Evaluating the Gasday Security Policy Through Penetration Testing and Application of the Nist Cybersecurity Framework

Andrew Nicholas Kirkham
Marquette University

EVALUATING THE GASDAY SECURITY POLICY THROUGH PENETRATION TESTING AND APPLICATION OF THE NIST CYBERSECURITY FRAMEWORK

by

Andrew N. Kirkham, B.S.

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2016

ABSTRACT
EVALUATING THE GASDAY SECURITY POLICY THROUGH
PENETRATION TESTING AND APPLICATION OF THE
NIST CYBERSECURITY FRAMEWORK

Andrew N. Kirkham B.S.

Marquette University, 2016

This thesis explores cybersecurity from the perspective of the Marquette University GasDay lab. We analyze three different areas of cybersecurity in three independent chapters. Our goal is to improve the cybersecurity capabilities of GasDay, Marquette University, and the natural gas industry.

We present network penetration testing as a process of attempting to gain access to resources of GasDay without prior knowledge of any valid credentials. We discuss our method of identifying potential targets using industry standard reconnaissance methods. We outline the process of attempting to gain access to these targets using automated tools and manual exploit creation. We propose several solutions to those targets successfully exploited and recommendations for others.

Next, we discuss GasDay Web and techniques to validate the security of a web-based GasDay software product. We use a form of penetration testing specifically targeted for a website. We demonstrate several vulnerabilities that are able to cripple the availability of the website and recommendations to mitigate these vulnerabilities. We then present the results of performing an inspection of GasDay Web code to uncover vulnerabilities undetectable by automated tools and make suggestions on their fixes. We discuss recommendations on how vulnerabilities can be mitigated or detected in the future.

Finally, we apply the NIST Cybersecurity Framework to GasDay. We present the Department of Energy recommendations for the natural gas industry. Using these recommendations and the NIST Framework, we evaluate the overall cybersecurity maturity of the GasDay lab. We present several recommendations where GasDay could improve the maturity levels that are cost-effective and easy to implement. We identify several items missing from a cybersecurity plan and propose methods to implement them.

The results of this thesis show that cybersecurity at a research lab is difficult. We demonstrate that even as a member of Marquette University, GasDay cannot rely on Marquette for cybersecurity. We show that the primary obstacle is lack of information - about cybersecurity and the assets GasDay controls. We make recommendations on how these items can be effectively created and managed.

ACKNOWLEDGEMENTS

Andrew N. Kirkham, B.S.

I would like to thank the Marquette University GasDay Lab for making this work possible. Thank you to Tom Quinn, Dr. Ronald Brown, and Nathan Wilson for giving me the opportunity to perform this research without hesitation.

I would especially like to thank Dr. George Corliss and Tunde Ishola for their countless hours of guidance throughout this work. Thank you for constantly pushing me to do better, even when I thought I was complete. Thank you to my committee members, Dr. Richard Povinelli and Dr. Debbie Perouli for providing insights and unique questions to challenge me along the way and improve the quality of this work.

I would also like to thank the graduate and undergraduate students of the GasDay Lab. Your constant remarks, questions, ideas, and jokes kept me motivated to finish this work. Without your support, this work would not have been possible.

This work is dedicated to my mother, Janet Kirkham, whose love and sacrifice made this work possible. This work is also dedicated to my grandfathers, Don and Richard, who inspired me to become an engineer and to never stop learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Cybersecurity in the Energy Industry

This thesis explores cybersecurity at Marquette University's GasDay Laboratory - a research lab that supports the natural gas industry. Our goal is to evaluate and test cybersecurity using industry standard techniques to increase the cybersecurity capability. This thesis is composed of three whitepapers addressing different facets of the cybersecurity policy of GasDay.

This chapter introduces cybersecurity, cybersecurity in the energy industry (specifically natural gas), and an overview of cybersecurity at GasDay. The focus of this research is in the context of GasDay, but is applicable to both the energy industry and to other university research labs. A synopsis of the research is also presented.

1. Brief Introduction to Cybersecurity

Cybersecurity is the practice of protecting computers, networks, and data from unauthorized access or modifications [1]. Cybersecurity also seeks to protect computers, networks, and data from unintended access and modifications - such as someone accidently deleting or changing an important document.

