The onboard ROM is 1MB and thus occupies the entire address space. The SRAM overlays the memory region beginning with address `0x3D000`. Upon system boot the device first copies 6144B of data from `0xFD000` to `0x3D000` before jumping to `0x3D000` where it begins executing.

4.6.2 Connecting the Bus Tap

We connected our bus tap to the infusion pump by soldering wires to each address pin, the CPU bus control pins (S_0, S_1 and S_2) and the ALE# pin. Our particular infusion pump selectively enabled the ROM or the SRAM through the use of chip enable pins wired from the ASIC to the respective chips. The address pins connected to both the ROM and the SRAM through the ASIC and used in conjunction with the

chip enable pin could be used to send and address to either the ROM or the SRAM. The address pins wired to both of these chips always showed the same addresses as the address pins on the CPU. Thus to simplify the soldering we actually soldered our wires to the address pins connected to the ROM rather than to the CPU as shown in Figure 4.4.

4.6.3 Error Correction

Due to the requirement that the bus tap interface only send valid addresses, the bus tap will occasionally drop an address if an error occurs. When this happens in the middle of a block, the bus tap will see two noncontiguous addresses and think that a jump has occurred. If it happens at the beginning of a block the bus tap will start the block one address after the actual starting address of the block. If it happens at the end of a block the bus tap will end the block one address before the actual ending address of the block (which may or may not be the real ending address because prefetching makes this impossible to discern). We have trivially handled each of these errors in software.

In our test implementation, we did not implement reliable delivery over our bus tap interface due to overhead in our FPGA libraries. Even without reliable delivery we were able to reliably build and enforce profiles. The lack of reliable delivery did affect profile building but it had no impact on profile enforcement. Indeed, in our tests, different device modes looked so different from the perspective of our enforcer

# of Samples in Profile	1	2	3	4	5	6	7	8
# Errors in Control Sample	2	2	0	0	0	0	0	0
# Errors in Experimental Sample	9250	9220	9049	8705	8374	8367	8365	8358

Table 4.2: Relationship between number of samples in profile, false positives, and detection accuracy.

that it was easy to distinguish between them. Similarly, despite the lossy interface we were able to build enforceable profiles with low false positive rates with just one sample, and we were able to build enforceable profiles with no false positives at all by the time we reached five samples. Thus, by profiling a device during typical use over a short period of time we can build reliable and enforceable profiles even with respect to lossy interfaces. This means that in practice a bus monitor can be significantly weaker than the system it is attached to, thus lowering the overall cost.

4.6.4 Enforcing a Profile

To demonstrate the efficacy of our Sentinel platform we use it to build a profile of a normal device booting and navigating to the "Infuse" screen where a healthcare provider can configure infusion rate and amount and start an infusion. We intentionally build the profile to not include the "Options" screen. Thus, if a user accesses the options screen this should be flagged as a security violation. We show that we are able to build an enforceable profile with no false positives after just three sample.

4.6.4.1 Testing the Profile

In order to test the efficacy of enforcing a security profile we took eight execution traces each containing 8,388,608 addresses. The execution traces consisted of the addresses of all basic block boundaries observed by our bus monitor while we booted the pump to the "Infuse" screen. We also took a ninth execution trace of the device booting to the "Infuse" screen to use as an experimental control. Our test case consisted of the first 8,388,608 basic block boundary addresses observed by our bus monitor while we booted the pump to the "Infuse" screen and then navigated to the "Options" menu. Note that in all cases parts of the data corresponding to some screens were null due to data loss through our system.

We created eight profiles to enforce by concatenating up to eight execution traces together. For example, profile one contained execution trace one, test case two contained execution traces one and two, test case three contained execution traces 1-3 and so on. We enforced each profile on our control sample and on our experimental sample. In our control sample we would expect to see no security exceptions since we are enforcing our "Infuse" profile on an "Infuse" test case. In the experimental sample we would expect to see numerous security exceptions since we are enforcing the "Infuse" profile while the user accesses an unprofiled device mode. The results of our experiment confirm our hypotheses and are reflected in Table 4.2.

4.6.4.2 Discussion

From our results we see that our method of capturing partial execution traces can be used to reliably differentiate between device modes and to enforce profiles. We see that in the control test we had a few initial false positives but that the number of false positives quickly decreased to zero as we added more samples to our profile.

We also observed that we were immediately able to differentiate between our control and experimental test cases even with just one experimental sample in our profile. As we concatenated more samples to build our profile the number of false positives quickly dropped. After combining four samples into a profile all false positives not caused by physical errors had been eliminated. Thus, our results imply that only a few samples are needed in order to successfully profile a given device feature. Furthermore, accessing a "locked" device mode generated several orders of magnitude more security errors than accessing a profiled device mode and thus even if we had not combined multiple samples to build a device profile we still could have distinguished between the control and experimental cases with high accuracy.

4.6.4.3 Automation Through Keypad Emulation

Because we have shown that building accurate profiles is best accomplished by combining multiple samples, we devised a method for automatically generating samples by emulating keypad inputs on our Alaris SE infusion pumps.

In order to differentiate key presses, the host device typically uses a microcontroller

to poll the button rows in sequence. When a key is pressed, the microcontroller reads from the column output and derives the button location in the matrix. When no keys are pressed, there is no output on the column lines. When a key is pressed, one of the column lines will go high and the microcontroller can determine the individual button based on the row it was polling.

The Alaris SE uses 24 buttons. The row and column lines are fed to the host device through two 8-pin FFC (Flexible Flat Cable) cables. In order to access these lines, we soldered hookup wires to the the mount points on the FFC connectors on the underside of the PCB. Using a logic analyzer, we reverse-engineered the electrical signals corresponding to various keypresses. Since buttons in the same column trigger output on the same line, the first step was to map out the complete button layout to determine which pins corresponded to column lines. The row lines were trivial to ascertain with the logic analyzer since the microcontroller polls were clearly evident. We found the column lines by pressing each button and observing which line responded. Using this process we discovered that there were five row lines and seven column lines. The remaining lines were unused.

Once each button's column was known its corresponding row was discovered by comparing the output signal to one of the five possible inputs. Since holding down a button will result in the exact signal output as one of the inputs, this process was simply done through direct comparison and was exhaustively applied to determine the matrix location of each button. Once complete, it was possible to wire a given row

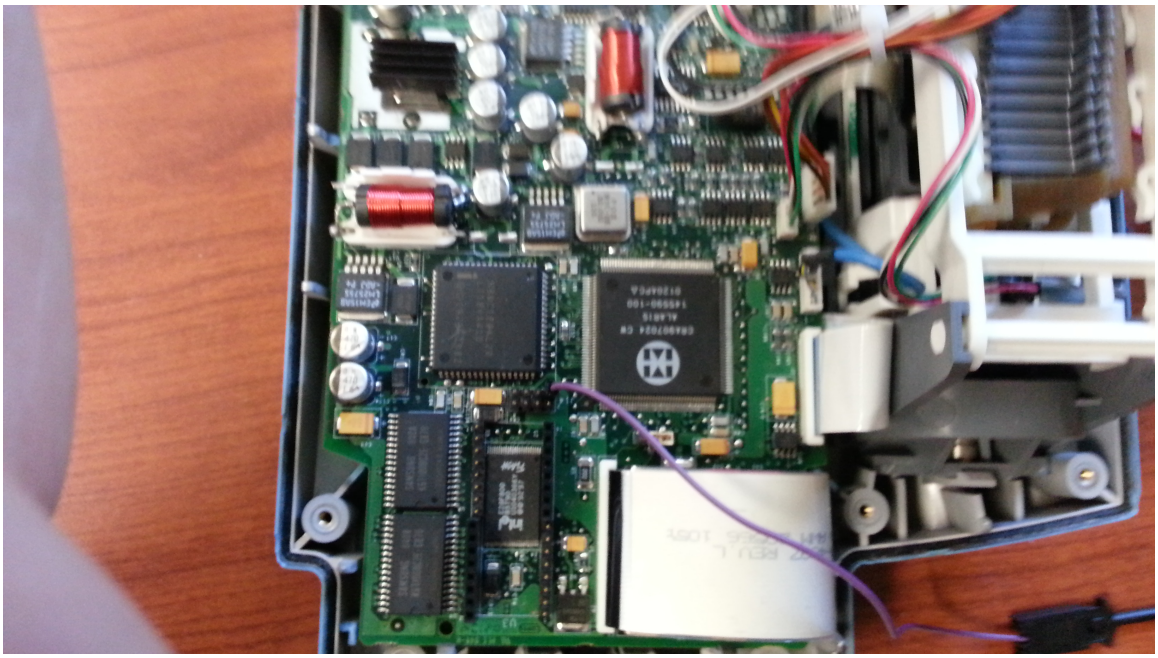


Figure 4.3: Board Layout of the Alaris SE 7132. Redundant SRAM chips on left, CPU top-middle, flash between header pins, ASIC top-right.

to a column and force the microcontroller to interpret a button press. This enabled external command of the keypad interface without physically pressing buttons.

Using GPIO on a Raspberry Pi we replaced desirable buttons by wiring their corresponding row and column lines through a relay. We used a script to activate buttons in sequence, allowing for total control over the device interface.

4.7 Additional Capabilities

In addition to the device profiling capabilities already outlined, our Sentinel prototype also includes several other important and useful features that are a corollary to the fact that we have completely separated the implementation of the bus tap and

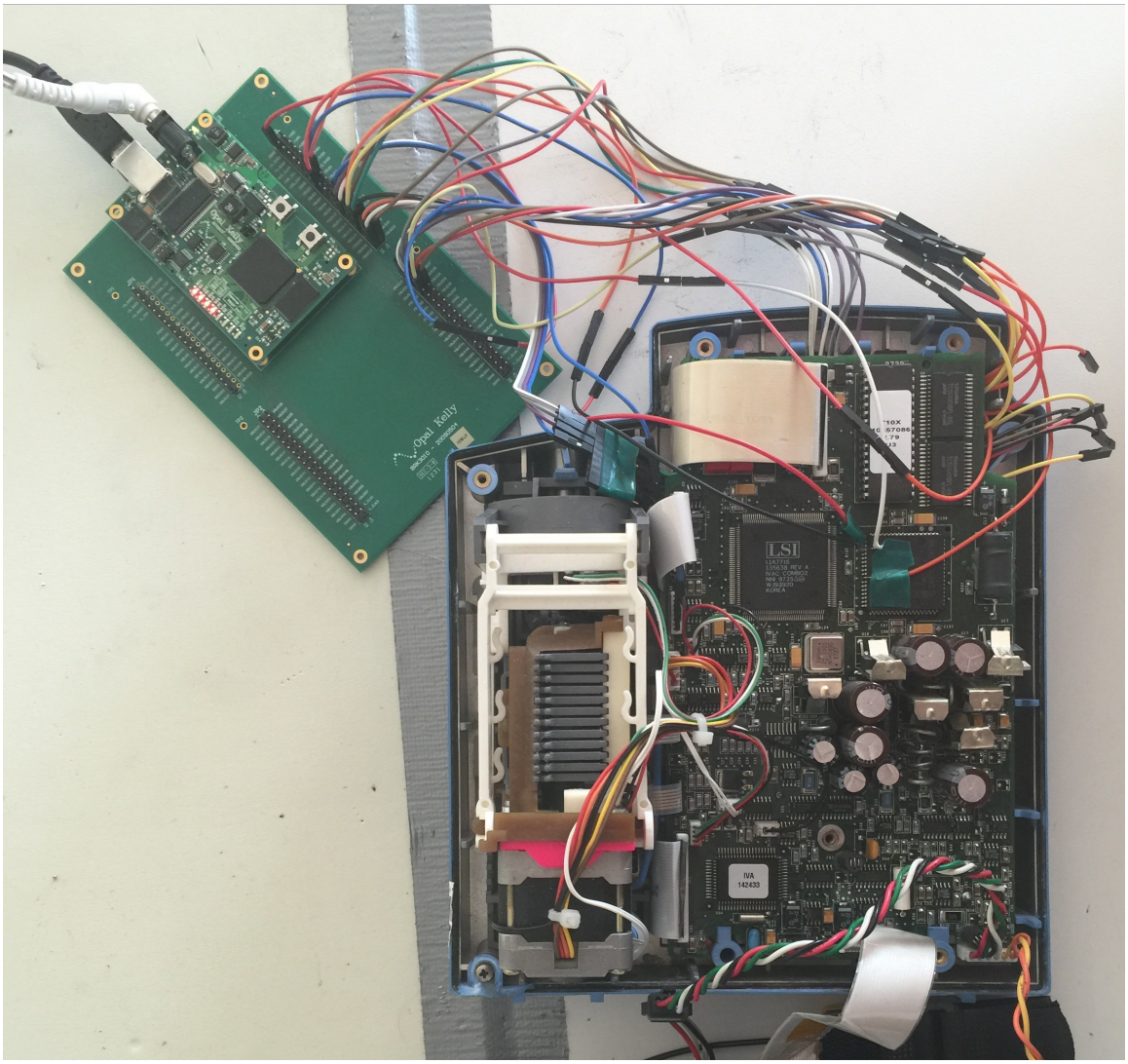


Figure 4.4: Image of Sentinel Security Profiler attached to an Alaris SE 7100 Infusion Pump.

device profiler. Since our bus tap forwards all basic block data to a workstation for analysis, we have written a number of tools to use this data for a variety of other security related tasks.

4.7.1 Address Writer

The most basic tool that we wrote is the Sentinel address writer. The address writer simply writes an arbitrary number of captured addresses to a file. The address writer was used extensively in debugging the bus monitor and in understanding the operation of our infusion pump. In one case we were able to use the address writer combined with manual analysis of the firmware image to discover an interrupt not listed in the interrupt table. Without this insight we would not have been able to get the enforcement algorithm to work with our infusion pump.

4.7.2 Assisted Disassembly

To assist with disassembly of the specific functionality of interest, we modified our bus tap to begin capturing when we pressed a switch. We were then able to begin capturing addresses immediately before activating the device functionality corresponding to the disassembly that we were interested in viewing. We wrote a corresponding utility that turns address streams into IDAPython[180] scripts that automatically disassemble all basic blocks observed during the capture. One challenge of assisted

disassembly is syncing to the start of basic blocks. This is a challenge because the device is already running when the switch is pressed. This means that the first address could either be the beginning address or the end address of a basic block.

To solve this seemingly troublesome issue we alternated between every address, splitting them into two groups. We then did a frequency count of the number of interrupt start addresses per group. We ignored the interrupts that were placed immediately after iret instructions because we cannot distinguish between whether this is an interrupt return or an interrupt start due to prefetching. The group that had more interrupt start addresses was deemed to be the group of basic block entry addresses.

4.7.3 Assisted Fuzzing

Our security enforcer prints all addresses not in the security profile. This information is useful for assisting in the fuzzing process to find device vulnerabilities. We can use this information as part of our fuzzing process as it allows us to automatically fuzz input vectors on our devices and then manually determine if a particular crash is exploitable. Thus, by running a fuzzer in conjunction with enforcing a liberal security policy we can semi-automate the process of vulnerability discovery on a given device.

4.8 Future Work

This work opens up several additional avenues for future research questions to explore and extend upon.

4.8.1 FPGA Implementation

Sentinel's current prototype implementation works with execution traces rather than infinite address streams. If we were to implement the Sentinel security profiler on an FPGA it could be used as a real-time bus monitor. It would be able to detect and handle violations as they occur and it could be shown to prevent malware attacks in addition to physical attacks. This would be a worthwhile extension to the platform that would further increase its usefulness. Although malware detection using address bus monitoring has been done previously, using it in conjunction with partial control flow graphs would be an interesting exercise that could allow us to study the relationship between graph coverage of a security profile and that profile's malware protection capabilities.

