Mithören

by

Colin Buckles and Michael Wolf

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Colin Buckles and Michael Wolf — 04/17/17

Bogdan Vykhovanyuk, Faculty Advisor — 04/17/17

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

March 2017

Mithören

Wireless Keystroke Penetration Test Kit

Developed by
Colin Buckles and Michael Wolf
Students of the University of Cincinnati,
College of Education, Criminal Justice, and Human Services,
School of Information Technology
cech.uc.edu/it

March 2017

TABLE OF CONTENTS

TABLES

FIGURES

No. Page

Abstract

The development of wireless computer peripherals has been focused on design, affordability, and speed-to-market. Unfortunately, this has left the products without substantial security oversight. Poorly designed proprietary protocols atop a physical layer vulnerable to eavesdropping opens a large swath of wireless keyboards and mice in the wild to fingerprinting, keylogging, and replay attacks. Especially in enterprise environments with Bring Your Own Device (BYOD) policies, keyboards can be targeted for data exfiltration and password theft. Mithören has been developed to provide a simple-to-use, extensible platform for wireless packet collection and vulnerable device identification. Integrating existing tools, Mithören simplifies software-defined-radio (SDR) surveillance and development for researchers and penetration testers. By creating a wireless peripheral inspection suite, this addition to the security analyst's toolkit can cover another gap in analysis in the ever-more-connected world.

1. Problem Statement

1.1. Introduction

A recent whitepaper conducted by Ovum Consulting showed that nearly half of American businesses have a bring-your-own device (BYOD) policy, allowing for employees to supply the components with which they interface with company systems (1, p. 2). As wireless technology continues to expand into the home and enterprise environments, connecting systems that exchange sensitive material, the tradeoff of security must be taken into account. Wireless technology, fundamentally based on broadcast radio waves over the air, is vulnerable to eavesdropping, denial of service, and replay attacks. When products that fail to take into account the risks involved in this communication paradigm, they leave their customers open to data loss and digital compromise. One such vulnerable set of products are several keyboards and mice operating on the 2.4 Ghz band with several proprietary protocols. These devices generally speak to a Nordic Semiconductor radio receiver in the USB port of the user's computer, however researchers at Bastille research determined a way to intercept, repeat, and inject packets containing user keystrokes (2, p. 5). This flaw, referred to as Mousejack, leaves user input unsecured. While this particular attack has been partially patched by manufacturers, more measures need to be taken in order to further testing and research of wireless peripheral security.

1.2. Project Description

We will be designing and implementing a simple, daemonized application which will serve as a platform for developers, penetration testers, and researchers to analyze nearby wireless keyboards and integrate tools into an extensible command line interface. Informed by existing penetration testing tools, this project looks to create a proof-of-concept, plug-and-play tool for interfacing with Bastille's Mousejack and other vulnerabilities in the radio ecosystem.

1.3. Problem

The problem which this project seeks to solve is the difficulty of bootstrapping software-defined radio (SDR) and other radio-oriented tools to test the security of wireless peripherals. Traditionally, SDR equipment cost-prohibitive and the software used to program it is not trivial to understand. Even tools written for the Mousejack exploit require flashing firmware onto USB receivers. An extensible solution that is ready to be used out of the box in penetration tests is missing.

1.4. User Profile

This project is relevant to security researchers, enthusiasts, penetration testers, and those conducting internal audits. Users would be familiar with open source software, command line interfaces, and the basics of radio.

1.4.1. Security Researchers

Security researchers and developers of SDR-analysis tools would be able to integrate their tools easily to abstract testing and packet aggregation. Employees of hardware manufacturers would be able to utilize this suite to verify the security of their products before they are released to the public.

1.4.2. Penetration Testers

White-hat hackers and individuals testing the security of a network or business environment would benefit from an additional tool to test for devices transmitting packets susceptible to eavesdropping. This tool would be simple to use, effective for a decent distance, and potentially integrate into existing penetration testing tools and reports.