Cybersecurity is comprised of three elements: confidentiality, integrity and availability (CIA triad) [2]. Confidentiality represents measures or rules that restrict access to information; only those who are allowed to view information can view it. Integrity is assuring the accuracy and trustworthiness of data; changes in data are tracked

or prevented from accidental modifications. Availability guarantees the information is accessible to authorized people. If any element in the triad is missing, then a system is not secure. For example, a system that is easily available and tracks all changes but does not prevent any individual from accessing it is not secure.

Cybersecurity experts are tasked with finding vulnerabilities in computer systems. A vulnerability is a weakness in the system that allows an attacker to compromise an element in the CIA triad. When an attacker successfully reduces the security of the system, they have exploited the vulnerability.

Testing for vulnerabilities is similar to software testing – the goal is to find errors. Finding errors indicates a successful test. Not finding errors does not mean the system is error-free, it only suggests that no errors were detected. Therefore, cybersecurity experts wish to find faults. Throughout this thesis, we will discuss different tactics for finding and mitigating vulnerabilities in GasDay.

2. Cybersecurity Concerns of the Energy Industry

In 2014, the energy industry was the fourth most targeted industry for cyberattacks [3] including some electrical utilities that reported daily attacks [4]. There are generally two motivating factors for targeting energy organizations: money and disruption. Intellectual property, research, and prospective gas/oil fields can all be worth a large amount of money to certain people. However, this same information can be used for disruptive purposes. A disruption in the energy grid of the United States would result in chaos. In one estimate, an attack on a Northeastern United States electricity generation

control room would result in up to $1 trillion loss accompanied by increased mortality rates and a collapse of infrastructure [5].

The attack mentioned in [5] was theoretical, but historical incidents suggest that it is very possible. Several historical attacks (such as Stuxnet, Duqu, and Flamer [6], [7]) aimed to destabilize the control systems of energy substations over time. In 2015, a group known as APT1 was reported to have gained administrative access to several energy utilities. This allowed APT1 to gather engineering blueprints and access controllers for pressure valves on natural gas pipelines [8].

The energy industry is very cautious about any software that needs to be installed on site. Adding software or communicating with a third party adds another risk that the energy utility needs to evaluate. GasDay is one such software package on which many natural gas utilities rely.

3. Marquette University's GasDay Laboratory

The Marquette University GasDay Laboratory is a research lab and small business within Marquette University that forecasts approximately 20% of the natural gas consumed by residential, commercial, and industrial customers in the United States. These forecasts are delivered daily to natural gas local distribution companies (LDC) by a program also named GasDay or with GasDay Web. GasDay Web is an upcoming web-based version of GasDay that will allow customers to generate forecasts on a website outside of GasDay or the LDC. With GasDay indirectly influencing several billion dollars of natural gas flow, cybersecurity is a factor in many decisions.

GasDay wants the software that we deliver to customers to be secure. This work is to demonstrate GasDay's efforts in securing the software product. By finding vulnerabilities and gaps in security, we are able to continuously improve GasDay's ability to deliver a secure product. Identifying weaknesses early in development allows GasDay to remedy the issues before the product is in the customer's hand.

4. Cybersecurity at GasDay

Before this thesis, GasDay's primary concerns were for the students employed in the lab, Marquette University, and the quality of forecasts that are delivered to customers. Since the start of this work, GasDay retains its primary concerns but understands better what it means to quantify and protect against potential cyber threats.

As a member of Marquette University, GasDay is subject to the Family Educational Rights and Privacy Act (FERPA) [9]. FERPA prevents GasDay from disclosing information about the students who are employed without their written consent. This means that any publicly available item at GasDay needs to be screened to comply with FERPA. FERPA also extends to the internal documents GasDay might wish to display to members of the lab.

As a member of the energy industry, GasDay is subject to any regulations that exist for third party supporters. However, the United States does not have any official regulations for the energy industry. The Department of Energy does make several recommendations for cybersecurity, which we will explore in depth in Chapter 4.

This thesis evaluates and makes suggestions for the practice of cybersecurity at GasDay. This thesis also demonstrates where gaps are present in the cybersecurity policy. The next section explains how this thesis is organized to achieve these goals.

5. Contents of Thesis

Chapters 2, 3, and 4 are the primary contents of this thesis. Each chapter is considered a separate document and therefore has its own introduction and conclusion. With minor edits, each chapter can be given to any GasDay customer interested in a specific cybersecurity facet of GasDay.