4.8.2 Testing Out-of-Order Execution and Branch Prediction

Because the 80C188XL found in our test device does not support features such as branch prediction or out-of-order execution we were not able to test our out-of-order execution algorithm in practice. We believe that this would be a useful exercise as it would improve confidence in our algorithm and definitively open up Sentinel to a new class of devices. Testing the algorithm would require a new device, and a bus tap implemented on an FPGA capable of transferring data at higher speeds than that of our bandwidth-limited USB 2.0 prototype [5].

4.8.3 Observational Study

In this work, we have shown the feasibility of using bus captures to build and enforce execution profiles on embedded devices. It would be interesting to study these execution profiles in practice. Future work could look at the impact of connecting Sentinels to embedded devices deployed in a variety of settings. We could then study how often devices deviated from their execution profiles in typical use cases and could thus derive an understanding of applications for which Sentinel would be well-suited and for which applications Sentinel would be poorly suited given that Sentinel's

[5] In our prototype, bandwidth was further limited by the significant overhead that the USB library produced by the FPGA manufacturer introduced into the system. This was acceptable for our low-speed application but would cause problems in a higher-speed system.

enforcement model depends on having a predictable and inclusive execution profile.

4.9 Conclusion

Sentinel is a useful and practical tool for utilizing bus captured execution traces to build partial control flow graphs of embedded devices. These control flow graphs can be used to audit execution traces of embedded devices in order to detect physical attacks. The Sentinel platform can be attached to existing devices with no modifications to the underlying design. We have shown our bus monitoring technique to be effective in actually building partial control flow graphs and we have used it to successfully detect physical attacks on a popular legacy model of infusion pump. In the future, this work could lead to ASIC-implemented hardware co-processors that can be used to secure legacy embedded devices by enforcing limited-use device profiles on otherwise feature-rich devices in real-time.

Chapter 5

Beacon+: Applications of Short-Range Authenticated Unidirectional Advertisements

5.1 Introduction

Tracking and managing assets in real-time is critical for large organizations. For example, in hospitals, "more than one third of nurses spend at least 1 hour per shift searching for equipment and the average hospital owns 35,000 inventory SKUs and utilization hovers around 32-48%, with nearly $4,000 of equipment per bed, lost or stolen each year".[181] Moreover, tracking needs to be secure; specifically, it needs to be resilient to active and passive attacks that aid in the misappropriation of assets. We

implement a real-time tracking system using low-cost Bluetooth Low Energy (BLE) devices that provide authenticated wireless communication to securely track assets and people.

Assets in our system are tracked with a device that can receive BLE transmissions[1] and transmit on common media such as Wi-Fi. To provide location information, we implement Beacon+: a BLE device that extends the design of Apple's popular iBeacon specification[182] by modifying the advertisement, or unidirectional broadcast, to contain a monotonically increasing sequence number and CBC-MAC tag. The sequence number provides temporal freshness that is resilient to clock skew without requiring synchronization, and the CBC-MAC tag authenticates the Beacon+ to a trusted server over a potentially untrusted link (e.g., smartphone). Specifically, the trusted server validates the authenticity of Beacon+ advertisements and updates location data about the tracked assets using the absolute location of each Beacon+.

We use the real-time tracking system coupled with Beacon+ as a foundation for other applications such as access control that enforces location-based restrictions, which we implement. This application relies on the authenticity of received Beacon+ advertisements to compute the relative location to an asset and provides access to asset data if and only if the accessor (i.e., person whom requires the data) is within close physical proximity. Location here is only one factor in a multi-factor access control scheme to authenticate a user. For example, in the context of a hospital,

[1]Some assets support BLE and therefore do not require an additional device.

nurses and doctors who are away from their personal computer but moving around with a hospital-issued tablet must login to the tablet with their credentials and be within close physical proximity of a patient to access her medical record.

Access control that enforces location-based restrictions is novel in that it addresses the threat of a single authorized device being compromised or stolen to access a large portion of private asset data. That is, an attacker must be physically present to obtain a fraction of the private data rather than simply gaining access to everything on the device.

We also use the real-time tracking system to describe an indoor navigation application using Beacon+. Specifically, a user's smartphone forwards authenticated advertisements to a backend server that calculates the user's position and pushes location data to be displayed on a map. Active spoofing or injection can be used to misguide navigation; however, Beacon+'s authenticated advertisements mitigate these attacks.

The linchpin of our applications is Beacon+, and to build a more secure and interoperable Beacon+ we require the following capabilities: (1) perform symmetric key operations; (2) modify advertisement fields; (3) transmit unidirectional advertisements; and (4) retain traditional beacon (e.g., iBeacon) advertisement structure. We are aware of only one similar, authenticated beacon called Trusted Beacon.[183] Beacon+ differs from Trusted Beacon in its choice of cryptographic primitive and number of advertisements for a single transmission. Specifically, Trusted Beacon

lacks capabilities (1), (2) and (4).

Trusted Beacon uses a 320-bit asymmetric RSA private key to sign a random value that is transmitted via multiple advertisements and is valid for 5 minutes. The requirement of multiple advertisements is the result of having a signature that is longer than the message to be signed, and thus does not conform to the iBeacon specification. In contrast, Beacon+ uses a 128-bit symmetric AES key to compute a CBC-MAC tag on a monotonically increasing sequence number that fits in a single advertisement and is only valid for 1 second. Beacon+ conforms to the iBeacon standard and is compatible with BLE applications that interact with beacons. Furthermore, Trusted Beacon's chosen private key size is small and thus easily breakable.[184,185]

For ease of implementation and testing, we prototyped Beacon+ using a TI MSP430 Launchpad and BLE BoosterPack (see Section 5.4). The Launchpad provides non-volatile storage and additional computational resources that we use to maintain state and perform symmetric key operations. The BLE BoosterPack exposes an interface to broadcast BLE advertisements with our modified fields. This prototype satisfies the required capabilities listed above.

Our contribution. In this work we implement Beacon+, a BLE device that transmits short-range unspoofable, authenticated advertisements. We implement a real-time tracking system that uses Beacon+ to more securely track assets and provides a foundation for other applications. We describe a real-time indoor navigation ap-

plication, and design and implement a novel application that provides location-based restrictions on access control. Lastly, we validate the feasibility of our Beacon+ prototype with respect to accuracy and cost.

5.2 Background

Privacy and access control has been an area of concern for years. While there has been prior work in protocol design for location-based access control ,[186–188] there has, to the best of our knowledge, been little work done regarding the implementation and evaluation of these suggested protocols. Although existing solutions have had varying levels of success using RFID,[189,190] GPS,[191] and WiFi[181] to track and manage assets based on location, Bluetooth beacons are able to accomplish this task in a more straightforward, efficient and cost-effective fashion. In this section, we will explore the functionality of these other technologies, and discuss how their limitations necessitated the invention of Beacon+.

5.2.1 Radio Frequency Identification

Radio Frequency Identification (RFID)[192] was originally designed for short-range asset tracking. RFID consists of *tags* and *readers*. RFID readers interrogate tags and receive unique identifiers along with other data. RFID tags can be either passive or active. Passive RFID tags are powered entirely by the signal transmitted from an

RFID reader and are thus computationally limited in their abilities. An active RFID tag is powered externally which allows for more computational power than a passive tag, yet also makes it more expensive. RFID readers are typically placed at ingress and egress points of a particular set of areas.[193] The readers then read all RFID tags entering or leaving the monitored area. Thus, RFID is typically used to track assets in warehouses or to monitor assets as they flow through a supply chain.

The typical communication range for RFID is limited to tens of centimeters.[194] Different bands of RFID communication (low frequency, high frequency, ultra high frequency) can increase the range up to 12 meters.[194] However, higher frequency RFID requires expensive antennas to extend the range. Deploying these expensive antennas throughout an extensive area such as a hospital is impractical.

5.2.2 Global Positioning System

Global Positioning System (GPS)[195] was devised as a reliable global satellite system for providing time and location information to any receiver with a clear view of at least four satellites. GPS satellites broadcast a pre-defined pseudorandom number stream that is used by the receiver to calculate a time of arrival (TOA). Satellites also broadcast a time of transmission (TOT) and a position. The receiver uses the TOA, TOT and position information from four satellites to compute its own x, y and z coordinates and clock drift compared to satellite time.[196]

GPS is well-suited to outdoor tracking applications, but it does not function well

when there is no direct line of sight to at least four satellites. Thus, GPS is generally not suitable for establishing indoor positioning[196,197] because it is often not accurate enough within buildings.

5.2.3 Wi-Fi

Wi-Fi[198] was devised as a means to facilitate wireless networking over mid-ranged distances. In business and university campuses, multiple wireless access points are often used to provide coverage to the entire area; specifically, access points are connected to the same underlying network and positioned such that each access point covers a certain area, seamlessly handing off clients to its neighbors as clients move throughout the area of coverage.[199] These access points each have unique identifiers that are tied to specific locations. Therefore, an administrator could, in theory, track the location of individual clients on the wireless network by observing the order and location in which the clients connect with access points over a given time period. In fact, several companies[200] have begun manufacturing inexpensive Wi-Fi tags that can be attached to objects to track where the objects are in relation to base stations.

Wi-Fi communication meets the accuracy, timeliness, and communication range requirements for indoor position management and tracking. Indeed, previous work has looked at using Wi-Fi tags for exactly this puporse.[201] One of the benefits of Wi-Fi-based solutions is easy adoption; Wi-Fi tags are simply attached to devices or staff, and communicate with existing access points. However, adhesive Wi-Fi tags are not

securely integrated with the devices they manage (and in fact may not fit on smaller devices). Tags can be mixed up or maliciously removed, compromising tracking accuracy. In addition, Wi-Fi is not as power efficient as other wireless technologies, requires a layer of management (e.g., password, SSIDS, etc.), and requires bidirectional communication, which introduces a more sophisticated adversarial model than unidirectional beacons. For example, an adversary can continuously communicate with the Wi-Fi device, attempting to authenticate and gain access.

5.2.4 Near Field Communication

Near field communication (NFC)[202] was invented for extremely short-range communication, on the order of several inches. Therefore, for the applications considered in this work, NFC is infeasible, as it would require an unreasonable number of NFC devices.

5.2.5 Bluetooth

Bluetooth[203] is a short-range communication protocol found in mobile devices and computers. Bluetooth was originally designed to allow computing devices to communicate with peripherals wirelessly.[204] Bluetooth-enabled devices initiate connections to host devices by entering *discoverable mode* and waiting for a scanning device to make a connection inquiry. The device then responds to the connection inquiry by

sending information including a device name and a device class. If the host chooses to connect to the client device then the two devices go through a *pairing process* to establish a *bond*.[205]

Bluetooth technology has been used in the past to build tracking systems.[206–210] Previous work using Bluetooth technology has generally used older Bluetooth versions (older than v4.0) and did not consider security as a design goal. Some tracking systems had the tracked entities establish connections with Bluetooth infrastructure devices (similar in functionality to our beacons), resulting in two-way communication with potentially untrusted entities, which could lead to compromises.[209] Other tracking systems[210] require that the infrastructure devices have a dedicated power supply and a connection to a LAN or Wi-Fi network, conflicting with our lightweight design goals (e.g., low cost and battery usage), and adding other potential security vulnerabilities.

Beacons. Bluetooth has gone through many revisions and extensions since its initial inception. One such extension is Bluetooth low energy (BLE), which was originally introduced by Nokia in 2004[211] and integrated into the Bluetooth 4.0 standard in 2010.[212,213] BLE uses significantly less power than classic Bluetooth and BLE devices can advertise information to a host device without requiring the host device to pair. Conceptually, BLE is similar to NFC, but it is capable of operating at much longer ranges than NFC. BLE is designed to be integrated into devices, such as sensors, that need to broadcast small snippets of data at irregular intervals. Due to these

advantages, BLE has been widely adopted in many Internet of Things (IoT) enabled devices.

One particularly useful implementation of BLE technology is beacons, which are inexpensive BLE devices (in the range of $5[214] to $30[215]) that broadcast a fixed unique identifier, similar to NFC tags. Beacon identifiers are interpreted by applications and are used for a variety of purposes. Beacons can broadcast their identifier over a spectrum of distances (where the distance is typically proportional to the cost of the Beacon).

Beacon+ is based off of the iBeacon protocol, which is a popular beacon implementation designed by Apple and used by a variety of vendors in their beacons.[216] iBeacons advertise their frames at fixed intervals. The frame is composed of the following fields:[216]

- UUID: a sixteen byte unique number used to identify all iBeacons in a particular deployment.
- Major: a two-byte number used to distinguish groups of iBeacons within a deployment from other groups.
- Minor: a two-byte string used to distinguish individual iBeacons in a given group from the other beacons in that group.

Although previous work has looked at using Beacons for indoor tracking,[217–222] the insecurity of the iBeacon protocol makes it poorly suited for this task in the presence

of an attacker.

5.3 Threat Model

We describe Beacon+ as having unspoofable, temporal and authenticated advertisements; as such, we recognize the following security goals specific to Beacon+.

1. *Integrity.* Advertisements should not be modifiable by an unauthorized entity.

2. *Availability.* Advertisements should be accessible.

We omit confidentiality because Beacon+ advertisements contain no private data. Moreover, we do not claim any privacy goals for Beacon+ as the application of tracking inherently relinquishes the privacy of an asset or person.

Adversaries are distinguished based on their goals, capabilities, and relation to Beacon+. Thus, we have the following classification criteria.

1. *Active/Passive Adversary.* Active adversaries are able to read, modify, and inject advertisements; or more specifically, BLE communication. Passive adversaries are able to eavesdrop advertisements.

2. *Internal/External entity.* Internal entities have legitimate Beacon+ access (e.g., hospital administrator).

3. *Single/Coordinated group entities.*

4. *Sophisticated/Unsophisticated Adversary.* Sophisticated adversaries have access to specialized equipment (e.g., high gain antennas). Unsophisticated adversaries have access to common equipment (e.g., BLE sniffers).

Beacon+, the BLE device, smartphone, and backend server may all be used as *attack surfaces*. For example, an adversary may disrupt Beacon+ advertisements by physically destroying Beacon+ devices, or jamming or dropping advertisements. We classify Beacon+ security threats into the following categories:

1. *BLE interface threats.* Adversary is able to passively eavesdrop on advertisements, or actively jam, replay, modify, forge, or drop advertisements.

2. *Software threats.* Adversary is able to alter the logic of Beacon+ through software vulnerabilities.

3. *Application threats.* Adversary is able to compromise the intended functionality of an application.

Application-specific threats are unique to Beacon+ and non-obvious. For example, an active adversary may attempt to circumvent location-based restrictions by physically moving all Beacon+s to one central location. There exists threats with respect to BLE devices, smartphones, and backend servers that we do not cover because it is beyond the scope of Beacon+.

5.4 Beacon+

Beacon+ is motivated by the lack of temporal freshness and security in the majority of deployed beacons. In particular, these beacons lack authentication and are subject to spoofing attacks, i.e., an attacker can advertise another beacon's identification number to trick receivers into thinking that the real beacon is within range. There is also no mechanism to provide users with a sense of when an advertisement was generated because every advertisement from an individual beacon is identical.

To address these concerns, Beacon+ prevents spoofing attacks by adding lightweight authentication (e.g., message authentication codes) and provides temporal freshness by adding a monotonically increasing sequence number to each BLE advertisement. Recipients can verify the identity of the sender and the time at which the advertisement was sent. The Beacon+ design is based on the iBeacon[182] specification, with all Beacon+ data fitting into a single 27-byte BLE advertisement payload, to be universally compatible with existing BLE devices. The communication protocol is strictly one-way from Beacon+ to listening devices. Beacon+ is targeted to run on existing low-cost BLE hardware and thus we choose lightweight cryptographic mechanisms (i.e., AES-128 CBC MAC) to keep computation and power consumption low.