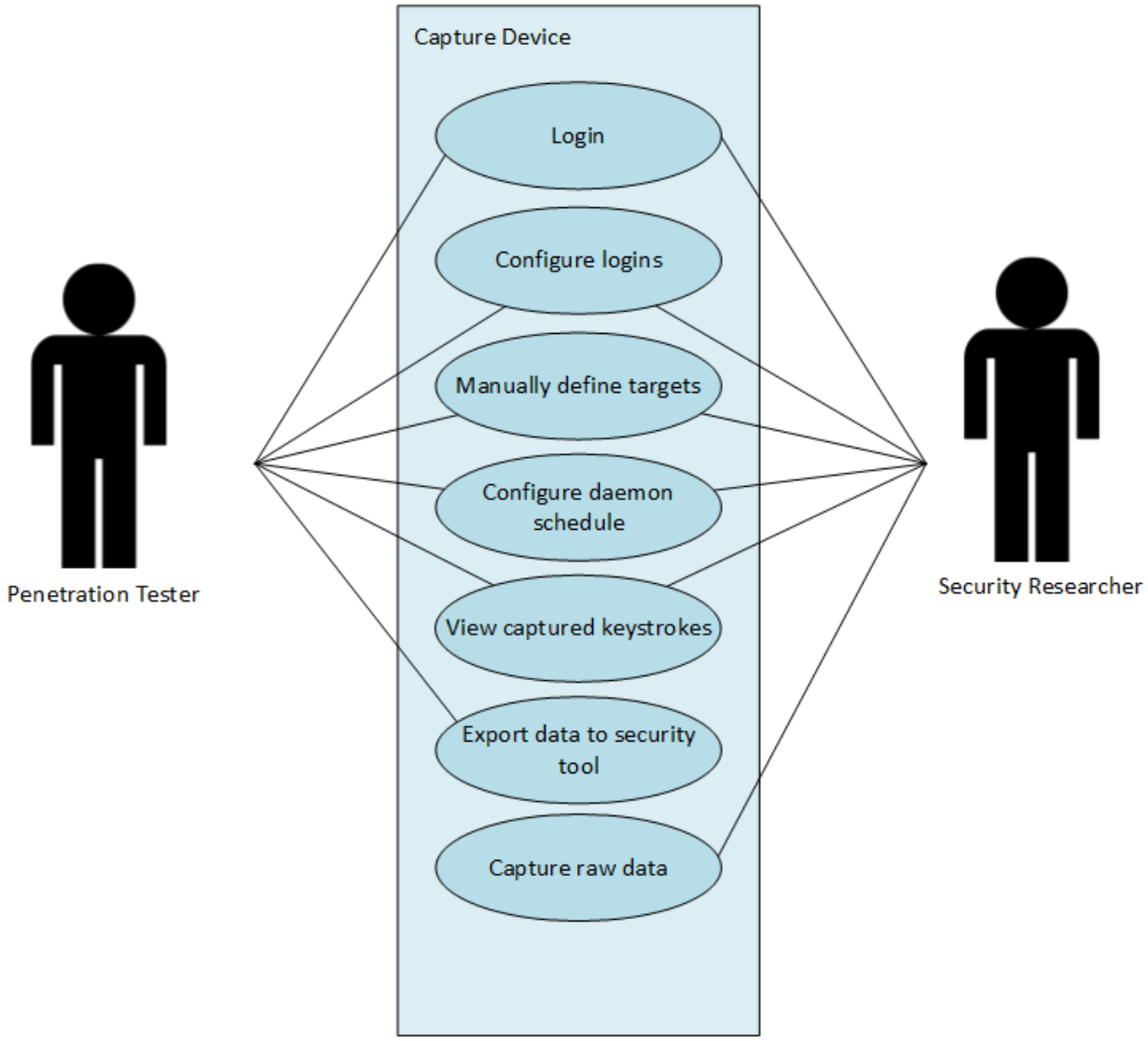
1.4.3. Internal Audit

Audit departments will be able to test to see if any rogue wireless peripherals are being used within the business. If the business is not Bring Your Own Device and wants its employees to only used business issued wired keyboard and mice, the tool would be able to sniff out and see if any employees are using wireless devices in the vicinity and breaking the business's policy. See *Figure 1* for details regarding the user profile and *Figure 2* for a projected use case diagram.

Figure 1 User Profile

Application: Unified wireless keyboard keylogging suite
Potential Users: Security Researchers and Developers Penetration Testers
Software and Interface Experience: Understanding of TCP/IP, radio, and wireless technology Experience with commandline, text-based applications
Experience with Similar Applications: Nmap, AirCrackNG, metasploit, responder.py Wired keyloggers
Task Experience: Real time feedback via text based output Understanding of keylogging
Frequency of Use: Several commands per minute Scheduled security scans
Key Interface Design Requirements that the Profile Suggests: Does not require extensive interface, but should be simple enough to run and analyze within ten minutes of installation/boot-up Should not require knowledge of radio or SDR in order to begin capturing keystrokes. Should provide feedback to user and log appropriately

Figure 2 Use Case Diagram



2. Project Management

2.1. Objectives/Deliverables

Table 1 presents the major project deliverables

Table 1 Project Objectives/Deliverables Due Dates

MAJOR PROJECT MILESTONES (DELIVERABLES)				
Pre Planning	**9/23/16**		Networking and Email Milestone	**01/30/17**
Pre Setup Milestone	**9/30/16**		Database Milestone	**02/20/17**
Hardware Configuration Milestone	**10/28/16**		Testing Milestone	**03/06/17**
Daemonization Milestone	**11/11/16**		Documentation Milestone	**03/20/17**
User Interface Milestone	**11/25/16**		Retrospective Milestone	**03/25/17**
Presentation Milestone	**11/30/16**		Final Presentation Milestone	**04/06/17**

2.2. Project Schedule

Figure 3 displays the estimated timeline of the project including both project work and product work over the course of the fall of 2016. *Figure 4* presents a similar representation for the spring of 2017. A more in-depth depiction of the project timeline is enumerated in *Table 2*