Chapter 2 discusses network penetration testing and discusses the findings after performing a penetration test on the GasDay lab. Penetration testing is the process of attempting to gain access to resources of GasDay without knowledge of any valid credentials. We present our method of identifying potential targets and the results of attempting to gain access to these targets. We present several solutions to those targets successfully exploited and recommendations for others.

Chapter 3 discusses GasDay Web and techniques to validate the security of GasDay Web. We use a form of penetration testing specifically targeted for a website. We present several vulnerabilities that are able to cripple the availability of the website and recommendations to mitigate these vulnerabilities. We then present the results of performing an inspection of the GasDay Web code to uncover vulnerabilities undetectable by automated tools.

Chapter 4 applies the NIST Cybersecurity Framework to GasDay. We present the Department of Energy recommendations for the natural gas industry. Using the

Department of Energy recommendations and the NIST Framework, we evaluate the overall cybersecurity maturity of the GasDay lab. We present several recommendations where GasDay could improve the maturity levels that are cost-effective and easy to implement. We also present recommendations to GasDay to implement a continuous evaluation.

Chapter 5 summarizes the findings of this thesis. The results show that cybersecurity at a research lab is tough to implement. We analyze the common difficulties from each assessment and summarize the recommendations. We identify lack of information about cybersecurity and the capabilities of GasDay as the primary obstacle. We demonstrate that even as a member of Marquette University, GasDay cannot be reliant on Marquette for cybersecurity. We make recommendations on how these items can be effectively created and managed in the future.

CHAPTER 2

Applying Network Penetration Testing to GasDay: A Case Study

This chapter discusses finding weaknesses in a networked system. Networked systems offer a variety of potential security weaknesses, including misconfigured systems, unintentionally public information, or out-of-date systems. GasDay wants to find such vulnerabilities before an attacker does. GasDay has experienced one reported incident in which it was targeted from an unknown Chinese IP address. GasDay also could become the target in the future since natural gas companies are starting to be targeted [8]. GasDay also had physical items stolen, and it was unknown if the thief would be able to access GasDay resources. We present penetration testing as a method to identify vulnerabilities and minimize future security incidents. Throughout this paper, the symbol ⌘ will be used to denote fixes for discovered vulnerabilities.

1. Introduction to Penetration Testing

Penetration testing is the process of attempting to gain access to resources of an organization without knowledge of any valid credentials [10]. The difference between a penetration tester and an attacker is permission from the owner. For both an attacker and a penetration tester, the methods are similar:

1. Acquire as much information about the target as possible,
2. Use relevant information to build an attack plan, and
3. Attack the target.

A penetration tester wants to find vulnerabilities in the system before an attacker does. The test is similar to an annual physical. A doctor examines you and performs tests for various dangers - even if no symptoms are demonstrated. However, like a medical test, a negative test, one that finds no problems, does not mean that the system is without fault and does not need to be checked on in the future.

A penetration test looks for any weakness in the system and assesses the severity of those weaknesses. Beyond finding a weakness, the tester also identifies the risk of the vulnerability,

$$Risk = Impact * Probability / Cost.$$

Impact is the potential operational cost of a successful attack. A server that holds all of GasDay's customer data would have a high impact if it were compromised compared to a laptop outside of the firewall with no access to GasDay data. *Probability* is the likelihood of the weakness being exploited. This includes the feasibility of the exploit as well as the chance of network defense responding to the attack. *Cost* is the effort to mitigate the vulnerability. This cost is measured in both financial terms and man-hours.

A penetration test also has a second goal to identify hard-to-access systems. At the heart of information security are three goals known as the CIA triad: confidentiality, integrity, and availability [2]. For a policy or procedure to be effective, it must meet each goal in the triad. A system or resource that is confidential but impossible to access is not secure. A resource that is easy to access and confidential, but can be modified at will (the integrity is not preserved) is also not secure.

By identifying vulnerabilities, we can either fix or mitigate them to increase their security. We might also find that a resource is hard to access, and we can increase

security by making it more accessible. The penetration test is repeated to verify that we are satisfied with our overall security level at each point of the triad. To identify insecure resources and vulnerabilities, we first need to acquire information about our resources.

2. Information Gathering

The first step of a penetration test is to gather information about the target. Information is used to determine various entry points into an organization. Entry points are often electronic, but can be physical or biological. Many exploits require a combination of entry points to be successful. For example, an exploit might rely on a user clicking a malicious file to exploit an out-of-date software package. The more entry points we can find, the more attack vectors we can use.