During initialization, each Beacon+ is assigned a unique identification number (ID), an initial value for the monotonically increasing sequence number (typically 1), and a secret key that is used to compute message authentication codes. The secret key is assigned a priori to deployment. In addition, as with current beacons, the

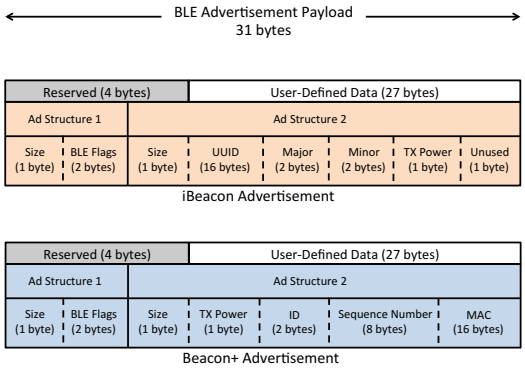


Figure 5.1: iBeacon and Beacon+ advertisement formats. BLE advertisements can support up to a 31-byte payload – 4 bytes are reserved for BLE structures and flags, leaving 27 bytes for user-defined data.

TX Power, or the signal strength to the Beacon+ at 1 meter in Decibel-milliwatts (dBm), is measured and set. The ID, current sequence number, secret key, and TX Power are stored in non-volatile memory on the Beacon+ to ensure that the values persist even if power is removed. When power is restored, Beacon+ loads its ID, sequence number, secret key, and TX Power in order to continue operation without interaction. Both the initial sequence number and the secret key are shared with the entities that will authenticate the Beacon+ advertisements and check for temporal freshness. The MAC is computed over the concatenation of the Beacon+ TX Power, ID, and current sequence number with padding. Each second, Beacon+ increments it sequence number, computes a new MAC tag, and replaces the previous advertisement with the current one.

Figure 5.1 shows the Beacon+ advertisement format compared with that of traditional iBeacons. Note that although the Bluetooth 4.0 specification indicates that there are 31 bytes to use for advertisement data, 4 of the bytes are reserved to define

and describe substructures contained within the advertisements. Therefore, only 27 bytes can be used for data. Beacon+ uses 1 byte for the TX Power, 2 bytes for the ID, 8 bytes for the monotonically increasing sequence number, and the remaining 16 bytes for the MAC. One restriction of this specific byte allocation is that only 2^{16} or 65535 unique IDs can be supported. We choose to use 2 bytes for the ID in order to allocate 8 bytes for the sequence number. Our notion for the prototype implementation was that each organization would maintain its own set of IDs, and therefore IDs would not be globally unique. However, it may be the case that more than 2 bytes for the ID are desired. One option is to assign 4 bytes to the ID and 6 bytes for the sequence number. This setup supports 2^{32} (or approximately 4 billion) IDs and provides a monotonically increasing sequence number that will survive approximately 9 million years[2].

Beacon+ broadcasts its current advertisement at a specified rate. Faster rates (e.g., eight times per second) improve the likelihood that receiving devices detect the Beacon+ if it is in range, but increase the power consumption of the underlying hardware. Slower rates conserve power consumption, but may result in receiving devices failing to detect Beacon+ that are actually in range. Currently, we configure Beacon+ to broadcast advertisements at a rate of eight times per second (every 125 microseconds), which matches the rate of many iBeacon implementations.

For Beacon+, we represent time using monotonically increasing sequence numbers

[2]Provided that the sequence number increases at a rate of once per second.

that are incremented at a regular timeout rate, i.e., once per second. When each Beacon+ is setup, the initial sequence number is shared with the entities that will collect and validate the advertisements. Upon receiving (and verifying) an advertisement from a Beacon+, the entities can compare the sequence number in the advertisement with the highest sequence number seen so far from that Beacon+. If the sequence number in the advertisement is higher than of the highest sequence number seen, the advertisement is accepted.

A more traditional approach than sequence numbers is to use timestamps measured on the local Beacon+ clock. However in practice, low-power devices that are typically used for beacons are susceptible to clock skew, which affects the accuracy and effectiveness of receiving valid Beacon+ advertisements. While Beacon+ could be setup to occasionally synchronize its clock with an external source, we chose to strictly enforce the one-way communication paradigm and avoided the problem of distributed clock synchronization.

5.4.1 Implementation

We implemented the Beacon+ specification using the Texas Instruments MSP430FR5969 LaunchPad Development Kit[223] and Bluegiga Bluetooth Low Energy BoosterPack for the LaunchPad[224] (see Figure 5.2). The MSP430 board runs the control logic of Beacon+. During initialization, each MSP430 board is assigned an ID, starting sequence number (usually 1), secret key, and the appropriately calibrated TX Power. The

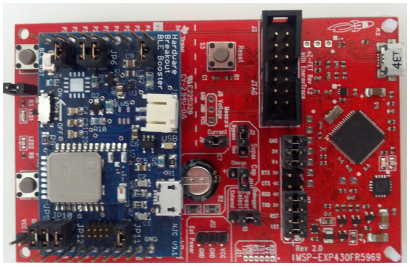


Figure 5.2: Beacon+ is implemented using the TI MSP430 LaunchPad (underlying red board) and Bluegiga Bluetooth BLE BoosterPack (top blue board).

MSP430 board is placed at a chosen location in the environment, and the ID, starting sequence number, secret key, and chosen location are shared with the backend server.

Once per the timeout rate, the MSP430 board increments the sequence number, computes the message authentication code using AES-128 bit CBC MAC, and sends the new advertisement to the BLE BoosterPack via the UART communication interface. The BLE BoosterPack receives the latest advertisement from the MSP430 and sends it out at a regular interval of eight times per second. The transmitted advertisements are then collected by devices moving throughout the environment and passed to the backend server for validation (see Section 5.5).

5.5 Applications

Beacon+ serves as a foundation for building many applications across a variety of domains. We describe three such applications enabled by Beacon+, namely secure real-time asset tracking, location-based restrictions on access control, and real-time

navigation.

5.5.1 Secure Real-Time Asset Tracking System

Using Beacon+ (Section 5.4), we create a secure real-time asset tracking system that is resilient to active and passive attacks using the temporal and authenticated BLE communication. The tracking system is composed of three components: (1) Beacon+, (2) BLE–speaking devices that will be tracked (e.g. smartphone or tablet), and (3) trusted backend server that validates Beacon+ advertisements and calculates tracked devices' positions.

The tracking system is initialized by placing Beacon+ throughout the environment at chosen locations (possibly hidden to mitigate tampering) that provide good coverage of the area. This chosen location and the Beacon+'s assigned unique ID, secret key, and starting sequence number is shared with the trusted backend server, which is run by the system administrator. Note that at any time, an administrator can return to a Beacon+ to refresh keys, apply firmware updates, or even replace it entirely. As per the specification, each Beacon+ periodically broadcasts the authenticated BLE advertisement containing its unique ID, monotonically increasing sequence number, TX Power, and the corresponding MAC of the data.

Tracked BLE–speaking devices periodically collect the authenticated BLE advertisements and corresponding received signal strength (RSSI) from all Beacon+ within range. The device then sends a device update that contains the latest collected Bea-

con+ advertisements to the backend server using some other communication medium such as Wi-Fi, cellular, or wired LAN.

This device functionality can be added to existing medical devices that support BLE with only a small modification. Note that some medical devices cannot speak directly with the backend server. In this case, devices can leverage their existing two-way BLE communication with data collectors (computers, smartphones, etc.) by passing the Beacon+ broadcasts and RSSI values to a data collector, which in turn will forward the data to the backend server. For older devices that do not support BLE, a small standalone BLE module can be assigned and attached to each one. The module will take care of collecting Beacon+ advertisements and forwarding device updates to the backend server. As long as the module stays attached to the device, the device is effectively tracked as though it was directly participating in the protocol.

To track personnel, each individual can carry their own smartphone or borrowed hospital-issued tablet. These types of computing devices are increasingly used in health-related environments due to the adoption of health information technology and Bring-Your-Own-Device (BYOD).[225] An App is installed on the devices that collects Beacon+ advertisements and sends them over Wi-Fi or cellular networks to the backend server. One concern with smartphones and tablets is the possibility that the devices will be compromised via other software running on the device. Other BLE devices, such as a ring or bracelet,[226] can be used in place of smartphones or tablets to avoid some of these concerns.

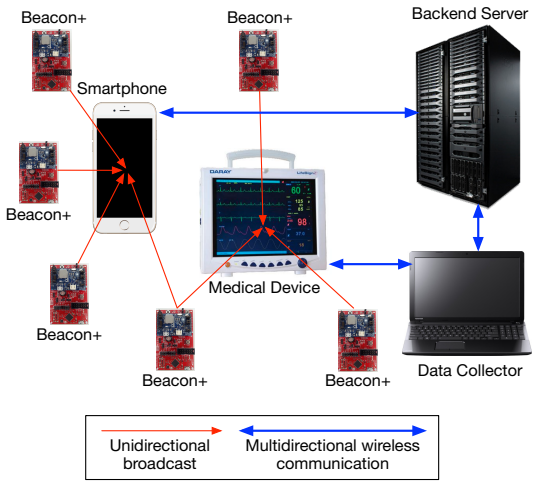


Figure 5.3: Secure Real-Time Asset Tracking System based on Beacon+. BLE-speaking devices collect authenticated Beacon+ advertisements and forward them to the trusted backend server, which calculates devices' positions.

Figure 5.3 shows an example of two different devices that are tracked. The first device is a doctor's iPhone, which can communicate directly to the backend server. The second device is a heart rate monitor that cannot communicate directly with the backend server, and relies on a data collection computer for forwarding.

Upon receiving a device update, the backend server validates each of the Beacon+ advertisements contained within that update. The backend server checks, using the shared secret key for each Beacon+, that the MAC appended on an advertisement matches the computed MAC over the data. If the MAC does not match, that advertisement is discarded and not included in the location calculation. In addition, each advertisement is checked for freshness by comparing the monotonically increasing sequence number on the advertisement with the highest received sequence number received so far from that Beacon+. If the sequence number on the advertisement

is not within a valid range of the highest sequence number seen to date (e.g., more than X sequence numbers older), that advertisement is not valid, as it may be an old advertisement that an adversary is trying to replay).

After Beacon+ advertisements in a device update are validated, the backend server can compute the location of the device. Given a device's RSSI value to a Beacon+, the backend server can calculate the distance between the two entities using the following equation:

$$rssi = -10n * log_{10}(d) + A \tag{5.1}$$

$$d = 10^{\frac{(rssi \quad A)}{10n}} \tag{5.2}$$

where $rssi$ is the measured received signal strength in dBm, A is the signal strength to the Beacon+ (in dBm) at 1 meter (i.e., the TX Power), d is distance in meters between the Beacon+ and the device, and n is the propagation constant or path-loss exponent (free space has $n = 2$ for reference, this value should be calibrated depending on the environment).

With a distance calculation between the device and at least three of the Beacon+, and with the physical location of each Beacon+ known to the backend server, the backend server can determine the location of the device using trilateration[227–229] (see Figure 5.4). The device is located at the intersection of three circles, one circle centered at each Beacon+, where the radius of each circle is equal to the distance

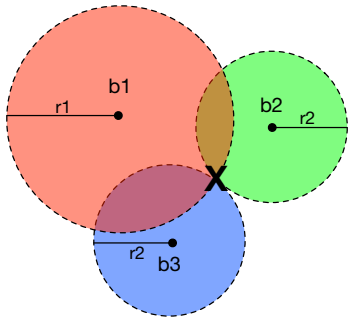


Figure 5.4: Trilateration Example. $r1$, $r2$, and $r3$ (radius of the b1, b2, and b3 circles respectively) correspond to the calculated distance between the tracked device and each Beacon+. The intersection of the three circles (marked by an X) determines the location of the device.

calculated between the device and that Beacon+. Note that in order to track a device's position at all times using trilateration, it must be within range of at least three Beacon+ in order for the computation to succeed at the backend server. Therefore, the number of Beacon+ deployed and their location should support this requirement and provide a good coverage of the area. It may be worth setting up the Beacon+ such that devices are within range of four or five Beacon+, which will provide fault tolerance in case some of the Beacon+ fail (e.g., the battery ran out of power). The resiliency gained incurs only a small financial cost since the cost of each Beacon+ is cheap.

In addition to computing a device's location, the backend server continually updates a database, which contains the location of each Beacon+, the location of each tracked device, acceptable boundaries for each device, and a log of system events. The database is read by a web application that visually displays the location of each Beacon+ and tracked devices, the boundaries of each device, and the system events

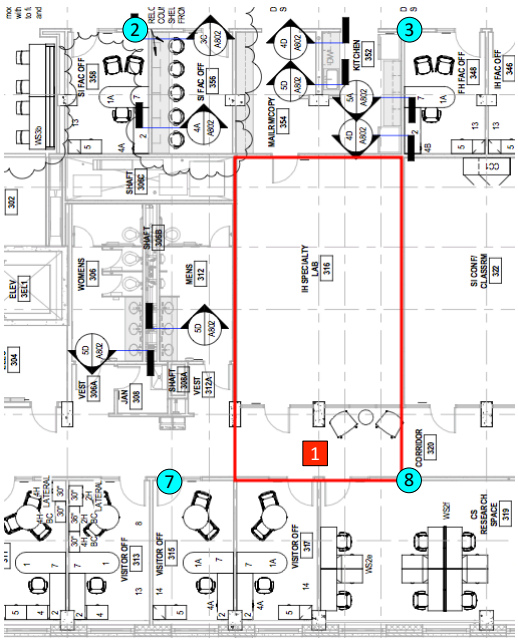


Figure 5.5: Example Web-Based Application Showing Secure Real-Time Tracking System. The blue circles are Beacon+, the solid red block is a tracked device, and the red square outline is the acceptable boundary of that device.

as they occur in real-time. The backend server and web application can take action (e.g., raise an alarm, send an email or text message) in response to problematic events, such as when a device has left or is close to leaving the acceptable boundary.

Figure 5.5 shows a snapshot of an example web application that visualizes the location of ten Beacon+ (blue circles), one device being tracked (solid red block), and the acceptable boundary of that device (red square outline) on a single floor of a university building. The web application enforces access control to ensure that the location of devices (and Beacon+) can only be seen by authorized individuals.

Future Work. We use stationary Beacon+ devices in the implementation of our

secure real-time asset tracking application. However, we note that it need not always be the case that Beacon+ devices are fixed, nor do we design our application to manage expected windows of missed advertisements due to the lack of tracked devices to forward communication. We describe these scenarios below.

Static Infrastructure Devices. Since the Beacon+ do not communicate with one another and do not communicate directly with the backend server, if there are currently no tracked devices within range of a Beacon+, the backend server will not receive advertisements from that Beacon+. In this case, it is impossible to determine if a Beacon+ is simply not in range of a device, or if the Beacon+ is experiencing a benign failure (e.g., ran out of power) or an attack. Therefore, in addition to tracked devices that move around, we can place static infrastructure devices in the environment which guarantee that each Beacon+ always has at least one active device forwarding its advertisements. We can then expect to see regular heartbeats from both Beacon+ and devices, which is helpful in defeating many potential attacks.

Tracking Beacon Movement. Up to now, the Beacon+ were considered stationary. It is useful to detect if a Beacon+ has moved locations. With the original designated location for each Beacon+ and previous locations of devices and staff, the backend server can distinguish normal from problematic Beacon+ advertisements forwarded by tracked devices. Two examples include: (1) If a Beacon+ advertisement appears in a location far from its designated area, it was likely moved (2) If a Beacon+ advertisement is missing from locations that it normally appears in, the Beacon+ may

have moved, the battery may have died, or it may not be functioning correctly due to tampering. In essence, the Beacon+ and assets can track each other simultaneously. Moreover, current beacon hardware has the capability to provide analog sensor data such as temperature and accelerometer data and we can include this data in additional advertisements to indicate movement as it occurs (rather than solely relying on detecting movement a posteriori).