Figure 3 Fall 2016 Project Schedule Gantt Chart

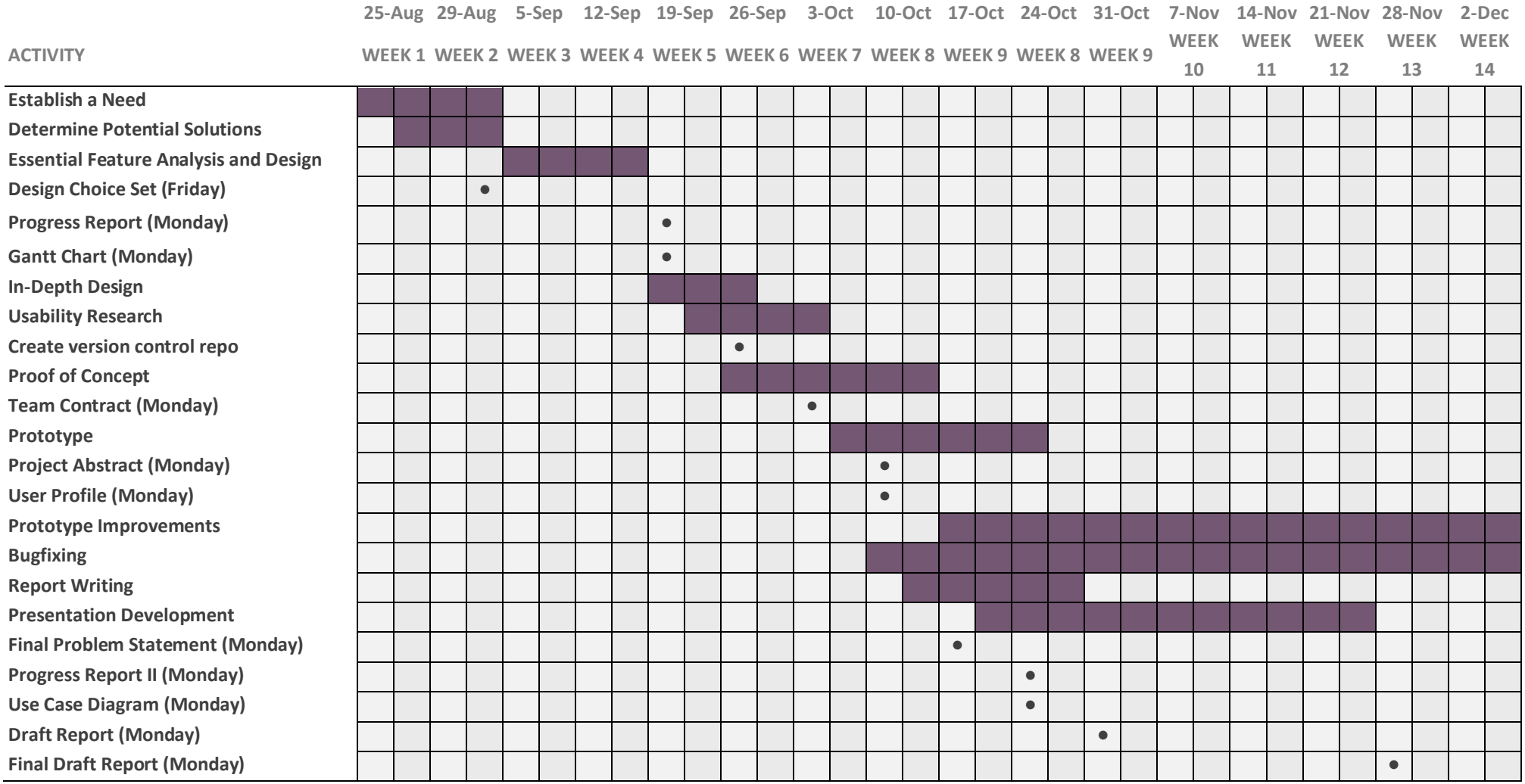


Figure 4 Spring 2017 Project Schedule Gantt Chart

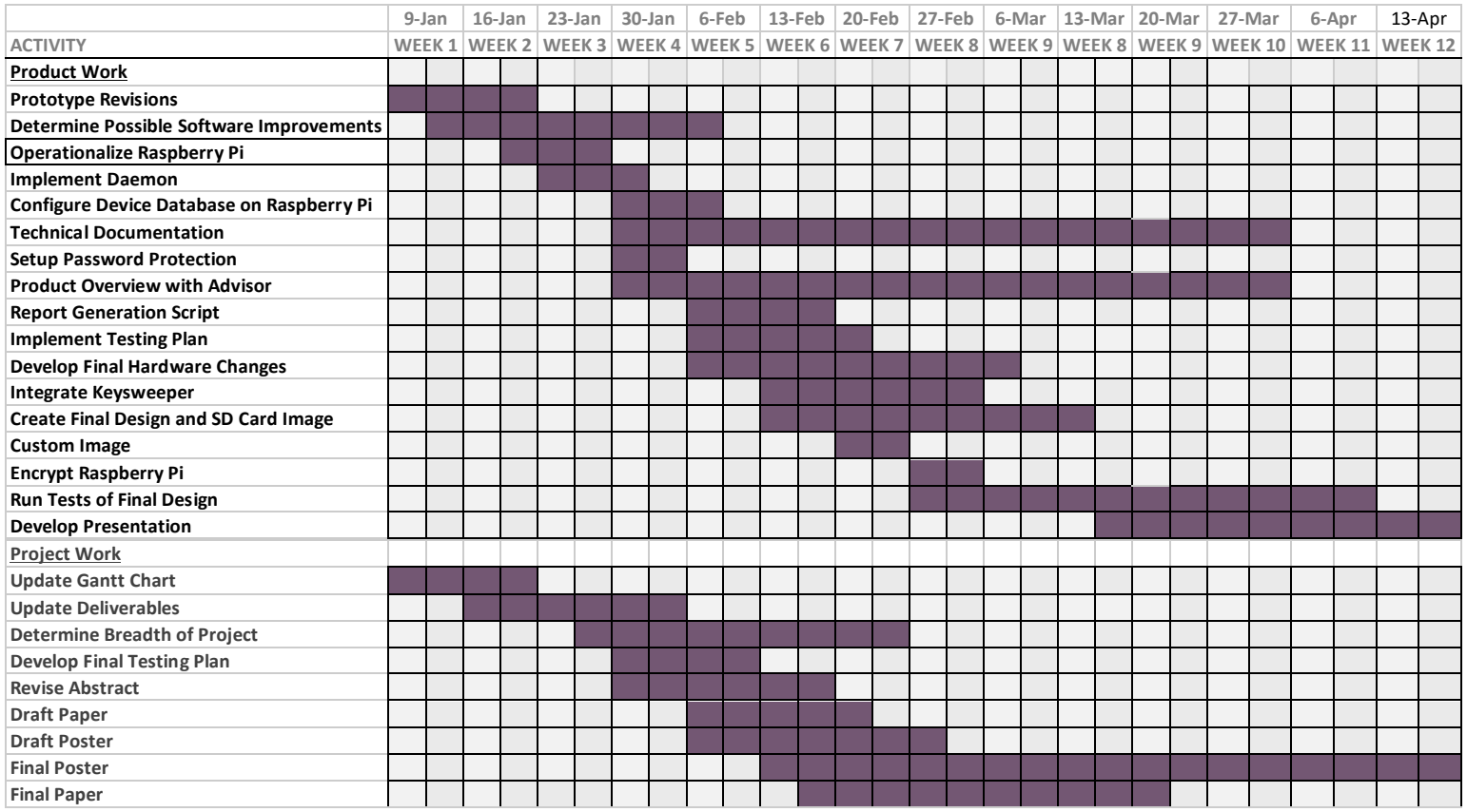


Table 2 In-Depth Timeline Estimate

First Semester	**Week 1**	● Establish a need ● Develop concepts for solutions ● Determine project
	Week 2	● Determine feature set ● Finalize project choice
	Week 3	● Begin feature analysis and design ● Conduct research into the extent of the project ● Order hardware
	Week 4	● Continue feature analysis and design ● Develop a working model of basic architecture ● Test Mousejack exploit
	Week 5	● Test Mousejack ● Begin in-depth design and usability research
	Week 6	● Conduct usability research ● Develop prototype platform
	Week 7	● Begin prototype improvements and bugfixing
	Week 8-12	● Continue prototype development ● Develop final presentation
Second Semester	**Week 1**	• Update project requirements and scope • Fill out Spring Gantt chart and deliverables • Begin developing the daemon configuration • Create a system image with mithoren preinstalled
	Week 2	• Continue work on daemonization • Finalize scope decisions • Begin configuration of the vulnerable devices database
	Week 3	• Finalize database set-up • Begin final stage of documentation • Begin password-protection engineering • Begin final testing plan