In this section, we discuss two strategies to gather intelligence, open source intelligence and footprinting. Open source intelligence is the process of gathering information via publicly available sources. Footprinting gathers information by direct interaction with the target. Both of these tactics produce information that can be combined to build an action plan that will guide the penetration test.

2.1. Open Source Intelligence

Open source intelligence (OSINT) consists of any piece of information that can be gathered from a publicly available source. These sources can be physical, such as newspapers and television, digital sources, such as blogs and social media websites,

academic sources, such as publications and conferences, or amateur reporting sources, such as radio monitors and Google Earth.

OSINT focuses only on actionable intelligence - intelligence that can be used later. Old or inaccurate information may not be useable. We may not know if information is accurate. Therefore, we need to validate any information that we gather. Validating information is done by finding multiple sources or by direct interaction (footprinting).

An attacker focusing on GasDay might look for former/current employees, our customers, or information about our models and techniques. An attacker also might be interested in our physical location or where we purchase our coffee, but these factors are omitted from our assessment. The scope of intelligence gathering is any information that can be acquired from publicly available Internet resources that might assist in mounting an attack.

2.1.1. Customer Information

GasDay needs to protect the identity of our customers. Each of our customers has a non-disclosure agreement with Marquette University, and any information that identifies our customers might violate that agreement.

A large amount of information about the internal staff (both full time and students), customers, and delivery schedule can be found through a rate case filed by Enstar Natural Gas [11]. This document also contains information about Enstar staff, modeling techniques used by the lab, and a description of the data that Enstar provides the lab. Also, testimony provided to by New Mexico Gas Company can be found on their website that outlines their use of GasDay [12].

Further information about GasDay customers can be found through the Marquette Office of Research and Sponsored Projects. This document outlined awards that faculty members received and mentioned customers or entities acting on behalf of customers [13].

2.1.2. Infrastructure Assets

We also want to protect our intellectual property. GasDay hosts the data from our customers and code for our models internally. If an attacker were to access our network directly, they would be able to access our intellectual property. If an attacker can find information about the network infrastructure, then their potential attack vectors grow as well as their confidence in those vectors.

The first step an attacker might take would be to determine the IP address block for GasDay. GasDay's IP address can be approximated through Marquette's registered IP block (134.48.0.0/16). By Googling for *"gasday" + "research"*, we find a Marquette wiki page written by a former lab member [14]. This page mentions a "GasDay wiki." By attempting to reach the wiki through *wiki.gasday.com*, we can obtain an exact IP address for that server, as well as the GasDay network.

Attempting to access this IP directly results in the WAMP server configuration page seen in Figure 1. From here, we can hypothesize a potential naming schema for computers: GAS-XX-NNNN, where X appears to be the type (VM for virtual machine, DSK for desktop), and NNNN is a unique ID. We also gain a potential username for the administrator account for GasDay (labadmin). ⌘ The server configuration page has had its access restricted to the server and is no longer available. The DNS servers also have

been updated to query *gasday.com* instead of the lab IP. ⌘ By visiting an old webpage about Capstone projects and looking at the CSS for the page, we confirm the naming schema [15].