Attack Mitigations. An active inside or outside adversary may steal a device for monetary gain or to cause harm. This attack is defeated because devices are tracked in real-time and authorities are notified if devices are being moved from their intended locations. In addition, this type of adversary may physically damage a Beacon+, remove the power source from a Beacon+, or perform sophisticated communication jamming. Since the tracking system expects Beacon+ advertisements and device updates (i.e., heartbeat), the backend server can implement a detection policy (much like a network intrusion detection system) that generates alerts on unexpected behavior. Or, the backend server can generate audit logs for retroactive analysis.

5.5.2 Location–Based Restrictions on Access Control

Sensitive data such as electronic medical records are protected using encryption and single-factor access control mechanisms (e.g., PIN numbers, passwords) to limit access to authorized individuals. However, this approach raises a major security concern: an adversary that is able to bypass or break the access control security gains access to all of the sensitive data in the database with a single breach. This threat is made worse in the context of a hospital, where computing devices are often used to access sensitive patient information, and a stolen or compromised device can provide an attacker with a large portion of private data.

To address this threat, we implement a prototype application that provides an access control mechanism that enforces location-based restrictions. The application relies on the authenticity of received Beacon+ advertisements to compute the relative location of an authenticated device compared to an asset, and provides access to the asset data if and only if the device is within close physical proximity. In the hospital setting, nurses and doctors who are away from their personal computer but moving around with their smartphone must be within close physical proximity of a patient to access her medical record. With this scheme, an access control breach only results in a small fraction of sensitive data leakage, since an attacker that steals an authenticated device only gets access to data that is within close proximity. The location is only

one factor in a multi-factor access control scheme to authenticate a user.

Implementing the location-based restrictions application requires only minor additions to the secure real-time tracking system. Personnel can use the same smartphone or BLE device they sign into for the tracking system to access sensitive data. As personnel move about the organization, the backend server tracks their location. When the tracked device enters close proximity of assets, the backend server checks the credentials of the device and authenticity of the Beacon+ advertisements and sends the device the appropriate data from assets in range. Similarly, when devices leave close proximity of an asset, the backend server revokes access to that asset's data and the App removes the record[3]. The backend server can choose the level of granularity on which to enforce location-based restrictions. For example, in the hospital context, the backend server may choose to organize patient records based on room, rather than solely using distance as the metric. In addition, the backend server can tailor the information sent to the devices based on the credentials of the user (e.g., doctors may be sent more sensitive information about a patient than nurses).

This approach provides location-based restrictions without the need of additional authentication at every step. While an adversary that steals one of these authenticated devices can see the sensitive information about nearby patients, the threat is not much different from the existing accepted threat in which an adversary could walk around the hospital and take the paper medical records that often sit unat-

[3]The App is also setup to remove data from the display after a configurable timeout, which protects against an adversary that cuts network communication in an effort to force an asset's data to persist on the screen even after moving out of range of the asset.

tended outside of patient rooms. One possibility is to have doctors re-authenticate upon entering each room. This prevents an attacker from walking around with a device to get basic patient information, but puts a burden on doctors and nurses. This is a trade-off between privacy and usability which can be set as desired, and the App supports both configurations.

In some cases, a doctor might require accessing more details of a patient's health records or may require accessing a medical record for a patient that is not in the same room. In this case, the App on the device allows doctors to provide further forms of authentication (e.g., fingerprint, additional password) to increase their access. Note that this access is only provided temporarily each time additional authentication is provided, preventing an adversary from breaking the location-based restrictions if the device is stolen. Additionally, doctors can always return to their private offices to use traditional access control techniques to gain access to a wider range of medical records.

By using location-based restrictions for access control, hospitals get the technological and convenience benefits of electronic medical records with the traditional privacy model of paper medical records, in that successful attackers only get access to localized sensitive information rather than access to a large database of many records.

Attack Mitigations. An active inside or outside adversary may perform a denial-

of-service attack on the tracking system to cause patient harm or thwart productivity. This attack is mitigated by having authorized individuals use additional authentication methods to bypass the location-based restrictions to temporarily gain access to the records, use paper records as a backup, or return to their office to access records. In addition, this type of adversary can steal a device that is used to obtain sensitive data via the location-based restrictions application. This attack is mitigated by having the device delete records when it is moved out of range of assets, as well as having devices configured to delete records after a specified time elapses.

Note that the location-based restrictions application requires that the backend server have knowledge of patient locations in the hospital (either at a physical location or at room-level granularity). The tracking system can be made to track patient locations by associating BLE devices with patients, or the backend server can link with existing hospital management techniques that track patient locations.

5.5.3 Real-Time Navigation

In addition to security-driven applications, it is possible to provide user convenience applications that run on top of the tracking system. One application that can be useful in large organizations is a real-time navigation system for individuals that are unfamiliar with the premises. For example, a patient or visitor in the hospital may have a difficult time navigating the facility to find a particular patient room or a specific doctor's office. Moreover, hospital staff may be burdened by frequently hav-

ing to help visitors find their way and prevent them from wandering into restricted areas.

The secure real-time tracking system can be used to provide real-time navigation. In the hospital setting, patients and visitors are given a hospital-issued BLE device (e.g., the same tablet that doctors or nurses are given) and sign in as a "guest" with no security credentials. The device has a Navigation App that allows the guest to enter a target room number or select a destination that is listed on the App (e.g., cafeteria, cardiology center). The backend server tracks the device as the guest moves through the hospital and sends the location back to the guest's Navigation App. With the current location and target destination, the App calculates the path that the guest should take to get to the destination. This path is visually displayed as an overlay on the device that visitors can easily follow, similar to GPS for indoor navigation. In addition, the App tells visitors (or raises an alarm to alert staff) when they enter or are close to entering a restricted area, reducing the burden on staff and raising staff efficiency.

5.6 Experiments

We deployed the Beacon+ prototype in our environment to emulate a typical deployment setting in order to evaluate the accuracy of the Beacon+ tracking system, the feasibility of using Beacon+ compared to traditional beacons, and the effectiveness

of using Beacon+ to enforce location-based restrictions on access control to sensitive data.

5.6.1 Experimental Setup

We deployed 8 of the Beacon+ prototypes, evenly spaced to obtain good coverage of one side of the floor in our building, which emulates a setup similar to one that would be used in typical hospital settings. Each Beacon+ was placed at its chosen location and assigned a unique ID and secret key that is shared with the backend server. Upon startup, each Beacon+ begins broadcasting an authenticated BLE advertisement containing its unique ID, latest sequence number (monotonically increasing once per second), calibrated transmit power at 1 meter, and MAC. Advertisements are broadcast every 125 microseconds (8 times per second), matching the specification of most iBeacon implementations. We experimented with several values for n, the propagation constant from equation 5.1, and ultimately decided on $n = 2.7$ for our experiments. It provided the most accurate measured distance from Beacon+ prototypes compared with the actual location of the tracked devices.

In our setup, we used a Google Nexus 4 smart phone as the tracked device. We created an Android App to periodically scan and collect all Beacon+ advertisements within range (aggregating the measured RSSI values for each Beacon+ ID). The collected advertisements are then bundled into a device update and sent via Wi-Fi to the backend server, which authenticates each of the advertisements in the update

and calculates the position of the device. After calculating the position, the backend server updates the position of the device in a SQL database (to graphically display on a website), and in the case of the location-based restrictions application (see Section 5.6.4), sends a message back to the device containing the relevant information to display on the screen.

5.6.2 Tracking System Accuracy

To measure the accuracy of our Beacon+ tracking system, we placed the device at various locations and compared the calculated location from the tracking system with the actual location in the building. Initially, we measured the accuracy using the trilateration approach, using the measurements from the three Beacon+ prototypes with the strongest received signal strength for that update. However, we found that the measured signal strength from our BLE hardware contained a fair amount of noise, often causing the trilateration calculation to fail (the resulting circles created from the distance measurements did not intersect).

In general, trilateration provides a very accurate position calculation, but has rigid requirements that may be hard to meet with inexpensive hardware producing noisy measurements. Moreover, in the case of the tracking system, if trilateration fails for a device's update, the tracking system is essentially blind to changes in a device's position until a future update arrives on which trilateration succeeds. While this is not an issue if the device being tracked is predominantly stationary, if the device is

usually moving throughout the environment (e.g., nurse or doctor), failures to update the tracking system at a regular interval can be problematic. Therefore, rather than using trilateration in our experiments, we calculated the position of devices using an approach that is less accurate, but more flexible. We describe this approach in detail next.

Translated Midpoint Method. For each device update received, the backend server sorts the valid Beacon+ advertisements in order of received signal strength and can calculate the device's position for this update as long as at least two advertisements are valid (see Figure 5.6). If there are three or more valid advertisements, the backend server uses the top three Beacon+ ads (based on RSSI values) and forms a triangle, with one vertex corresponding to each of the Beacon+ locations in the environment. Each vertex is then translated toward the midpoint of the opposite side of the triangle, with translation distance proportional (or in our case, equal) to the measured distance between the device and that Beacon+. If there are only two advertisements, a line is formed between the two Beacon+ locations, and each point is translated toward the other point with a distance equal to the measured distance from the device to that Beacon+. Finally, the device's position is calculated as the centroid of the resulting triangle (in the case of three valid Beacon+ advertisements) or midpoint of the resulting line (in the case of two Beacon+ advertisements).

Using the new approach, the resulting Beacon+ tracking system is flexible and accurate, providing a position calculation with precision of 1-2 meters in the best case

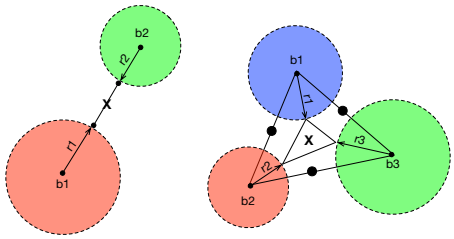


Figure 5.6: Translated Midpoint Method to calculate device position. With two valid Beacon+ advertisements, a line is formed between the two Beacon+ locations ($b1$ and $b2$). $b1$ and $b2$ are translated toward each other with distance $r1$ and $r2$ respectively, where r is the measured distance between the device and the Beacon+. With three Beacon+ advertisements, a triangle is formed between $b1$, $b2$, and $b3$. Each vertex is translated toward the opposite side of the triangle with distance $r1$, $r2$, and $r3$ respectively. In both cases, once the vertices are translated, the device position is calculated as the midpoint of either the resulting line or triangle (marked by an X in the example).

and 9-10 meters in the worst case. Compared to the trilateration approach, the translated midpoint method achieves a better overall tracking system in the environment of our experimentation.

5.6.3 Beacon+ Power Consumption

Estimating the production power consumption of Beacon+ is difficult due to the fact that our prototype is running on an MSP430 LaunchPad development board and thus power consumption is not indicative of real-world performance; the development components of the board significantly contribute to the overall power draw. However, we were able to provide a rough estimate of the overhead of Beacon+ compared to a standard beacon by performing power analysis of our development board running as both a standard iBeacon and Beacon+, and comparing the results.

To perform our power analysis, we connected an MSP-430 LaunchPad to an Agilent programmable power supply. Since the MSP430 LaunchPad runs off of a +5V power source, we set the output voltage to 5 volts and maximum current to 1 A (much higher than was needed). Our power supply showed that in the case of the MSP430 emulating an iBeacon, the power draw was between 15 and 20 mA. In the case of the MSP430 emulating a Beacon+, the power draw was between 22 and 25 mA. Therefore, the overhead of Beacon+ over a standard Beacon running on our test platform was between 20% and 46%.

5.6.4 Location-Based Restrictions on Access Control

With an accurate tracking system in place, we evaluate the practicality of using Beacon+ to provide an access control mechanism that enforces location-based restrictions. We created an Android App that collects and forwards Beacon+ advertisements to the backend server and displays database records (i.e., patient data) that are sent in return. After validating a device and calculating its position, the backend server compares the device position with the location of patients in the building and only sends the device the records of nearby patients, i.e., the patients that are within some threshold distance (10 meters in our experiments). When a device moves out of range of a patient, that patient record is removed from the list in the App, since

the backend server sends a new update that does not contain that patient record.

For this experiment, we created a mock patient record database on the backend server based on the OpenMRS Demo Data,[230] and set the location of four of the patients in the database to locations in the building environment (yellow squares shown in Figure 5.7). Then, we walked around the building with the smartphone running the App to view the records of the nearby patients, i.e., the patients that were within 10 meters of the device's tracked position.

Figure 5.7 shows four snapshots (a through d) of the experiment in action. The visual GUI of the Beacon+ tracking system is shown on the right. The GUI shows the location of the Beacon+ prototypes (blue circles), the patients in the building (yellow squares), and where the device is located at each snapshot (a through d). For each snapshot in Figure 5.7, we also include the screen capture of the device running the patient record access App at its respective location on the map.

In Figure 5.7a, the device is only within range of one patient (#1), and as a result the App only displays the record for patient 1, "John Smith." The patient locations were purposely set in the environment such that there is only a small area in which a device can be in range of all four patients simultaneously. Figure 5.7b shows this case, and the corresponding screen capture shows that the device can see the records of all four patients. In Figure 5.7c, the device is now out of range of the two patients on the southern side of the building, and only the two nearby patient records are available to view in the App. At any time, the user can click on one of the displayed records

Figure 5.7: Location-Based Restrictions on Access Control. 8 Beacon+ prototypes (blue circles) and 4 patients (yellow squares) are located in the environment as shown by the Beacon+ tracking system GUI (right side). The red square shows the location of an authenticated device (smartphone) that is requesting patient records at four locations (labeled *a* through *d*). At each location, the screen capture of the App running on the phone is shown, indicating which patients are in range (and thus which records are accessible). Note that in *d*, no patients are within range of the device. *e* shows patient-specific information that can be viewed in the App for patients currently in range.

in the App to view patient-specific information, e.g., current doctor and medication for that patient. This feature is shown in Figure 5.7e. Finally, if the device is out of range of all patients, the App will clear the screen, not showing any patient records – this situation is shown in Figure 5.7d.

The experiment shows that the device is only able to access records of patients that are within close proximity – the App cannot obtain records for patients that are not within range, nor patients that are in the database but not currently located in the environment. This validates that Beacon+ can provide another factor for multi-factor authentication, namely location information, to enforce access control of sensitive data.

5.7 Conclusion

In this work, we have shown that Beacon+, a BLE device that transmits short-range unspoofable, authenticated advertisements, can be used to implement multiple location-based applications. Specifically, we implemented a secure real-time tracking system that uses Beacon+ to more securely track assets and provides a foundation for two related applications: real-time indoor navigation, which we described, and location-based restrictions on access control, which we implemented. Lastly, we validated the accuracy and feasibility of Beacon+ by deploying low-cost BLE hardware prototypes in our environment and running our applications.

Chapter 6

KBID: Kerberos Bracelet Identification

6.1 Introduction

Modern computer systems require a user to prove his or her identity through an authentication process. While a variety of authentication processes including public key authentication, biometrics, and passwords exist, passwords remain the most widely used authentication mechanism because of their ease of use and implementation. However, password-based authentication is only as strong as the passwords that users choose, and users often choose passwords that are too simplistic and easily guessed.[231] Thus, administrators encourage or force users to select more complex and longer passwords, decreasing user satisfaction as password selection becomes increas-

ingly difficult.

Exacerbating the initial selection challenges, the likelihood of entering a password incorrectly during subsequent uses increases with password complexity and length, as does the time to enter and re-enter it. In certain settings such as healthcare environments, complex passwords may not be compatible with the time-sensitive nature of the workflow, as such passwords may be too cumbersome to repeatedly enter.