		• Revise abstract to fit updated scope
	Week 4	• Finalize the report format for the data export component of the architecture • Implement final hardware changes • Implement testing plan • Work on draft paper • Work on draft poster
	Week 5	• Finalize the draft poster and paper • Attempt integration of Keysweeper tool • Create the final touches on the user interface • Develop the VM image • Work on draft paper • Work on draft poster
	Week 6	• Encrypt the Raspberry Pi following image creation • Continue testing • Continue integration of Keysweeper tool • Work on draft paper • Work on draft poster
	Week 7	• Revise draft paper • Continue integration of Keysweeper tool • Revise draft poster
	Week 8	• Continue Keysweeper integration • Continue final paper and final poster revision • Continue testing and product revision
	Week 9	• Finalize Keysweeper integration • Continue final paper and final poster revision • Continue testing and product revision
	Week 10	• Continue final paper and final poster revision • Continue testing and product revision • Prepare for tech expo
	Week 11	• Continue final paper and final poster revision • Continue testing and product revision • Prepare for tech expo

2.3. Deliverables

Table 3 presents the deliverables that have come out of this project. These are listed in order of estimated date of completion and contain both tangible and intangible objectives.

Table 3 Deliverables

Deliverable	Description	Duration	Est.Completion
Operational Raspberry Pi	Have an operational installation of mithoren on the Raspberry Pi.	1 week	January 23rd
Custom Image	Have a VM image of Raspbian with Mithoren preinstalled	1 day	January 25th
Implement a Functional Daemon	A daemon will be configured which can start/stop/restart mithoren on the OS level	1 week	February 3rd
Setup Database	Have a database configured and ready for data which can contain information about discovered vulnerable devices	1 week	February 12th
Finalize Vulnerability Report and Email	The final data export format will be established and email will function as intended	2 weeks	February 20th
Integration of Keysweeper	If time allows, add a module which operates the Keysweeper tool into the suite	2 weeks/?	February 27th
Draft Poster and Report	An initial version of the poster and report will be completed featuring all necessary components and desired designs	3 weeks	March 6th
Encrypt R. Pi	The Raspberry Pi will be full-disk encrypted.	1 day	March 13th
Technical Documentation Milestone	Full documentation of classes, methods, and files used in the Mithoren project as well as an installation guide and summary	4 weeks	March 27th
Final Poster	Revisions to the poster	1 week	March 20th
Final In-Class Presentation	Have a deck and presentation ready for a final in-class presentation	2 weeks	March 27th
Tech Expo	Be prepared to present the project	3 weeks	April 11th
Final Report	Finalize all changes to the project report	3 weeks	April 17th

2.4. Budget

Table 4 presents the budget for this project. Estimates for labor have been made respective to the time that will be needed to develop this solution. The only costs actually incurred are those regarding purchased equipment.

Table 4 Project Budget

No.	ITEM	UNIT, EACH	UNIT PRICE, $	LINE ITEM TOTAL
Hardware and Hardware Configuration				
1	Labor	10	$75.00	$750.00
2	Raspberry Pi (with Screen)	1	$50.00	$50.00
3	Crazyradio PA	2	$40.00	$80.00
4	Logitech Unifying Dongle	2	$20.00	$40.00
5	Dell 714 Keyboard	1	$60.00	$60.00
6	Insignia Wireless Keyboard	1	$30.00	$30.00
7	Logitech K360	1	$30.00	$30.00
Software Development				
4	Labor	150	$75.00	$11,280.00
Supplies				
5	Misc. Supplies	-	$ -	$
	Subtotal			$12,280.00
	Withholding Labor			$(12,000).00
	Total to Authors			$280.00

3. Technical Elements

3.1. Software

Software for Mithören is comprised of the operating system image, the application itself and the keylogging/radio interception tools. This software is what interfaces with both the hardware radio equipment and the user. The software was developed on Unix-like systems in vim and Atom. Testing was carried out through the Python unittest and mock libraries. Software is maintained in source control with code reviews carried out by each developer. All software is open source. Further technical explanations and diagrams are shown in Section 4.

3.1.1. Keylogging Program

The primary tool used in keylogging is the existing Mousejack utility, written in Python. This tool communicates with the firmware loaded into a Nordic Semiconductor Unifying USB Dongle, Crazyradio PA, or similar antenna capable of operating on the 2.4 Ghz band. The tool comes with four programs to carry out sniffing, scanning, and network mapping. This data is logged into a local directory in plain text.

3.1.2. Daemon

A daemon was developed based on the systemd architecture that begins sniffing and crawling the network within range of the Mithören capture device once the device is booted up. This service operates based on a set of configurable options and was written

in Bash and Python. The daemon controls the keylogging application and produce human-readable output as well as host all integration to external security tools including the email report.

3.1.3. User Interface

The front-end of the project was written in Python and provides intuitive, menu-based commands for manually interacting with the sniffing service and email console. This interface is heavily documented and informed by existing paradigms for console-based interface design.

3.1.4. Operating System Environment

The operating system environment was chosen to be Raspbian, a Debian variant for the Raspberry Pi architecture. This operating system was turned into an image and maintained in the source control system. Upon loading any committed image to an SD card, the capture device is able to be turned on and immediately start scanning the 2.4Ghz band instantly.