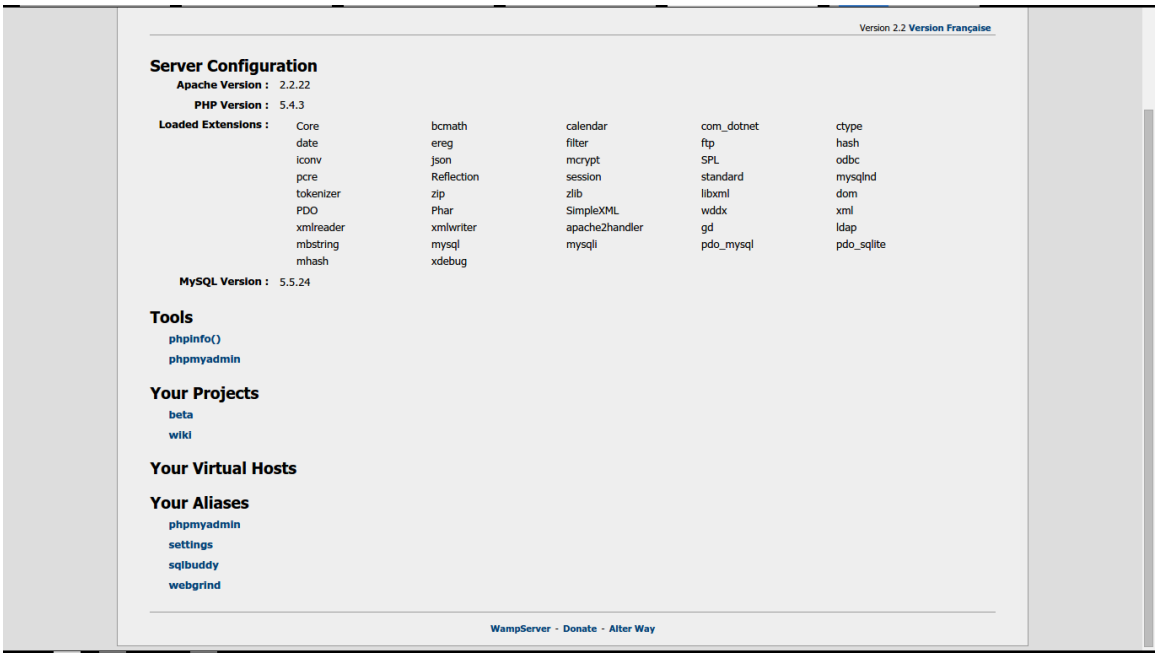

Figure 1: WAMP Server showing out of date software versions This server is accessible anywhere and shows out of date PHP, MySQL, and Apache versions. In addition, it shows two websites available on the server which are not accessible anywhere

By visiting the GasDay website, we find a valid email address. By using the testimonies in [11], [12], we identify another valid email address. Searching for "GasDay" in Google Scholar reveals several articles GasDay has published, giving several valid email addresses. We can determine the provider of the email by performing a mail exchange lookup [16]. We can use these to launch targeted social engineering attacks, such as phishing emails, or as potential usernames to the system.

Table 1: Valid email address formats for GasDay

Provider	Format	Valid usernames (examples)
1and1	{role}@gasday.com	sales; grad
Marquette	{firstname}.{lastname} @marquette.edu	ronald.brown thomas.quinn

With this information, we have discovered several lab members, GasDay customers, GasDay infrastructure, and several entry points. By directly interacting with the target (GasDay), we can validate this information and gather more information.

2.2. Footprinting

Footprinting consists of gathering information by directly interacting with the target [17]. This gives information from an external perspective about the organization. These interactions are usually electronic, but can be physical interactions or interactions with people. For this thesis, only electronic interactions are considered.

By probing devices on a network and services on the devices, we learn about the hosts on the network. Our goal is to develop a prioritized list of targets. High priority targets are those that are most likely exploitable or have the best chance of leaking information.

In this section, we present external footprinting techniques that can be used to gather information about GasDay: port scanning, service identification, and sweeping

(SNMP and SMB). We describe each technique, tell how we applied it, present the results, and discuss their implications.

2.2.1. Port Scanning

An attacker wants to know what systems are on the network. One way we can determine this is through a port scan. Port scanning tests a host to see if a specific port is open. If no port is open, then the host either doesn't exist, or it will not communicate to us. From our open source intelligence, we know of a web server with which we can start.

We use the industry standard tool Nmap (network mapper [18]). Nmap uses IP packets to determine what hosts are available, their ports and running services, and the operating system of the host. It can also configure packets to evade firewalls, filters, and intrusion detection systems.

The first scan we run is a ping scan. The intent of a ping scan is to quickly find active IP addresses that we can target. By attempting to ping every IP address within a subnet, we quickly reduce our possible targets. Running a host discovery Nmap scan against 134.48.94.0/24 gives 31 hosts available.

The next scan we run is a TCP SYN scan. The intent of this scan is to determine what ports are open, closed, and filtered on a target. We can use this information to guess what services are running, and begin to tailor our attack plan. This scan sends a SYN request to a port and waits for a SYN/ACK response. A response indicates an open port that is listening, while a reset indicates that the port is closed. Using this technique, we can identify what ports are open on the 31 hosts.

Since we never finish the TCP handshake (by not sending the last ACK packet), this technique is stealthy - it does not alert anyone or set of any alarms. During this process, we want to hide our identity as much as possible. By setting off alarms, we may prevent future attacks from working and provide the target with our information. During a penetration test, we might want to try to set off some alarms to test them. Because our test seeks to mimic an attacker, we attempt to remain as stealthy as possible.