To address this issue, we introduce KBID. KBID is a design for a wearable authentication system that allows its user to enter a password as infrequently as once per day. Our proposed system stores authentication information on a wearable bracelet and transmits this information to devices the user wishes to authenticate. The transmission between the bracelet and device is achieved by touching the bracelet to a contact terminal, and, therefore, is not vulnerable to existing radio attacks on wireless authentication technologies. Our goal is to reduce the impact on user satisfaction and efficient workflow without compromising the security of the underlying system by removing most of the difficulty of using a complex password.

Of course, although our direct-contact bracelet is not vulnerable to RF-based attacks, it could have still be vulnerable to other types of attacks such as man-in-the-middle attacks and replay attacks. However, we have designed our system to be robust against these types of attacks as well, as we will show in Section 6.5.

Our KBID design can authenticate a user in less than one second given that

user has already provided valid credentials once after putting on the bracelet for that session, such as at the beginning of his or her shift. We believe that such a system will reduce the resistance to strong password requirements by allowing users to enter their passwords infrequently. To put it simply, users will be more willing to use a strong password if they only have to enter it once a shift as opposed to being prompted to enter the password each time they use a computer system.

We believe KBID would be particularly well suited to solving authentication problems in healthcare environments. In such environments, workers often encounter emergency situations where they do not have time to remember and enter a long and complex password. In such a situation, KBID would allow the healthcare worker to authenticate instantaneously, thus ensuring patient privacy while simultaneously allowing instant access. KBID could further be modified to allow a user to log-in even in the event of a failed authentication as long as the log-in is registered so that the access could be appropriately scrutinized after the emergency situation is resolved.

6.2 Background

KBID originated from the idea of integrating a wearable device to achieve some additional property in an authentication system (e.g., de-authentication). Its design was partially inspired by the design of zero-effort bilateral recurring authentication (ZEBRA) by Mare et al.,[62] which is discussed in the related work section. A fundamental

limitation of ZEBRA is that it was only envisioned as a method to de-authenticate a user from a computer system and did not include a way to initially authenticate the user to the system. Starting with the assumption that a ZEBRA user has already consented to wearing a deauthentication bracelet, we looked to add the functionality necessary to support rapid authentication as well in order to increase the usefulness of the system.

We avoided using radio frequency (RF) technologies because RF emits information into the environment. That information, once emitted, can be received and rebroadcasted. This opens up RF-based systems to active attacks such as ghost-and-leech attacks.[232] Ghost-and-leech attacks occur when an attacker uses a more powerful radio transmitter than the transmitter found on a wireless device in order to capture and rebroadcast the wireless signal; the goal is to fool a target system into believing that the wireless device is in closer proximity to it than it actually is at the time. While it may be possible to avoid some such attacks through the use of cryptography, our system eliminates this attack surface entirely by requiring that the user be in physical contact with the authentication device to transmit authentication information.

In the medical context, an attacker could capture and rebroadcast the authentication signal off of an authentication wearable in order to log in to a computer terminal even when an authorized user is not near the terminal. For example, in a hospital, a doctor might wear an authentication bracelet which continuously broad-

casts a changing login credential using Bluetooth Low Energy. An attacker might place a receiver next to a target doctor's office. The receiver could then capture the signal and send it to a portable rebroadcaster, which the attacker could use to unlock a computer terminal far from the doctor's office. Due to the wireless nature of this type of authentication wearable, these attack approaches can be extremely difficult to protect against.

In addition to this concern, RF also relies on the security of the underlying communication infrastructure. If a security flaw is discovered in an RF communication framework, e.g. Bluetooth low energy (BLE),[233] then the system as a whole could be vulnerable to the flaw.

Furthermore, we designed KBID such that an authenticated bracelet is non-transferrable (i.e., a user cannot authenticate and then give the bracelet to someone else). To support this feature, our design zeros all authentication information upon bracelet removal. To facilitate data transmission using KBID, our prototype uses direct contact with an authentication plate to transmit the authentication information. In the future we believe that KBID could also be implemented using body-coupled communication (BCC) to transmit its information over human skin.

6.3 Related Work

In addition to the deauthentication work of ZEBRA which was briefly mentioned in the previous section, there have been a few other previous attempts at developing wearable authentication technology. Two existing technologies of note include the Bionym Nymi[234] and the Intel Authentication Bracelet,[235] but both are wireless devices.

6.3.0.1 ZEBRA: Zero-Effort Bilateral Recurring Authentication

ZEBRA[62] was developed to solve a very different problem than the authentication bracelets. Instead of authenticating a user to a computer terminal, ZEBRA is focused on detecting when a user is done using the computer terminal and deauthenticating the user from the terminal even if the user forgets to do so. In ZEBRA, a user wears a bracelet that encapsulates a wireless radio, accelerometer, and gyroscope; these components record and transmit wrist movements to a computer system that is currently being used. The computer system continually compares received movement measurements to input it receives from its keyboard and mouse. If these two measurements are not correlated, the current session is de-authenticated.

6.3.0.2 Bionym Nymi

The Bionym Nymi[234] is an authentication wristband that broadcasts a digitally signed authentication signal derived from a user's heartbeat to nearby devices using Bluetooth Low Energy. The Nymi extracts reproducible but unique features from the user's ECG waveform. The Nymi is meant to be used as a biometric authentication system and is envisioned as operating without any user input.

6.3.0.3 Intel Authentication Bracelet Prototype

Intel, in partnership with Fossil, has announced a prototype for an authentication bracelet based on its Curie platform.[235] The bracelet is designed to automatically unlock a device when a given user is nearby. When the bracelet is first put on, the user must log in to a bracelet-enabled device with a standard password. However, subsequent logins to bracelet-enabled devices do not require a password and the bracelet will continue to broadcast an authentication token until the bracelet has been taken off, causing the device's volatile storage to lose the authentication credential. The bracelet uses Bluetooth Low Energy to broadcast its credential to nearby bracelet-enabled devices.

6.4 Threat Model

KBID is an authentication mechanism that requires a wearable device to transmit short-range authentication data through direct contact with a contact plate. We recognize confidentiality, integrity, and availability as security goals specific to KBID. Particularly, data stored on the device and transmitted to the authentication module should be kept secret from and not modifiable by unauthorized entities, and the data should be accessible to both bracelet and authentication module. We are not concerned with the privacy of the authenticating user because the system must validate her access to the system or resource.

As a hardware and software solution, the threat model for KBID includes many subjects that apply to any such system. These include attack types (denial of service, message forging or tampering, hardware tampering, and others) as well as a study of potential adversaries and other topics. Here we focus on two threats that are unique to KBID. These threats require an active adversary that has the ability to enter within close proximity of where KBID is used. For example, it may be possible to impersonate an authentication module. An attacker could use a counterfeit authentication module that issues *Get Status* commands when a user touches something connected to it. Our design accounts for such an attack by integrating a challenge-response protocol into the handshake between the bracelet and the authentication module.

6.5 Design

Here we describe in detail the design of the KBID system. First we discuss the high level design and explain the four major components of the system. Next, we discuss the system workflow. Finally, we present the authentication protocol.

6.5.1 High Level Design

The system is composed of four main parts: a bracelet (Figure 6.1), an authentication module (Figure 6.2), an authentication client, and a Kerberos authentication server. The bracelet is a wearable device that fastens to the user's wrist. The bracelet makes contact with the user's skin and applies a signal directly to the user's skin. The authentication module has a sensor with a button under it. When the user touches the sensor and depresses the button, the authentication module initiates communication with the bracelet. The authentication module is attached via RS-232 serial to the computer system the user wants to authenticate. A workstation hosts the authentication client. The client monitors the serial connection for data and, when necessary, opens a connection to the Kerberos server for authentication. Finally, the Kerberos server is a default installation and uses the default implementation of the authentication protocol.

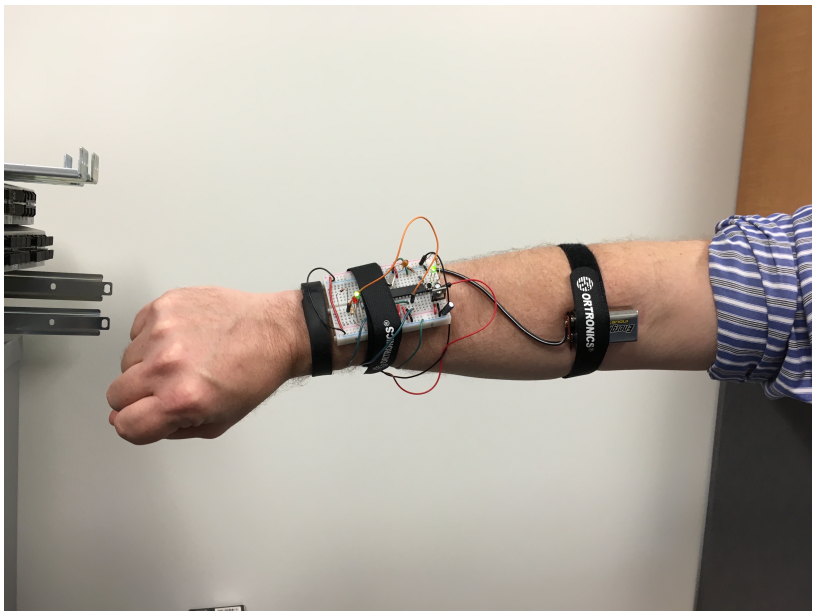


Figure 6.1: KBID Prototype Bracelet

6.5.2 System Workflow

The system workflow can be described in two use cases; for the sake of brevity we do not include any error handling. In the first use case (Figure 6.3) the user is wearing a bracelet but the bracelet is not yet authenticated. The user touches the bracelet to the contact pad on the authentication module, the authentication module sees that the user's bracelet is not authenticated, and the authentication module then relays this information to the authentication client. The client prompts the user for their username and password. The client verifies this information with the Kerberos server, then instructs the user to touch the sensor on the authentication module again. The client then instructs the authentication module to write the authentication information derived from the ticket (described below) to the bracelet.

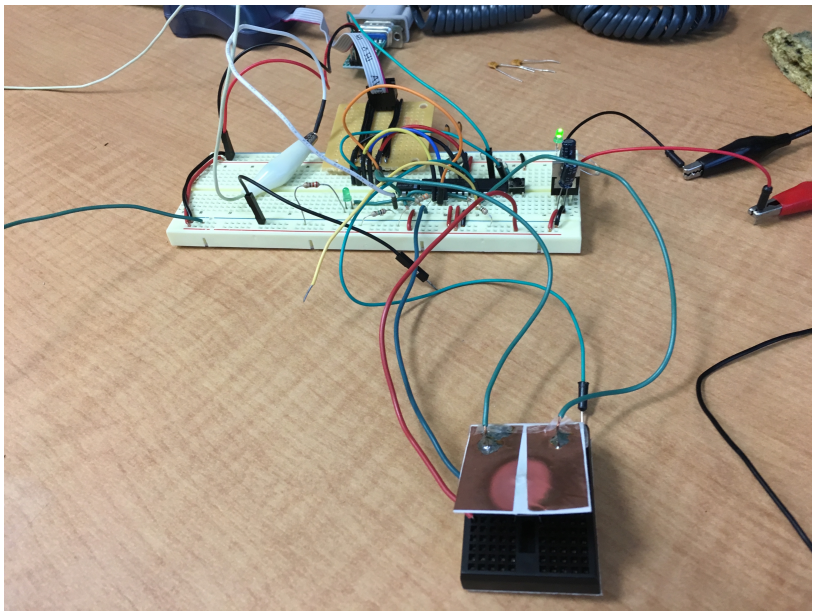


Figure 6.2: KBID Prototype Authentication Module

Once this information has been written, the client unlocks the workstation.

In the second use case (Figure 6.4) the user has already authenticated the bracelet. The user touches the sensor on the authentication module. The authentication module sends a challenge and asks for a status, and the bracelet provides responds to the challenge by sending it the transformed authentication information (described below). The module passes this information along to the client which uses the information to recover the Kerberos ticket. The client verifies the ticket with the Kerberos server and unlocks the workstation. Our goal is to perform this use case in less than one second.

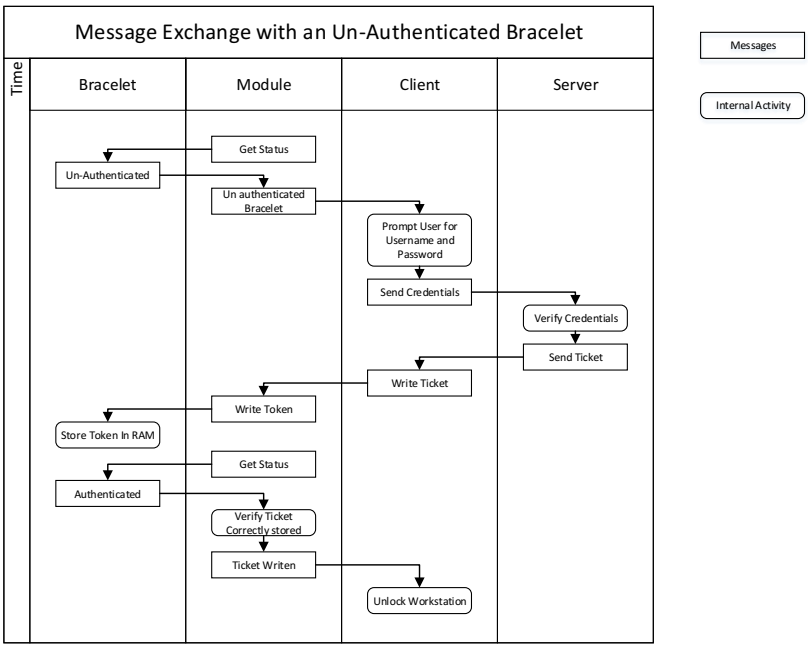


Figure 6.3: Un-Authenticated Message Exchange

6.5.3 Authentication Protocol

The actual protocol used to store security information on the KBID bracelet is designed to be as small and lightweight as possible in order to fit within the limited resources of our prototype implementation bracelet. Despite our emphasis on size, we have made no security tradeoffs whatsoever to achieve our goals.

6.5.3.1 Authentication Protocol Workflow

To authenticate to a terminal for the first time, a KBID user touches the unauthenticated bracelet to the authentication module, enters a password, and touches the bracelet again. The terminal uses the password to authenticate the user to a Kerberos server. Provided that the authentication information is correct, the Kerberos server

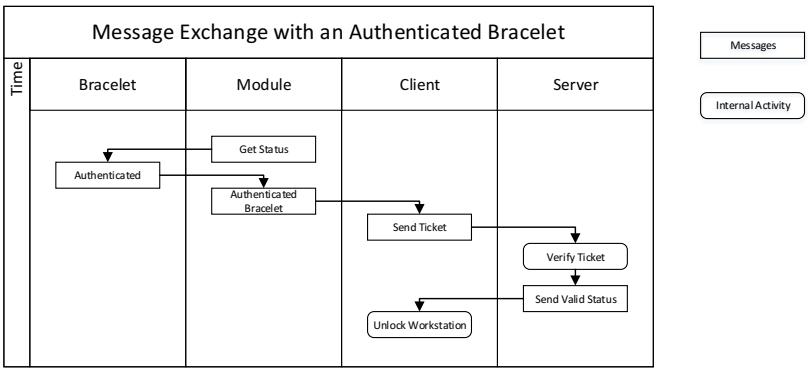


Figure 6.4: Authenticated Message Exchange

sends a Kerberos ticket to the terminal. The terminal saves the ticket, hashes it, and generates a random key. It pushes the hash value, H and the key, K, to the bracelet, which is now considered to be authenticated.