3.2. Hardware

Hardware refers to both the listening device, a Raspberry Pi, and the associated radio antennae.

3.2.1. Radio Equipment

A variety of antennas are capable of operating as the packet capture medium. In this project, the Crazyradio PA is the primary antenna for the proof-of-concept Mousejack module. The nRF24LU1+ Nordic Semiconductor device listens on the 2.4Ghz band and has an effective range of nearly a kilometer when provided with line of sight. This device is flashed with a firmware developed by Bastille Research and included in their Mousejack utility. This updated firmware allows it to receive and send packets used by wireless peripherals.

3.2.2. Processing Hardware

A Raspberry Pi was used as the fundamental processing platform, hosting the operating system image, daemon, tools, and user interface. This device additionally has a screen which allows for easy viewing and demonstration, but is expected to be accessed over an SSH connection when being used in the field.

3.2.3. Sample Victim Keyboards

Several keyboards, listed in *Table 5* have been utilized during the testing, development, and presentation process in order to mock a real-life keylogging situation. The keyboards represent a wide selection of existing and popular keyboards which have been shown to have vulnerabilities.

Table 5 Keyboards Utilized

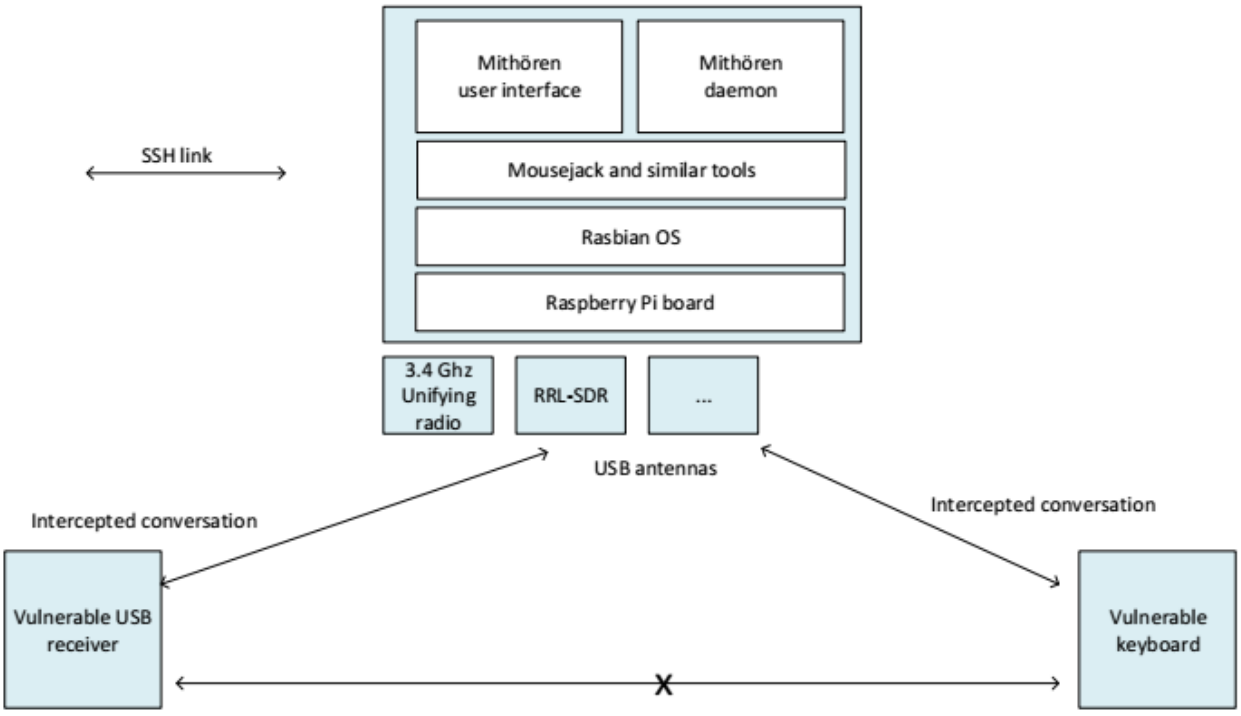
Name	Frequency	Unifying Device?	Product Page
Dell 714 Keyboard	2.4 Ghz	Yes	http://www.dell.com/en-us/shop/accessories/apd/332-1396?l=en&sku=332-1396
Logitech K360	2.4 Ghz	Yes	http://www.logitech.com/en-us/product/keyboard-k360
Gigabyte GK-KM7580	2.4 Ghz	No	http://www.gigabyte.com/products/product-page.aspx?pid=3226#ov
Insignia Wireless Keyboard	2.4 Ghz	No	https://insigniaproducts.com/products/computer-speakers-accessories/NS-PNC5011.html

4. Application

4.1. Application Architecture

Figure 5 displays the overall components and technologies utilized in the Mithören system. The application resides on a Raspberry Pi located within a range of approximately one kilometer of a wireless keyboard and USB dongle operating on the 2.4 GHz band. The specifics about the software package itself will be described further in this section. Each component has been written with test driven development in mind. More in-depth application flow can be seen in *Figure 5.*