The next scan (called a version detection scan) we use interrogates each open port to determine what the port is doing. This scan determines protocols (FTP, HTTP), application names, and version numbers for those open ports. Using this information, Nmap can determine the underlying device (router, server, printer, etc.) and operating system the device is running. Knowing what services are running in addition to the underlying device type allows us to identify specific hosts that are out of date or are likely to contain valuable information.

This scan generates a lot more traffic than the SYN or ping scan and can be detected by intrusion detection systems. We choose to interrogate only those ports where we cannot make an intelligent guess (such as ports that are unregistered or have multiple services associated with the port). ⌘ The Marquette firewall will temporarily blacklist an external IP address after a certain number of service probes are detected. The duration of this blacklist increases after each detection. ⌘

After performing a full port scan, we notice several machines hosting web sites on port 80. These websites host several types of services including printers, software services (TeamCity and Subversion), and a default Apache homepage. The WAMP

server page in Figure 1 also showed several in-house websites that are not available outside of the GasDay network.

2.2.2. SNMP Sweep

Another tactic that can yield actionable information about the system is a simple network management protocol (SNMP) sweep. SNMP is used to expose data about a device to the network. This data could contain information such as the processor and all processes on the device. SNMP contains all of this data in the form of a database and set of data objects. The database and data objects are queried through the management information base (MIB). Each managed device keeps its own MIB, so the data is dependent on the implementation of the device [19].

Performing an SNMP sweep on the GasDay network reveals three printers. Two printers are owned by GasDay, and one is a Marquette PrintWise printer. Each of these devices displays information about ports on which it communicates with various devices around the lab and Marquette campus. One printer reveals that it is running one process for its RAM and one for its NAND drive.

2.2.3. Other Services

We look for other services during footprinting including databases, virtual machines, and web applications. These services are found at organizations and can be used as entry points into a network. GasDay Web is not covered by this test, and information about its vulnerabilities can be found in the next chapter. Databases can be

found during port scanning. Different types of databases run on different ports: MySQL runs on port 3306 by default, while Microsoft SQL runs on ports 1433-1434. Each database provider has different vulnerabilities and different credentials. We can attempt to connect to the database using the default user and administrator credentials. For GasDay, five databases were found to be accessible during port scanning.

Several hosts on a network might be virtualized to save money or physical space. Exploiting a virtual machine might prove more difficult than physical machines, but exploiting the underlying server could grant access to all of the virtual machines. Nmap can detect the presence of certain hypervisors such as VMWare ESXI and Microsoft Hyper-V. GasDay maintains one ESXI server and two Hyper-V servers, which can be identified by their hostnames during port scanning (GAS-ESXI and GAS-WSHV). Only one of the Hyper-V servers can be accessed outside of the Marquette domain.

Our attack plan can now be tailored to the known running systems. We have identified several open web servers running critical applications, databases, and open virtual machine servers. We have also identified several other services that are exposed outside of the Marquette domain. Each service is vulnerable to different exploits that we can look for automatically or by checking manually.

3. Vulnerability Assessment

Vulnerability testing is the process of searching for and discovering weaknesses in a system ranging from software misconfigurations (such as leaving the default username and password) to insecure application designs (such as the Heartbleed vulnerability [20]) [17]. Vulnerabilities are dependent on the system under test; no two

systems will have the same set. Vulnerability analysis can be used to verify if a security monitor is working correctly, test administrative access, or check that known vulnerabilities are mitigated. The breadth of this analysis is every machine that is permanently connected to 134.48.94.0/24; laptops are not considered.

Footprinting is used to guide this assessment. Knowing what ports are open and which services are running on those ports narrows potential vulnerabilities greatly. Because footprinting is only an estimate, automated vulnerability tools can result in many false positives and false negatives. Therefore, a manual approach is required to verify the existence (or absence) of a vulnerability.

In this section, we outline the use of automated tools OpenVAS and Metasploit. We discuss how the tools work in conjunction with a manual assessment, show how they are applied to GasDay, and discuss the results.