When the user wants to authenticate to the terminal in the future, the user touches the bracelet to the authentication module. The terminal recognizes the bracelet and sends a nonce, N to the bracelet. The bracelet responds by sending $ENC_K(N|H)$ to the terminal. The terminal, knowing K, H, and N, decrypts the message in order to get $(N|H)$. It computes $N|H|N$ in order to recover the hash value of the Kerberos ticket, H. Having obtained this value, the terminal looks up the actual ticket in its table using H as its key. It uses this ticket to re-authenticate the user. The nonce in the protocol prevents replay attacks. Note that this system could be vulnerable to man-in-the-middle attacks. However, we believe that hardening the bracelet against such attacks is not worth the added space requirements, given that such attacks would be the result of a physically compromised authentication module. If the authentication module is deployed with suitable care then such tampering

should be easy to identify.

6.6 Implementation

We have built a hardware prototype for the KBID bracelet and the authentication module. Our prototypes are based on the Atmel ATMega328 microcontroller operating at 20 MHz, and an LM358AN Amplifier. Both the bracelet and the authentication module have copper pads that make contact with each other. The prototype for the authentication client has been written in Python. We also implemented the functionality that clears the authentication information when the bracelet is removed. This was accomplished by using one of the hardware interrupts on the microcontroller.

6.6.1 Interfaces and Communication

The KBID system includes three interfaces: the interface between the bracelet and the authentication module; the interface between the authentication module and the authentication client, which takes place over RS-232 serial; and finally, the interface between the authentication client and the Kerberos server, which uses the network.

6.6.1.1 Bracelet to Authentication Module

The communication protocol between the bracelet and the authentication module is a very lightweight protocol. The messages that the bracelet sends to the authen-

tication modules are called *statuses*. The messages that the authentication module sends to the bracelet are called *commands*. Each message sent over this interface is a length delimited series of bytes. A status message has the following structure: [Status ID] [Device ID] [Data Size (in bytes)] [Data]. The bracelet will send one of two statuses, *authenticated* or *un-authenticated*. If the status is authenticated, the authentication data will be transmitted in the data field.

Status ID	1 Byte
Device ID	4 Bytes
Token Size	2 Bytes
Token	0 to 65535 bytes

The Status ID will be sent as 0x01 if the bracelet is not authenticated and 0x02 if the bracelet is authenticated. The Bracelet ID will always be transmitted when the bracelet sends a status - it is a unique identifier for the specific bracelet. If the bracelet is not authenticated the Token Size field will be set to 0x00 and the token will be of length 0. If the bracelet is authenticated then the Token Size field will reflect the length of the token that follows. Naturally, the token data will be of a length defined by the token size field.

A command message has the following structure: [Command ID] [Device ID] [Payload Size (in bytes)] [Payload]. These fields are delimited by length according to the following table:

Command ID	1 byte
Device ID	4 bytes long
Payload Size	2 Bytes
Payload	0 to 65535 bytes

The authentication module will send three commands. The first command is a "Get Status" command; this is the only command to which the bracelet will respond. The Command ID is 0x01. The Device ID is set to a special value of 0xFFFFFFFF which the bracelet will regard as a broadcast ID, and the payload size is 0x0000 with a payload of length 0. When the bracelet receives a "Get Status" message it responds with a status message.

The second command is a "Set Token" command. The Command ID is 0x02. The Device ID is set to the ID of the bracelet, and the payload size will be set to the length of the token to be stored. The token will be appended after the message. When the bracelet receives a "Set Token" command it will store the payload in memory and set its status to "Authenticated."

Finally, the authentication module can send a "De-authenticate" command. The Command ID is set to 0x03 and the Device ID is set to the device ID of the bracelet to be authenticated. The Payload Size is 0x00 and the payload is of length 0. When the bracelet receives this message it will clear any token it has, byte by byte and set its status to "Un-Authenticated."

6.6.1.2 Authentication Module to Authentication Client

The authentication module and the authentication client communicate status and command messages as well. The authentication module can send three statuses to the authentication client. First is the *Un-authenticated Bracelet* message. This message is sent to the authentication client when the authentication module receives an un-authenticated status from a bracelet.

Next the authentication module can send an *Authenticated Bracelet* status to the authentication client. It will send this status when the bracelet sends a status of authenticated. The Authenticated Bracelet status will contain the ticket information that was in the token section of the bracelet's message.

Finally, the authentication module can send a *Ticket Written* status to the authentication client. This is a message that lets the client know that the ticket information has been successfully written to the bracelet.

The authentication client sends two commands to the authentication module. First the *Write Ticket* command. This instructs the authentication module to pass the ticket included in the command to the bracelet with a Set Token command. The authentication client can also send a *De-authenticate Bracelet* command. This instructs the authentication module to issue a De-Authenticate command to the bracelet.

6.7 Future Work

Future work will focus on improving the system.

6.7.1 Hardware Upgrades

One could implement the system using a microcontroller that can store a larger amount of authentication data. The Atmel ATMega328 only has 2 kilobytes of ram. Since the systems needs some RAM to perform operations, the amount of authentication data that can be stored in RAM is limited to about 1 kilobyte. While 1 kilobyte of RAM is sufficient for the Kerberos-based security protocol that we propose in this paper, this amount of RAM may not be sufficient enough to store authentication information in some real world use cases.

6.7.2 Password-less Authentication

In the long term, a future project could continue the development of the system model to include a system where passwords are not required at all. In one such implementation, a user would check in with a security technician who would personally and physically validate the user's identity then provide the user with a bracelet and assist with authenticating the user without the user entering any information into a computer at all.

6.7.3 Body-Coupled Communication

We believe that a natural extension to KBID would be to allow the bracelet to use body-coupled communication in order to provide even more seamless authentication. Using BCC has been proposed in other work as well. For exmaple, Chang et al. introduce a system for key exchange over a body area network.[59] By applying a very small voltage to the tissue of a dead mouse, they were able to communicate at a rate of 5Hz or 5 bits per second. However, this data rate is not acceptable for our work as we would need to communicate authentication data of at least 256 bits, and this would take nearly a minute to transmit. Thus a BCC implementation of KBID has been left as an exercise for future work.

6.7.3.1 Preliminary Tests

We ran some preliminary tests on integrating BCC into KBID in order to determine the feasibility of implementing BCC in a future project. We are able to send commands from the authentication module to the bracelet. The bracelet can correctly interpret those commands and it responds when issued a Get Status command.

Our test results showed that a full BCC-based implementation would have two major hurdles to overcome. First, in order to successfully send a signal, the bracelet and the authentication module must have a common reference for voltage. Second, while we have been able to get the signal to transmit from the authentication module to the bracelet, we have not been able to get the signal to travel in the opposite

direction and arrive in a way that the signal can be interpreted by the authentication module. To solve both of these issues, we believe that one could instead use capacitive coupling to transmit data over the skin.

6.8 Conclusion

Password-based authentication is only as strong as the passwords that users choose, and users often choose passwords that are too simplistic and easily guessed. Thus, administrators encourage or force users to select more complex and longer passwords, decreasing user satisfaction as password selection becomes increasingly difficult. In certain settings such as healthcare environments, complex passwords may not be compatible with the time-sensitive nature of the workflow, as such passwords may be too cumbersome to repeatedly enter. In this work we have presented KBID, a design for a Kerberos-based identification bracelet. KBID solves authentication challenges in fast-paced environments in a way that exposes a significantly smaller attack surface than existing solutions. KBID uses a lightweight and efficient protocol to secure its communications can be used in a clinical setting in order to improve the efficiency of modern healthcare in the context of privacy protection.

Chapter 7

Summary

In this dissertation we have explored many aspects of security and privacy in the medical space. We have presented tools to automatically audit accesses in electronic medical record systems in order to proactively detect privacy violations; to fingerprint network-facing protocols in order to non-invasively survey devices vulnerable to known attacks; and to authenticate healthcare providers to medical devices without a need for a password in a manner that protects against all known attacks present in radio-based authentication technologies. We also presented an extension to the widely-used beacon protocol in order to add security in the face of active attackers and we demonstrated an overhead-free solution to protect embedded medical devices against attacks that leveraging built-in device modes in order to maliciously reconfigure the device.

The technologies proposed in this dissertation will serve to transform the health-

care environment of the future. They will work together to form a new technological system that improves security and privacy while simultaneously making the job of a healthcare provider easier and more efficient. In many cases, healthcare providers resist integrating new security and privacy technologies into their workflow because they believe that these technologies add overhead that increases the complexity of their existing workflow. In contrast, the future healthcare environment proposed in this work does the opposite–it improves security and privacy while simplifying the healthcare providers workflow. We believe that these approaches represent the future of security and privacy in healthcare.

Bibliography

[1] P. D. Martin, D. Russel, Y. Yu, M. Ben-Salem, S. Checkoway, and A. D. Rubin, "Sentinel: Secure mode profiling and enforcement for embedded systems," 2016.

[2] P. Martin, A. D. Rubin, and R. Bhatti, "Enforcing minimum necessary access in healthcare through integrated audit and access control," in *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*. ACM, 2013, p. 946.

[3] J. Carrigan, P. D. Martin, and M. Rushanan, "Kbid: Kerberos bracelet identification," 2016.

[4] P. D. Martin, M. Rushanan, M. Green, S. Checkoway, and A. D. Rubin, "Classifying network protocol implementation versions: An openssl case study," 2013.

[5] P. D. Martin, M. Rushanan, T. Tantillo, and A. D. Rubin, "Beacon+: Applications of short-range authenticated unidirectional advertisements," 2016.

BIBLIOGRAPHY

[6] U. D. of Health and H. Services, "Health insurance portability and accountability act of 1996," *Public Law*, vol. 104, p. 191, 1996.

[7] O. of the National Coordinator for Health IT, "Strategic health it advanced research projects on security."

[8] N. S. Foundation, "Trustworthy health and wellness."

[9] W. Zhang, Y. Chen, T. Cybulski, D. Fabbri, C. Gunter, P. Lawlor, D. Liebovitz, and B. Malin, "Decide now or decide later?: Quantifying the tradeoff between prospective and retrospective access decisions," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1182–1192.

[10] Y. Wang, S. W. Smith, and A. Gettinger, "Access control hygiene and the empathy gap in medical it," in *Proceedings of the USENIX Workshop on Health Security and Privacy*, 2012.

[11] C. A. Gunter, D. Liebovitz, and B. Malin, "Experience-based access management: A life-cycle framework for identity and access management systems," *IEEE security & privacy*, vol. 9, no. 5, p. 48, 2011.

[12] P. E. Lam, J. C. Mitchell, A. Scedrov, S. Sundaram, and F. Wang, "Declarative privacy policy: finite models and attribute-based encryption," in *Proceedings*

of the 2nd ACM SIGHIT International Health Informatics Symposium. ACM, 2012, pp. 323–332.

[13] X. H. Le and D. Wang, "Development of a system framework for implementation of an enhanced role-based access control model to support collaborative processes," in *Proc 3rd USENIX Workshops on Health Security and Privacy*, 2012.

[14] M. Johnson and S. M. Bellovin, "Policy management for e-health records," *Policy*, vol. 3, p. 09, 1910.

[15] A. Nadas, M. E. Frisse, and J. Sztipanovits, "Modeling privacy aware health information exchange systems," in *University of Amsterdam, Amsterdam Privacy Conference*, 2012.

[16] J. A. Cooley and S. W. Smith, "Dr. jekyll or mr. hyde: Information security in the ecosystem of healthcare."

[17] W. Zhang, Y. Chen, C. Gunter, D. Liebovitz, and B. Malin, "Evolving role definitions through permission invocation patterns," in *Proceedings of the 18th ACM symposium on Access control models and technologies*. ACM, 2013, pp. 37–48.

[18] W. Zhang, C. A. Gunter, D. Liebovitz, J. Tian, and B. Malin, "Role prediction using electronic medical record system audits," in *AMIA Annual Symposium*

Proceedings, vol. 2011. American Medical Informatics Association, 2011, p. 858.

[19] X. Lu, "Diagnosis based specialist identification in the hospital," 2014.

[20] A. Nadas, T. Levendovszky, E. K. Jackson, I. Madari, and J. Sztipanovits, "A model-integrated authoring environment for privacy policies," *Science of Computer Programming*, vol. 89, pp. 105–125, 2014.

[21] O. Chowdhury, H. Chen, J. Niu, N. Li, and E. Bertino, "On xacmls adequacy to specify and to enforce hipaa," in *USENIX Workshop on Health Security and Privacy*, 2012.

[22] A. Nadas, L. Juracz, J. Sztipanovits, M. E. Frisse, and A. J. Olsen, "Policyforge: A collaborative environment for formalizing privacy policies in health care," in *Proceedings of the 5th International Workshop on Software Engineering in Health Care*. IEEE Press, 2013, pp. 20–23.

[23] E. Duffy, S. Nyemba, C. A. Gunter, D. Liebovitz, and B. Malin, "Requirements and design for an extensible toolkit for analyzing emr audit logs," in *USENIX Workshop on Health Information Technologies*, 2013.

[24] E. L. DUFFY, "Facilitating patient and administrator analyses of electronic health record accesses," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2013.

BIBLIOGRAPHY

[25] H. Zhang, S. Mehotra, D. Liebovitz, C. A. Gunter, and B. Malin, "Mining deviations from patient care pathways via electronic medical record system audits," *ACM Transactions on Management Information Systems (TMIS)*, vol. 4, no. 4, p. 17, 2013.

[26] S. Gupta, C. Hanson, C. Gunter, M. Frank, D. Liebovitz, B. Malin *et al.*, "Modeling and detecting anomalous topic access," in *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on.* IEEE, 2013, pp. 100–105.

[27] P. Jindal, C. A. Gunter, and D. Roth, "Detecting privacy-sensitive events in medical text," in *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics.* ACM, 2014, pp. 617–620.

[28] P. Jindal and D. Roth, "End-to-end coreference resolution for clinical narratives," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence.* AAAI Press, 2013, pp. 2106–2112.

[29] ——, "Using soft constraints in joint inference for clinical concept recognition." in *EMNLP*, 2013, pp. 1808–1814.

[30] J. L. Hall, B. Callan, and H. Nissenbaum, "Accountings of relationships," in *Proceedings of the 3rd USENIX conference on Health Security and Privacy.* USENIX Association, 2012, pp. 15–15.

BIBLIOGRAPHY

[31] J. Blocki, N. Christin, A. Datta, and A. Sinha, "Audit mechanisms for privacy protection in healthcare environments," in *Proceedings of the 2nd USENIX conference on Health security and privacy*. USENIX Association, 2011, pp. 10–10.

[32] O. Chowdhury, L. Jia, D. Garg, and A. Datta, "Temporal mode-checking for runtime monitoring of privacy policies," in *Computer Aided Verification*. Springer, 2014, pp. 131–149.

[33] D. Garg, L. Jia, and A. Datta, "Policy auditing over incomplete logs: theory, implementation and applications," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 151–162.

[34] M. C. Tschantz, A. Datta, and J. M. Wing, "Purpose restrictions on information use," in *Computer Security–ESORICS 2013*. Springer, 2013, pp. 610–627.

[35] ——, "Formalizing and enforcing purpose restrictions in privacy policies," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 176–190.

[36] J. Blocki, N. Christin, A. Datta, A. D. Procaccia, and A. Sinha, "Audit games," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 41–47.

[37] J. Blocki, N. Christin, A. Datta, and A. Sinha, "Regret minimizing audits: A

learning-theoretic basis for privacy protection," in *Computer Security Foundations Symposium (CSF), 2011 IEEE 24th.* IEEE, 2011, pp. 312–327.

[38] ——, "Adaptive regret minimization in bounded-memory games," in *Decision and Game Theory for Security.* Springer, 2013, pp. 65–84.

[39] ——, "Audit mechanisms for provable risk management and accountable data governance," in *Decision and Game Theory for Security.* Springer, 2012, pp. 38–59.

[40] B. Adida, I. S. Kohane, and K. D. Mandl, "Practical health information exchange using a personally controlled health record." in *HealthSec*, 2010.