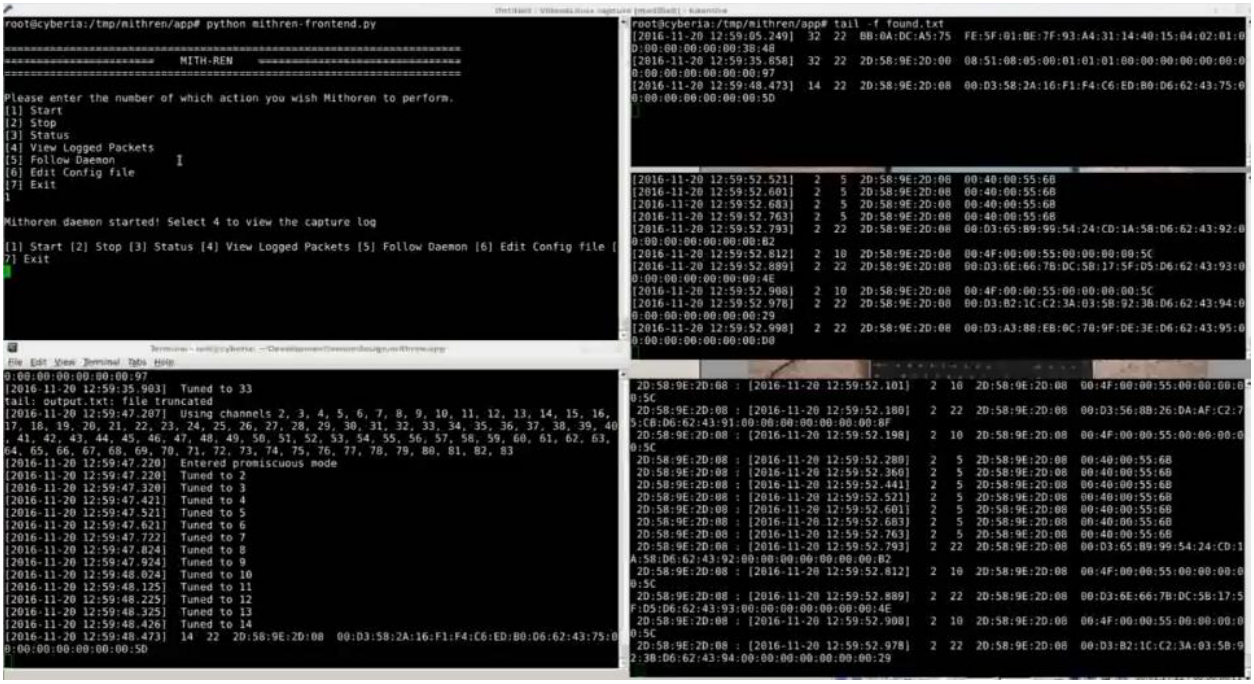
Figure 5 Application Architecture



4.1.1. Collection Daemon

The collection daemon is a background service running on the Raspberry Pi which communicates with the Mousejack tool and other SDR programs to feed into a local encrypted log aggregation system. This collection daemon operates from start-up, instructing Mousejack and any other installed module to scan continuously, waiting for a vulnerable device to be discovered. A screen capture of the internal processing of the daemon from an early debugging implementation of the Mousejack module can be seen in *Figure 6.* The service follows a scan, sniff, dump process described in *Figure 8* and receives instructions from a configuration file. The collection daemon opens a connection with the MongoDB vulnerable device database until the process is paused.

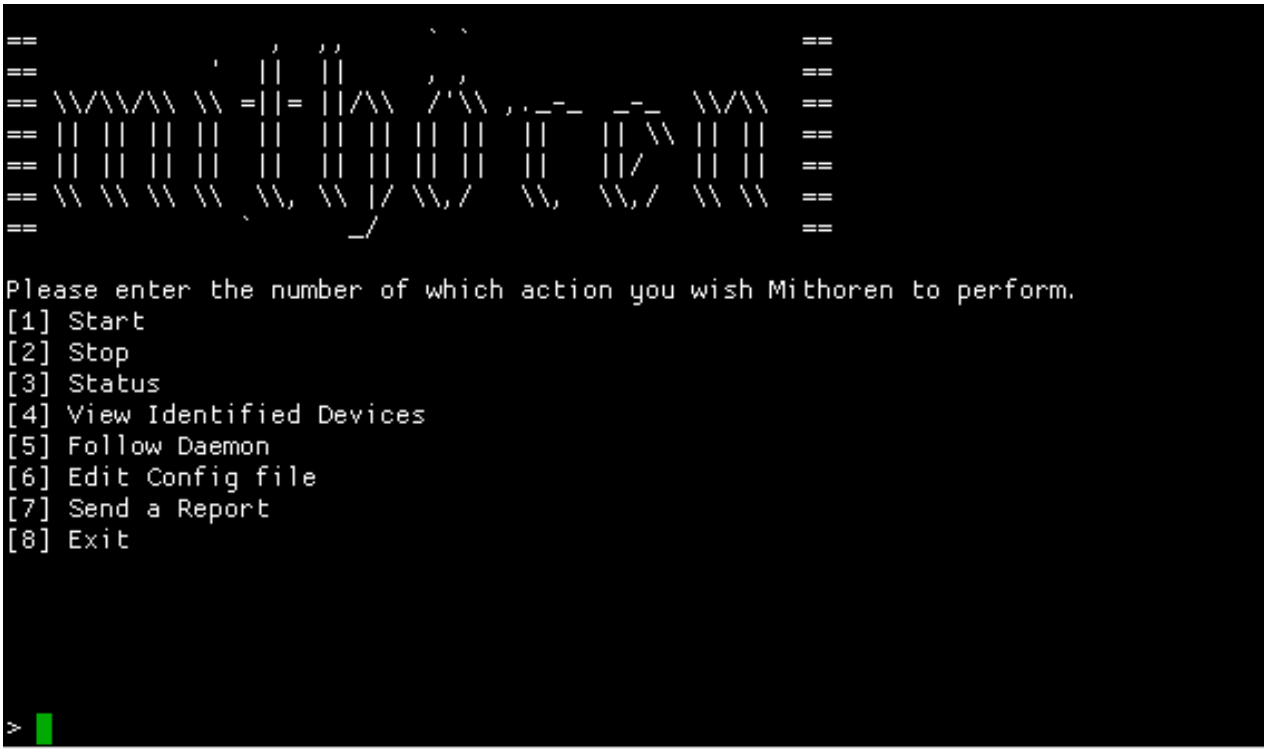
Figure 6 Daemon Processing an Intercepted Packet



4.1.2. Control Interface

The control interface is instrumented in Python and provides users with the ability to configure the collection application, view database entries, and monitor interceptions. A text-based menu has been provided. Data entered is sent to the daemon process or directly to the specified tool. Figure 7 shows the final design and layout of the front end's main menu. The system retains the simplicity of early prototypes and similar toolkits.

Figure 7 Final Control Interface Design



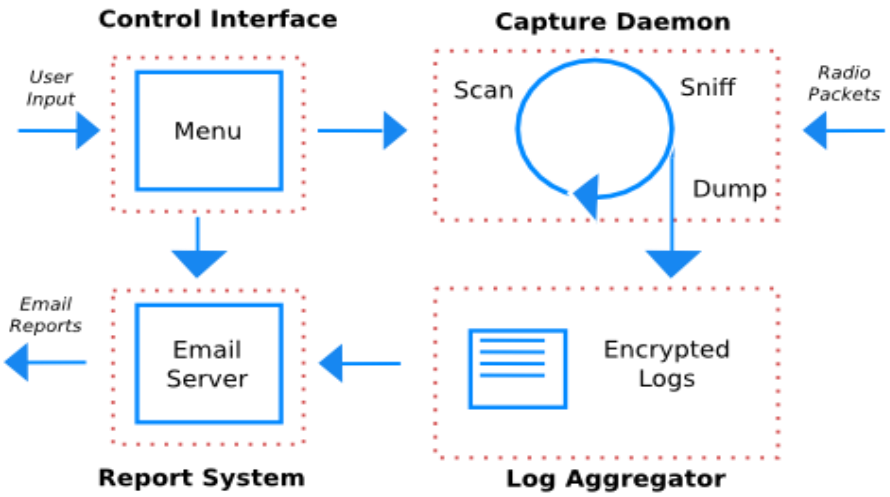
4.1.3. Logging Aggregator

A secured MongoDB database serves as the logging aggregator for the system. When the collector picks up a packet, it generates a post to the database including a timestamp, description, and captured data as well as a guess of the keyboard model from its ID. This data is held in the database until accessed later through the reporting system.

4.1.4. Reporting System

The reporting system serves as a method of data integration with external collection tools. This service integrates with the daemon process and logs to transmit the data over email. By using this protocol, reports are able to be generated and integrated into any modern security analysis tool or alert system.

Figure 8 Application Flow



4.2. Test-Driven Development

This project utilizes test-driven development to assure that functional and security requirements are met and kept in mind throughout the development process. Python provides a suite of testing utilities which simplify the process and allow for code coverage to be achieved rapidly.

4.2.1. Testing Overview

The Mithören testing plan consists of perpetual tests throughout the development lifecycle with heavy emphasis on testing at time of code comitted. Given the range of variables that affect the operation of mithören, system testing is done alongside code functional tests. This section describes the overall approach to our testing process as it applies to the final testing period which is to be held when Work is Done on March 20th, 2017. *Table 6* displays a set of inputs and expected outputs which have been determined for the application. This figure models the specific testing architecture utilized.

4.2.2. Scope

The scope of testing for mithören includes testing the operation of the Raspberry Pi device, the functionality of the operating system image, the requirements of the mithören daemon and frontend, as well as Internet and database connectivity

4.2.3. Objective

These tests function as validation that code committed and configuration changes made throughout the development process result in an operational system that outputs expected results. On March 20th, 2017 all tests were run one final time and recorded below in order to provide a final sanity check on the project.

4.2.4. Entry and Exit Criteria

Entry criteria:

- Work is Done
- Raspberry Pi is configured
- Testing environment is set up

Exit criteria:

- All tests are run
- Results are documented

4.2.5. Reporting

If discrepancies are discovered during testing between expected and actual results, the actual result are recorded and a bug identified. This bug was analyzed for scope. The developers then repaired the issues based on severity and impact on project.

4.2.6. Testing Procedure

Tests fall into one of the following categories:

Stability Test **-** This test focuses on the stability of the host device and our operating system image as well as the availability of device dependencies.

Start/Stop Test **-** This test focuses on the capacity for Mithören's daemon and frontend application to properly launch, stop, and restart according to the systemd service specifications.

User Interface Test **-** This test covers the correct responses of the user interface and validates that actions requested are returned with the proper results.

Functionality Test - This test involves the verification that larger multi-step features are properly carried out through the application.

Connectivity Test **-** This test assures that Internet-connected actions are properly completed.

4.2.7. Schedule of Testing

The data in *Table 6* provides the responsibilities of each team member to run through the testing plan throughout the development period.

Table 6 Testing Schedule

Team Member	Time Period	Frequency
Developer	11/20/16 - 03/20/17	Every commit
Project Manager	11/20/16 - 03/20/17	Weekly

4.2.8. Testing Plan

Table 7 is used as the testing plan. Each item is a key requirement for the functional operation of Mithören. Failed results were recorded and documented.

Table 7 Testing Plan

Stability Test						
Tester	Date	Item	Expected	Actual	Pass/Fail	Bug
CB	03/20/17	1.1 Raspberry Pi Launches	Raspberry Pi turns on	Turns on	Pass	
CB	03/20/17	1.2 Raspbian Boots	Raspbian operating system loads when board turned on	Turns on	Pass	
CB	03/20/17	1.3 Internet Connectivity	If in range of wifi/ethernet, device pings	Google tested	Pass	
CB	03/20/17	1.4 SSH-able	Device can be logged into through SSH	As user	Pass	
CB	03/20/17	1.5 USB-ready	USB keyboard and mouse produce output on system	Both worked	Pass	
CB	03/20/17	1.6 Screen	Screen responds to touch	Functions	Pass	
MW	03/20/17	2.7 Database	Database is intact	Data stable across boots	Pass	
Start/Stop Test						
Tester	Date	Item	Expected	Actual	Pass/Fail	Bug
CB	03/20/17	2.1 Daemon Starts On-Boot	When the Raspberry Pi is powered on, the daemon begins running	Viewed logs	Pass	
CB	03/20/17	2.2 Invoked Daemon Starts	Mithrend.service begins, visible in process list	PID is correct	Pass	