[41] M. Frank, B. Dong, A. P. Felt, and D. Song, "Mining permission request patterns from android and facebook applications," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on.* IEEE, 2012, pp. 870–875.

[42] C. Gunter, M. Berry, and M. French, "Decision support for data segmentation (ds2): Application to pull architectures for hie," in *Proceedings of the 2014 USENIX Summit on Health Information Technologies.* USENIX Association, 2014.

[43] E. M. Chan, P. E. Lam, and J. C. Mitchell, "Understanding the challenges with medical data segmentation for privacy," *Health Tech*, vol. 13, 2013.

[44] Y. Tang and L. Liu, "Searching hie with differentiated privacy preservation," in

Proceedings of the 2014 USENIX Summit on Health Information Technologies. USENIX Association, 2014.

[45] M. Chase and K. Lauter, "An anonymous health care system." *IACR Cryptology ePrint Archive*, vol. 2011, p. 16, 2011.

[46] S. E. Oh, J. Y. Chun, L. Jia, D. Garg, C. A. Gunter, and A. Datta, "Privacy-preserving audit for broker-based health information exchange," in *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM, 2014, pp. 313–320.

[47] J. A. Akinyele, C. U. Lehmann, M. D. Green, M. W. Pagano, Z. N. Peterson, and A. D. Rubin, "Self-protecting electronic medical records using attribute-based encryption," 2010.

[48] Q. Zhu, C. Gunter, and T. Basar, "Tragedy of anticommons in digital right management of medical records," in *Proceedings of the 3rd USENIX conference on Health Security and Privacy*. USENIX Association, 2012, pp. 10–10.

[49] T. Grandison and M. Kantarcioglu, "A risk management framework for health care data anonymization."

[50] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: efficient and secure testing of fully-sequenced human genomes," in

Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 691–702.

[51] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy in the genomic era," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 6, 2015.

[52] J. Wagner and H. Corrada-Bravo, "Privacy-preserving microbiome sequencing analysis and storage systems," in *Proceedings of the 2014 USENIX Summit on Health Information Technologies*. USENIX Association, 2014.

[53] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*. Springer, 2010, pp. 49–64.

[54] E. Ayday, J. L. Raisaro, M. Laren, P. Jack, J. Fellay, and J.-P. Hubaux, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," in *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech" 13)*, no. EPFL-CONF-187118, 2013.

[55] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 5, pp. 606–617, 2008.

[56] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic

computation," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 216–230.

[57] C. Cornelius, J. Sorber, R. Peterson, J. Skinner, R. Halter, and D. Kotz, "Who wears me? bioimpedance as a passive biometric," in *Proc. 3rd USENIX Workshop on Health Security and Privacy*, 2012.

[58] C. Cornelius, Z. Marois, J. Sorber, R. Peterson, S. Mare, and D. Kotz, "Vocal resonance as a passive biometric," 2014.

[59] S. Chang, Y. Hu, H. Anderson, T. Fu, and E. Y. L. Huang, "Body area network security: Robust key establishment using human body channel," in *Proc. 3rd USENIX Workshop on Health Security and Privacy (HealthSec)*, Aug. [Online]. Available: https://www.usenix.org/conference/healthsec12/workshop-program/presentation/Chang

[60] C. Cornelius and D. Kotz, "On usable authentication for wireless body area networks," in *Proceedings of the First USENIX Workshop on Health Security and Privacy (HealthSec)*, 2010.

[61] C. T. Cornelius and D. F. Kotz, "Recognizing whether sensors are on the same body," *Pervasive and Mobile Computing*, vol. 8, no. 6, pp. 822–836, 2012.

[62] S. Mare, A. Markham, C. Cornelius, R. Peterson, and D. Kotz, "Zebra: Zero-

effort bilateral recurring authentication," in *Security and Privacy (SP), 2014 IEEE Symposium on*, May 2014.

[63] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson, "Sok: Security and privacy in implantable medical devices and body area networks," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 524–539.

[64] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, K. Fu, and D. Song, "Take two software updates and see me in the morning: The case for software security evaluations of medical devices," in *Proceedings of the 2nd USENIX Workshop on Health Security and Privacy (HealthSec)*, 2011, pp. 1–5.

[65] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, K. Fu, and W. Xu, "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices," in *Proceedings of USENIX Workshop on Health Information Technologies*, vol. 2013, 2013.

[66] C. f. D. a. R. Health, "Safety communications - cybersecurity for medical devices and hospital networks: FDA safety communication." [Online]. Available: http://www.fda.gov/medicaldevices/safety/alertsandnotices/ucm356423.htm

[67] N. Paul, T. Kohno, and D. C. Klonoff, "A review of the security of insulin pump infusion systems," *Journal of diabetes science and technology*, vol. 5, no. 6, pp. 1557–1562, 2011.

BIBLIOGRAPHY

[68] J. Radcliffe, "Hacking medical devices for fun and insulin: Breaking the human scada system," in *Black Hat Conference presentation slides*, vol. 2011, 2011.

[69] G. J. Annas, "Hipaa regulations—a new era of medical-record privacy?" *New England Journal of Medicine*, vol. 348, no. 15, pp. 1486–1490, 2003.

[70] D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (rbac): Features and motivations," in *Proceedings of 11th Annual Computer Security Application Conference*. sn, 1995, pp. 241–48.

[71] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[72] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[73] P. Loscocco and S. D. Smalley, "Meeting critical security objectives with security-enhanced linux," in *Proceedings of the 2001 Ottawa Linux symposium*, 2001, pp. 115–134.

[74] X. Zhang, S. Oh, and R. Sandhu, "Pbdm: a flexible delegation model in rbac," in *Proceedings of the eighth ACM symposium on Access control models and technologies*. ACM, 2003, pp. 149–157.

[75] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring role-based access control

to enforce mandatory and discretionary access control policies," *ACM Transactions on Information System Security*, vol. 3, no. 2, pp. 85–106, May 2000.

[76] C. Gunter, D. Liebovitz, and B. Malin, "Experience-based access management: A life-cycle framework for identity and access management systems," *IEEE Security and Privacy*, vol. 9, no. 5, pp. 48–55, 2011.

[77] Y. B. Choi, K. E. Capitan, J. S. Krause, and M. M. Streeper, "Challenges associated with privacy in health care industry: implementation of hipaa and the security rules," *Journal of Medical Systems*, vol. 30, no. 1, pp. 57–64, 2006.

[78] M. Kapushion, "Hungry, hungry hipaa: When privacy regulations go too far," *Fordham Urb. LJ*, vol. 31, p. 1483, 2003.

[79] R. Bhatti and T. Grandison, "Towards improved privacy policy coverage in healthcare using policy refinement," in *Proceedings of the 4th VLDB conference on Secure data management*. Springer-Verlag, 2007, pp. 158–173.

[80] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini, "Audit-based compliance control," *International Journal of Information Security*, vol. 6, no. 23, pp. 133–151, 2007.

[81] D. Garg, L. Jia, and A. Datta, "Policy auditing over incomplete logs: theory, implementation and applications," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 151–162.

BIBLIOGRAPHY

[82] E. Afgan and P. Bangalore, "Embarrassingly parallel jobs are not embarrassingly easy to schedule on the grid," in *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on*, 2008, pp. 1–10.

[83] P. D. Smith, "Implementing an emr system: one clinic's experience," *Family practice management*, vol. 10, no. 5, pp. 37–52, 2003.

[84] S. Rehm and S. Kraft, "Electronic medical records: the fpm vendor survey," *Family Practice Management*, 2001.

[85] D. Horrocks, "Crisp: an introduction." *Maryland medicine: MM: a publication of MEDCHI, the Maryland State Medical Society*, vol. 11, no. 3, p. 20, 2010.

[86] T. G. Mattson, B. A. Sanders, and B. L. Massingill, *Patterns for parallel programming*. Addison-Wesley Professional, 2004.

[87] M. J. Flynn, "Some computer organizations and their effectiveness," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 948–960, 1972.

[88] R. Duncan, "A survey of parallel computer architectures," *Computer*, vol. 23, no. 2, pp. 5–16, 1990.

[89] E. E. Johnson, "Completing an mimd multiprocessor taxonomy," *ACM SIGARCH Computer Architecture News*, vol. 16, no. 3, pp. 44–47, Jun. 1988.

[90] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, vol. 11, p. 21, 2007.

BIBLIOGRAPHY

[91] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[92] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *IEEE 13th International Symposium on High Performance Computer Architecture*. IEEE, 2007, pp. 13–24.

[93] M. Donner, D. Nochin, D. Shasha, and W. Walasek, "Algorithms and experience in increasing the intelligibility and hygiene of access control in large organizations," in *Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security: Data and Application Security, Development and Directions*, 2001, pp. 195–316.

[94] A. Ferreira, R. Cruz-Correia, L. Antunes, and D. Chadwick, "Access control: how can it improve patients' healthcare?" *Medical and Care Compunetics Four*, vol. 4, p. 65, 2007.

[95] A. A. Boxwala, J. Kim, J. M. Grillo, and L. Ohno-Machado, "Using statistical and machine learning to help institutions detect suspicious access to electronic health records," *Journal of the American Medical Informatics Association*, vol. 18, no. 4, pp. 498–505, 2011.

[96] D. Fabbri and K. LeFevre, "Explaining accesses to electronic medical records

using diagnosis information," *Journal of the American Medical Informatics Association*, vol. 20, no. 1, pp. 52–60, 2013.

[97] Alexa The Web Information Company, "Alexa top sites," http://www.alexa.com/topsites, 2013.

[98] Orange, "Orange website," http://orange.biolab.si/, 2013.

[99] iSECpartners, "iSEC partners releases SSLyze," https://www.isecpartners.com/tools/application-security/isec-partners-releases-sslyze.aspx, 2013.

[100] iSEC Partners, "Details on the "Crime" attack," https://www.isecpartners.com/news-events/news/2012/september/details-on-the-crime-attack.aspx, 2012.

[101] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," http://www.ietf.org/rfc/rfc5246.txt, 2008.

[102] Google, "Official Gmail blog: Default https access for Gmail," http://gmailblog.blogspot.com/2010/01/default-https-access-for-gmail.html, 2010.

[103] Facebook, "The Facebook blog: A continued commitment to security," https://www.facebook.com/blog/blog.php?post=486790652130, 2011.

[104] @twitter, "Securing your Twitter experience with HTTPS," http://blog.twitter.com/2012/02/securing-your-twitter-experience-with.html, 2012.

[105] L. Constantin, "Most of the Internet's top 200,000 HTTPS websites are insecure, trustworthy internet movement says," *CIO Magazine*, 2012.

[106] Netcraft, "Netcraft: February 2013 web server survey," http://news.netcraft.com/ssl-survey/, 2013.

[107] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When private keys are public: results from the 2008 debian openssl vulnerability," in *Proceedings of IMC 2009*. ACM Press, 2009. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644896

[108] N. J. AlFardan and K. G. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS record protocols," in *Proceedings of IEEE Symposium on Security and Privacy ("Oakland") 2013*. IEEE Computer Society, 2013.

[109] OpenSSL Project, "OpenSSL security advisory," 2007. [Online]. Available: http://www.openssl.org/news/secadv_20071012.txt

[110] D. Goodin, "Hackers break SSL encryption used by millions of sites," http://www.theregister.co.uk/2011/09/19/beast_exploits_paypal_ssl/, 2011.

[111] Trustworth Internet Movement, "SSL Pulse: Survey of the SSL implementation of the most popular web sites," https://www.trustworthyinternet.org/ssl-pulse/, 2013.

[112] G. V. Bard, "A challenging but feasible blockwise-adaptive chosen-plaintext

attack on SSL," in *Proceedings of SECRYPT 2006*. INSTICC Press, 2006, pp. 7–10.

[113] B. Canvel, A. P. Hiltgen, S. Vaudenay, and M. Vuagnoux, "Password interception in a SSL/TLS channel," in *Proceedings of CRYPTO 2003*. Springer, 2003, pp. 583–599.

[114] N. J. AlFardan and K. G. Paterson, "Plaintext-recovery attacks against datagram TLS," in *Proceedings of NDSS 2012*, 2012.

[115] J. Kelsey, "Compression and information leakage of plaintext," in *Proceedings of FSE 2002*. Springer, 2002, pp. 263–276. [Online]. Available: http://www.iacr.org/cryptodb/archive/2002/FSE/3091/3091.pdf

[116] OpenSSL Project, "Openssl vulnerabilities," http://www.openssl.org/news/vulnerabilities.html, 2013.

[117] US-CERT/NIST, "OpenSSL CVE-2010-3864," http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-3864, 2010.

[118] ——, "OpenSSL CVE-2007-5135," http://www.cvedetails.com/cve/CVE-2007-5135/, 2007.

[119] ——, "OpenSSL CVE-2007-4995," http://www.cvedetails.com/cve/CVE-2007-4995/, 2007.

BIBLIOGRAPHY

[120] OpenSSL Project, "OpenSSL: Cryptography and SSL/TLS Toolkit," http://www.openssl.org/, 2013.

[121] Electronic Frontier Foundation, "The EFF SSL Observatory," https://www.eff.org/observatory, 2010.

[122] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your Ps and Qs: Detection of widespread weak keys in network devices," in *Proceedings of USENIX Security 2012*. USENIX, 2012.

[123] T. Saltman, "Fingerprinting SSL tutorial," http://travisaltman.com/fingerprinting-ssl-tutorial/, 2013.

[124] McAfee, "SSLDigger McAfee v1.02," http://www.mcafee.com/us/downloads/free-tools/ssldigger.aspx, 2013.

[125] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: A thorough analysis of the x.509 PKI using active and passive measurements," in *Proceedings of IMC 2011*. ACM Press, 2011. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068856

[126] T. Zoller, "SSL Audit - TLS/SSL Scanner," http://www.g-sec.lu/sslaudit/documentation.pdf, 2013.

[127] Symantec, "Solution overview: Industry perspectives - healthcare embedded medical device security and privacy," dec 2010.

[128] Lantronix, "Medical device networking for smarter healthcare - real world integration ? applications, integration issues and benefits," dec 2010.

[129] S. R. Rakitin, "Networked medical devices: Essential collaboration for improved safety," *MANAGEMENT & TECHNOLOGY*, pp. 332–338, jul 2009.

[130] H. Baldus, K. Klabunde, and G. Muesch, "Reliable set-up of medical body-sensor networks," in *Wireless Sensor Networks*. Springer, 2004, pp. 353–363.

[131] R. Dantu, H. Oosterwijk, P. Kolan, and H. Husna, "Securing medical networks," *Network Security*, vol. 2007, no. 6, pp. 13–16, 2007.

[132] C. Doukas, I. Maglogiannis, and G. Kormentzas, "Advanced telemedicine services through context-aware medical networks," in *International IEEE EMBS special topic conference on information technology applications in biomedicine*, 2006.

[133] C. Doukas and I. Maglogiannis, "Adaptive transmission of medical image and video using scalable coding and context-aware wireless medical networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, p. 25, 2008.

[134] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. Ko, J. H. Lim, A. Terzis, A. Watt, J. Jeng, B.-r. Chen *et al.*, "Wireless medical sensor networks in emergency

response: Implementation and pilot results," in *Technologies for Homeland Security, 2008 IEEE Conference on*. IEEE, 2008, pp. 187–192.

[135] "Midas technical handbook." [Online]. Available: http://microwatt.com/wp-content/uploads/2014/10/MIDASA001-Technical-Manual-ENG-rev18.pdf

[136] "Micrologix 1100 embedded web server." [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1763-um002_-en-p.pdf

[137] "Medical devices contain hard-coded passwords, ICS-CERT warns." [Online]. Available: http://threatpost.com/medical-devices-contain-hard-coded-passwords-ics-cert-warns/100994

[138] ICS-CERT, "Icsa-15-309-02." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-309-02

[139] ——, "Icsa-15-300-03." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-300-03

[140] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *IEEE Symposium on Security and Privacy, 2008. SP 2008*, May 2008, pp. 129–142.