CB	03/20/17	2.3 Invoked Daemon Stops	Mithrend.service is stopped, visible in process list	No process or output visible after stop	Pass	
CB	03/20/17	2.4 Invoked Daemon restarts	Mithrend.service restarted, process visible in process list, log shows restart	PID visible output correct	Pass	
User Interface Test						
Tester	Date	Item	Expected	Actual	Pass/Fail	Bug
CB	03/20/17	3.1 Menu: Start	When selected, starts daemon process	PID visible	Pass	
CB	03/20/17	3.1.1 State: is Started	“Daemon already started”	Output matches	Pass	
CB	03/20/17	3.1.2 State: is Stopped	“Daemon started” Starts daemon	Output matches	Pass	
CB	03/20/17	3.2 Menu: Stop	When selected, stops daemon process	PID gone, output stops	Pass	
CB	03/20/17	3.2.1 State: is Started	“Daemon stopped” Stops daemon	PID is gone Output stops	Pass	
CB	03/20/17	3.2.2 State: is Stopped	“Daemon already stopped”	Output matches	Pass	
CB	03/20/17	3.3 Menu: Status	Displays daemon status info	Output correct	Pass	
MW	03/20/17	3.4 Menu: View Packets	Displays last five packets captured	Displayed four rows	Fail	Not a bug, had recently flushed database
MW	03/20/17	3.5 Menu: Follow Daemon	Displays running log of daemon information	Output correct for each	Pass	

MW	03/20/17	3.6 Menu: Edit Config	Presents user ability to change config	Able to edit	Pass	
MW	03/20/17	3.6.1 Open Editor	Opens the config file through a text editor	Edits successful	Pass	
MW	03/20/17	3.6.2 Save Config	After confirmation, the changes are saved	Saves well. Thanks vim	Pass	
MW	03/20/17	3.7 Menu: Send Report	Report is sent to the designated email	Sent to mw and cb	Pass	
MW	03/20/17	3.8 Menu: Exit	Frontend exits	Exits without error	Pass	
Functionality Test						
Tester	Date	Item	Expected	Actual	Pass/Fail	Bug
MW	03/20/17	4.1 Privilege Failure	“Please run as root”	Output matches	Pass	
MW	03/20/17	4.2 Calls Mousejack	Begins the mousejack module	Mousejack output seen	Pass	
MW	03/20/17	4.3 Retrieves data from Mousejack	Data from mousejack entered into database	Mousejack output seen	Pass	
MW	03/20/17	4.4 Identifies Detected Device	Device Radio ID is associated with a particular product	Database entry created with a guess	Pass/Fail	Known shortcoming of product knowledge
MW	03/20/17	4.5 Generates report	Creates a string of data from databsae in proper format	Successful in both CLI and email	Pass	
MW	03/20/17	4.5.1 Report Accuracy	Report correctly reflects data from database	Report matches database	Pass	
CB	03/20/17	4.5.2 Report	Report content is	Seems so	Pass	

		Content	grammatically correct			
MW	03/20/17	4.5.3 Send Email	Report is mailed from system to recipient listed in config file	Email is opened from remote devices	Pass	
Connectivity Test						
Tester	Date	Item	Expected	Actual	Pass/Fail	Bug
MW	03/20/17	5.1 Database Connection	Daemon and frontend are able to communicate and authenticate with database	Database is secure but allows script access	Pass	
CB	03/20/17	5.2 Email Received	Recipient inbox receives generated report email	Email opened and read	Pass	

4.2.9. Problems Encountered

The primary issues encountered in the early stages of development revolved around the proper identification and tracking of keyboards through the radio channel they communicate on. This project had a substantial amount of technical overhead before the development of user-friendly interfaces could be developed.

4.3. Security

Noting that this device contains logs of potentially sensitive information, has an open SSH port, and is physically vulnerable to being picked up and manipulated, security has been a key objective of the project. The device was encrypted and the interface application requires authorized access to be set before any action occurs.

4.4. User Interface

The application's user interface is capable of being reached from either a local session or a remote SSH connection. After loading the application and setting the initial ownership, the user is presented with a text-based menu. From this menu listing, a user is able to select an option to view logs, configure the daemon, or export the database archives. This simple interface with the application is familiar to security professionals and security researchers.

5. Conclusion

5.1. Fall Semester 2016

The development of an application whose most marketable element is for use in penetration tests environments has presented a number of challenges within an IT environment. Beyond the technical challenges of configuring software defined radio systems, integrating this into a truly useful tool has required extensive research and design consideration. Thus, in the Fall semester, testing and research has consumed much of development time, but is leading to a skeletal prototype of the final product.

5.2. Spring Semester 2017

Emphasis on a firm proof of concept with extensive testing and validation allowed for the team to refine the scope of the project to a single module with a mature data collection and reporting solution. By focusing on open source and test-driven

development, the team hopes that its solid foundation will be able to be beneficial to the security community as a whole. The team remains dedicated to its development into a swiss-army knife for modular radio interception.

Despite the slow progression of this security utility toward the marketplace, this team believes that this project has been able to demonstrate the capability of wireless keystroke interception in an automated and form-factor application with extensive use cases throughout the ever-growing wireless environment.

6. References

1. Drury, Adrian, and Richard Absalom. *BYOD: An Emerging Market Trend in More Ways than One*. Ovum Consulting. May 03, 2013.

2. Newlin, Mark. *Mousejack: Injecting Keystrokes into Wireless Keyboards*. Bastille Research. February 22, 2016

7. Appendix A

7.1. Project Resources

Mithören GitHub repo: https://github.com/wolfmd/mith-ren

Mithören Slide deck: https://wolfmd.me/docs/mithren.pdf

Mithören use case video: https://wolfmd.me/docs/mithrenusecase.mp4

Mithören writeup: http://seclist.us/mithoren-is-an-extensible-platform-for-wireless-peripheral-keystroke-sniffing-for-microcomputers.html

7.2. Additional Resources

Mousejack tool webpage: https://www.mousejack.com/

Mousejack DefCon presentation: https://www.youtube.com/watch?v=00A36VABIA4

Mousejack Whitepaper: https://github.com/BastilleResearch/mousejack/blob/master/doc/pdf/MouseJack-whitepaper-v1.1.pdf

7.3. Expo Poster