[141] C. Li, A. Raghunathan, and N. Jha, "Hijacking an insulin pump: Security

attacks and defenses for a diabetes therapy system," in *2011 13th IEEE International Conference on e-Health Networking Applications and Services (Healthcom)*, Jun. 2011, pp. 150–156.

[142] Alaris, "Alaris medical systems ivac model 710x 720x series signature edition volumetric pump technical service manual," 1997.

[143] D. M. Zuckerman, P. Brown, and S. E. Nissen, "Medical device recalls and the fda approval process," *Archives of internal medicine*, vol. 171, no. 11, pp. 1006–1011, 2011.

[144] E. Snell, "Fda releases medical device cybersecurity warning," 2015.

[145] D. Arora, S. Ravi, A. Raghunathan, and N. Jha, "Hardware-assisted run-time monitoring for secure program execution on embedded processors," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 12, pp. 1295–1308, Dec 2006.

[146] H. Lee, H. Moon, D. Jang, K. Kim, J. Lee, Y. Paek, and B. B. Kang, "Ki-mon: A hardware-assisted event-triggered monitoring platform for mutable kernel object," in *Proceedings of the 22Nd USENIX Conference on Security*, ser. SEC'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 511–526. [Online]. Available: http://dl.acm.org/citation.cfm?id=2534766.2534810

[147] A. Benso, S. Di Carlo, G. Di Natale, and P. Prinetto, "A watchdog processor

to detect data and control flow errors," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*. IEEE, 2003, pp. 144–148.

[148] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.

[149] Z. Shao, Q. Zhuge, Y. He, and E.-M. Sha, "Defending embedded systems against buffer overflow via hardware/software," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 352–361.

[150] Z. Shao, C. Xue, Q. Zhuge, M. Qiu, B. Xiao, and E.-M. Sha, "Security protection and checking for embedded system integration against buffer overflow attacks via hardware/software," *Computers, IEEE Transactions on*, vol. 55, no. 4, pp. 443–453, 2006.

[151] A. Francillon, D. Perito, and C. Castelluccia, "Defending embedded systems against control flow attacks," in *Proceedings of the first ACM workshop on Secure execution of untrusted code*. ACM, 2009, pp. 19–26.

[152] R. Riley, X. Jiang, and D. Xu, "An architectural approach to preventing code injection attacks," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 351–365, 2010.

[153] S. Lukovic, P. Pezzino, and L. Fiorin, "Stack protection unit as a step towards

securing mpsocs," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on.* IEEE, 2010, pp. 1–4.

[154] F. Wolff, C. Papachristou, D. Weyer, and W. Clay, "Embedded system protection from software corruption," in *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on.* IEEE, 2010, pp. 223–229.

[155] L. Davi, A.-R. Sadeghi, and M. Winandy, "Ropdefender: A detection tool to defend against return-oriented programming attacks," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security.* ACM, 2011, pp. 40–51.

[156] D. Weber, "Optiguard: A smart meter assessment toolkit," *Black Hat USA*, 2012.

[157] ICS-CERT, "Icsa-10-214-01." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-10-214-01

[158] ——, "Icsa-15-300-02." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-300-02

[159] ——, "Icsa-11-216-01." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-11-216-01

[160] ——, "Icsa-12-030-01a." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-12-030-01A

[161] ——, "Icsa-12-018-01b." [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-12-018-01B

[162] N. L. Petroni, Jr., T. Fraser, J. Molina, and W. A. Arbaugh, "Copilot - a coprocessor-based kernel runtime integrity monitor," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 13–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251375.1251388

[163] H. Moon, H. Lee, J. Lee, K. Kim, Y. Paek, and B. B. Kang, "Vigilare: Toward snoop-based kernel integrity monitor," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 28–37. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382202

[164] K. Flanagan, "Bach: Byu address collection hardware, the collection of complete traces," 1992.

[165] Z. Liu, J. Lee, J. Zeng, Y. Wen, Z. Lin, and W. Shi, "Cpu transparent protection of os kernel and hypervisor integrity with programmable dram," in *Proceedings of the 40th Annual International Symposium on Computer*

Architecture, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 392–403. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485956

[166] F. Vahid and T. Givargis, *Embedded system design: a unified hardware/software introduction*. John Wiley & Sons New York, NY, 2002, vol. 4.

[167] W. Stallings, *Computer organization and architecture: designing for performance*. Pearson Education India, 2000.

[168] P. M. Kogge, *The architecture of pipelined computers*. CRC Press, 1981.

[169] J. E. Smith, "A study of branch prediction strategies," in *Proceedings of the 8th annual symposium on Computer Architecture*. IEEE Computer Society Press, 1981, pp. 135–148.

[170] K. Driesen and U. Hölzle, "The cascaded predictor: Economical and adaptive branch target prediction," in *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society Press, 1998, pp. 249–258.

[171] D. A. Patterson and J. L. Hennessy, *Computer organization and design: the hardware/software interface*. Newnes, 2013.

[172] A. One, "Smashing the stack for fun and profit," *Phrack magazine*, vol. 7, no. 49, pp. 14–16, 1996.

BIBLIOGRAPHY

[173] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 552–561.

[174] J. J. Carr, *Z80 User's Manual*. Prentice Hall PTR, 1980.

[175] I. Freescale Semiconductor. M68000 8-/16-/32-bit microprocessors user?s manual. Freescale Semiconductor, Inc. [Online]. Available: http://www.freescale.com/files/32bit/doc/ref_manual/MC68000UM.pdf

[176] D. Seal, *ARM architecture reference manual*. Pearson Education, 2001.

[177] Intel, "80c186xl/80c188xl 16-bit high-integration embedded processors," oct 1995.

[178] J. U. Skakkebæk, R. B. Jones, and D. L. Dill, "Formal verification of out-of-order execution using incremental flushing," in *Computer Aided Verification*. Springer, 1998, pp. 98–109.

[179] Intel, "8-mbit (512k x 16, 1024k x 8) smartvoltage boot block flash memory family," sept 1995.

[180] E. Carrera, "Introduction to idapython," 2005.

[181] Ekahu, "Asset tracking & management," 2015. [Online]. Available: http://www.ekahau.com/real-time-location-system/solutions/healthcare/asset-tracking-management

BIBLIOGRAPHY

[182] "iBeacon for Developers," https://developer.apple.com/ibeacon/, accessed: 2015-08-17.

[183] T. Labs, "Trusted beacon reference," Apr. 2015. [Online]. Available: http://docs.twocanoes.com/trusted_beacon/index.html

[184] S. Contini, "The factorization of rsa-140," *RSA Laboratories' Bulletin*, vol. 10, pp. 1–2, 1999.

[185] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik *et al.*, "Factorization of a 768-bit rsa modulus," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 333–350.

[186] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Proceedings of the 2Nd ACM Workshop on Wireless Security*, ser. WiSe '03. ACM, 2003, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/941311.941313

[187] E. Bertino and M. S. Kirkpatrick, "Location-based access control systems for mobile users: concepts and research directions." in *SPRINGL*. ACM, 2011, pp. 49–52. [Online]. Available: http://dblp.uni-trier.de/db/conf/gis/springl2011.html#BertinoK11

[188] M. Portnoi and C. Shen, "Location-aware sign-on and key exchange using

attribute-based encryption and bluetooth beacons," in *IEEE Conference on Communications and Network Security*, 2013, pp. 405–406. [Online]. Available: http://dx.doi.org/10.1109/CNS.2013.6682750

[189] R. RFID, "Rfid asset tracking," 2015. [Online]. Available: http://radiantrfid.com/asset-tracking.html

[190] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: indoor location sensing using active rfid," *Wireless networks*, vol. 10, no. 6, pp. 701–710, 2004.

[191] I. LiveViewGPS, "Gps tracking - tracking systems - you can trust," 2015. [Online]. Available: http://www.liveviewgps.com

[192] J. Landt, "The history of rfid," *Potentials, IEEE*, vol. 24, no. 4, pp. 8–11, 2005.

[193] M. Bhuptani and S. Moradpour, *RFID field guide: deploying radio frequency identification systems*. Prentice Hall PTR, 2005.

[194] K. Finkenzeller, *RFID Handbook: Radio-frequency identification fundamentals and applications*. Wiley, 1999.

[195] B. W. Parkinson and S. W. Gilbert, "Navstar: global positioning systemten years later," *Proceedings of the IEEE*, vol. 71, no. 10, pp. 1177–1186, 1983.

[196] J. B.-Y. Tsui, *Fundamentals of global positioning system receivers*. Wiley-Interscience, 2000.

BIBLIOGRAPHY

[197] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition.* Lincoln, MA: Ganga-Jamuna Press, 2006.

[198] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, "Ieee 802.11 wireless local area networks," *Communications Magazine, IEEE*, vol. 35, no. 9, pp. 116–126, 1997.

[199] P. Brenner, "A technical tutorial on the ieee 802.11 protocol," *BreezeCom Wireless Communications*, pp. 1–24, 1997.

[200] H. L. Moen and T. Jelle, "The potential for location-based services with wi-fi rfid tags in citywide wireless networks," in *Wireless Communication Systems, 2007. ISWCS 2007. 4th International Symposium on.* IEEE, 2007, pp. 148–152.

[201] F. Schrooyen, I. Baert, S. Truijen, L. Pieters, T. Denis, K. Williame, and M. Weyn, "Real time location system over wifi in a healthcare environment," *Journal on Information Technology in Healthcare*, vol. 4, no. 6, pp. 401–416, 2006.

[202] R. Want, "Near field communication," *IEEE Pervasive Computing*, no. 3, pp. 4–7, 2011.

[203] B. Specification, "Version 1.1," *Includes: IMS Learning Resource Meta-data Information Model IMS Learning Resource Meta-data XML Binding Specification*

IMS Learning Resource Meta-data Best Practice and Implementation Guide Available at: www. imsproject. org, 2001.

[204] J. C. Haartsen, "The bluetooth radio system," *Personal Communications, IEEE*, vol. 7, no. 1, pp. 28–36, 2000.

[205] P. Bhagwat, "Bluetooth: technology for short-range wireless apps," *Internet Computing, IEEE*, vol. 5, no. 3, pp. 96–103, 2001.

[206] S. Feldmann, K. Kyamakya, A. Zapater, and Z. Lue, "An indoor bluetooth-based positioning system: Concept, implementation and experimental evaluation." in *International Conference on Wireless Networks*, 2003, pp. 109–113.

[207] R. Bruno and F. Delmastro, "Design and analysis of a bluetooth-based indoor localization system," in *Personal wireless communications*. Springer, 2003, pp. 711–725.

[208] S. Hay and R. Harle, "Bluetooth tracking without discoverability," in *Location and context awareness*. Springer, 2009, pp. 120–137.

[209] S. K. Opoku, "An indoor tracking system based on bluetooth technology," *arXiv preprint arXiv:1209.3053*, 2012.

[210] Teldio, "Indoor positioning system: Product overview," 2015. [Online]. Available: http://media.teldio.com/collateral/product_collateral/Teldio-IPS-Brochure.pdf

[211] M. Honkanen, A. Lappeteläinen, and K. Kivekäs, "Low end extension for bluetooth," in *Radio and Wireless Conference, 2004 IEEE.* IEEE, 2004, pp. 199–202.

[212] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[213] B. Yu, L. Xu, and Y. Li, "Bluetooth low energy (ble) based mobile electrocardiogram monitoring system," in *Information and Automation (ICIA), 2012 International Conference on.* IEEE, 2012, pp. 763–767.

[214] Gimbal, "The gimbal store," Aug. 2015. [Online]. Available: https://store.gimbal.com

[215] Estimote, "Estimote: Real-world context for your apps," Aug. 2015. [Online]. Available: http://estimote.com/#jump-to-products

[216] A. Developer, "Getting started with ibeacon," 2014.

[217] J. Yang, Z. Wang, and X. Zhang, "An ibeacon-based indoor positioning systems for hospitals," 2015.

[218] Z. Chen, Q. Zhu, H. Jiang, H. Zou, Y. C. Soh, L. Xie, R. Jia, and C. Spanos, "An ibeacon assisted indoor localization and tracking system."

[219] S. A. Ortiz and L. M. Ortiz, "Systems and methods for tracking assets using associated portable electronic device in the form of beacons," Mar. 2014, uS Patent App. 14/194,953.

[220] M. Kouhne and J. Sieck, "Location-based services with ibeacon technology," in *Artificial Intelligence, Modelling and Simulation (AIMS), 2014 2nd International Conference on*. IEEE, 2014, pp. 315–321.

[221] M. Lu, W. Chen, X. Shen, H.-C. Lam, and J. Liu, "Positioning and tracking construction vehicles in highly dense urban areas and building construction sites," *Automation in Construction*, vol. 16, no. 5, pp. 647–656, 2007.

[222] K. C. Cheung, S. S. Intille, and K. Larson, "An inexpensive bluetooth-based indoor positioning hack," in *Proceedings of UbiComp*, vol. 6. Citeseer, 2006.

[223] Texas Instruments, "MSP430FR5969 launchpad development kit," Jul. 2015. [Online]. Available: http://www.ti.com/tool/MSP-EXP430FR5969

[224] Hardware Breakout, "Bluetooth low energy boosterpack for the launchpad," Aug. 2015. [Online]. Available: http://store.hardwarebreakout.com/index.php?route=product/product&product_id=65

[225] T. Bradley, "Pros and cons of bringing your own device to work," Dec. 2011. [Online]. Available: http://www.pcworld.com/article/246760/pros_and_cons_of_byod_bring_your_own_device_.html

[226] "Who wears me? bioimpedance as a passive biometric," in *Presented as part of the 3rd USENIX Workshop on Health Security and Privacy*. Berkeley, CA: USENIX, 2012. [Online]. Available: https://www.usenix.org/conference/healthsec12/workshop-program/presentation/Cornelius

[227] D. Manolakis, "Efficient solution and performance analysis of 3-d position estimation by trilateration," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 32, no. 4, pp. 1239–1248, Oct 1996.

[228] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *Robotics, IEEE Transactions on*, vol. 21, no. 1, pp. 93–101, Feb 2005.

[229] W. Murphy and W. Hereman, "Determination of a position in three dimensions using trilateration and approximate distances," *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95*, vol. 7, p. 19, 1995.

[230] "OpenMRS Wiki Rosources - Demo Data," https://wiki.openmrs.org/display/RES/Demo+Data, accessed: 2015-08-17.

[231] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security and Privacy*, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1109/MSP.2004.81

[232] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno, "Rfids and secret hand-

shakes: defending against ghost-and-leech attacks and unauthorized reads with context-aware communications," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 479–490.

[233] M. Ryan, "Bluetooth: With low energy comes low security," in *Proceedings of the 7th USENIX Conference on Offensive Technologies*. USENIX Association, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2534748.2534754

[234] A. Goode, "Bring your own finger–how mobile is bringing biometrics to consumers," *Biometric Technology Today*, vol. 2014, no. 5, pp. 5–9, 2014.

[235] B. Krzanich, "Intel developer forum san francisco opening keynote," Tech. Rep., 2015.

Vita

Paul D. Martin developed an interest in technology when he received his first computer at the age of ten. Since then, he has spent much of his time exploring this field. Initially a hobby, computer science quickly became a passion and central part of his life.

Paul received his B. S. and M. S. E. degrees in Computer Science from Johns Hopkins University in 2011 and 2013, respectively. He enrolled in the Computer Science Ph.D. program at Johns Hopkins University in 2011. He was inducted into the Upsilon Pi Epsilon International Computer Science Honor Society in 2013. His research interests include embedded systems security, operating system security, vulnerability analysis, reverse engineering, network protocol analysis, anomaly detection and big-data security analytics.