3.1. Automated Tools

Automated tools offer fast enumeration of the most common vulnerabilities for a variety of operating systems. The tools we will use here are OpenVAS [21] and Metasploit [22]. OpenVAS is a tool that automatically scans targets for potential vulnerabilities. It starts by running a port scan to find open services. Once the services are identified, they are tested for known vulnerabilities and misconfigurations. OpenVAS contains over 43,000 different tests that are run to look for vulnerabilities [21]. Each test has an associated percentage that indicates how confident OpenVAS is that the vulnerability exists. These results should be manually validated to reveal false reports. OpenVAS can be very aggressive and noisy; it can attempt to brute force login

credentials and exploit vulnerabilities without validation, which could set off intrusion detection systems. Figure 2 shows an example OpenVAS scan for the GasDay network. We find several critical vulnerabilities with a severity ranking, probability of a correct guess (QoD), and a description of how to mitigate the vulnerability. An attacker could produce a similar result to get an idea of what attacks are likely to work.

Vulnerability		Severity	QoD	Host	Location	Actions
VMSA-2015-0007: VMware ESXi OpenSLP Remote Code Execution (remote check)		10.0 (High)	97%	134.48.94.101	general/tcp	
Discard port open		10.0 (High)	75%	134.48.94.90	9/tcp	
MS15-034 HTTP.sys Remote Code Execution Vulnerability (remote check)		10.0 (High)	95%	134.48.94.129	80/tcp	
Microsoft SQL Server Multiple Vulnerabilities (3065718) - Remote		8.5 (High)	80%	134.48.94.129	general/tcp	
VMSA-2014-0008: VMware vSphere product updates to third party libraries (remote check)		7.5 (High)	97%	134.48.94.101	general/tcp	
php Multiple Vulnerabilities -01 June15 (Windows)		7.5 (High)	80%	134.48.94.50	80/tcp	
php Multiple Vulnerabilities -02 June15 (Windows)		7.5 (High)	80%	134.48.94.50	80/tcp	
php Multiple Vulnerabilities -03 June15 (Windows)		7.5 (High)	80%	134.48.94.50	80/tcp	
php Multiple Remote Code Execution Vulnerabilities July15 (Windows)		7.5 (High)	80%	134.48.94.50	80/tcp	
Lighttpd Multiple vulnerabilities		7.5 (High)	99%	134.48.94.68	80/tcp	
Oracle MySQL Multiple Unspecified vulnerabilities-01 Feb15 (Windows)		7.5 (High)	80%	134.48.94.109	3306/tcp	
VMSA-2015-0001: VMware vCenter Server, ESXi, Workstation, Player, and Fusion updates address security issues (remote check)		7.1 (High)	97%	134.48.94.101	general/tcp	
VMSA-2014-0006: VMware product updates address OpenSSL security vulnerabilities (remote check)		6.8 (Medium)	97%	134.48.94.101	general/tcp	
Microsoft SQL Server Elevation of Privilege Vulnerability (2984340) - Remote		6.8 (Medium)	80%	134.48.94.129	general/tcp	
OpenSSL CCS Man in the Middle Security Bypass Vulnerability		6.8 (Medium)	99%	134.48.94.15	443/tcp	
http TRACE XSS attack		5.8 (Medium)	75%	134.48.94.133	80/tcp	

Figure 2: OpenVAS scan on GasDay network There are several vulnerabilities marked as high severity on six different computers. Each vulnerability contains a description, a mitigation strategy, and links to resources

Metasploit is a tool that is used for developing and executing exploits against a target. Metasploit contains over 3000 built-in exploits and exploit fragments that can be configured [23]. Metasploit is used to check for and run manual exploits against a target. These exploits are tailored for specific software versions and operating systems. Using

the information gathered during footprinting, we can search the Metasploit database for known exploits.

3.2. Manual Assessment

During port scanning (in Section 2.2.1), three printers were discovered. Printers can be queried using a language standard to all printers: printer job language (PJL) [24]. Metasploit comes packaged with PJL queries that allow a user to upload and download files, change environment variables, and list the contents of a directory. The contents of both the RAM and the NAND drive of a printer can be listed. Figure 3 shows the contents of the drives displaying documents that have been printed recently. Using Metasploit, we can download these files without disrupting the normal operation of the printer. In Figure 3, two purchase logs and a research assistant renewal can be seen. These documents might contain information such as credit card numbers or student information. Other documents that GasDay prints could contain detailed customer information, Marquette information, or personal documents. We also have the option of uploading files to the printer directly. This could be used to house payloads to be used later or to upload a custom firmware that will monitor all future print jobs. GasDay should disable PJL, SNMP, and NFS disk access to disable these access points. Disabling these access points should not disrupt the normal operation of the printer and would prevent these exploits from occurring.

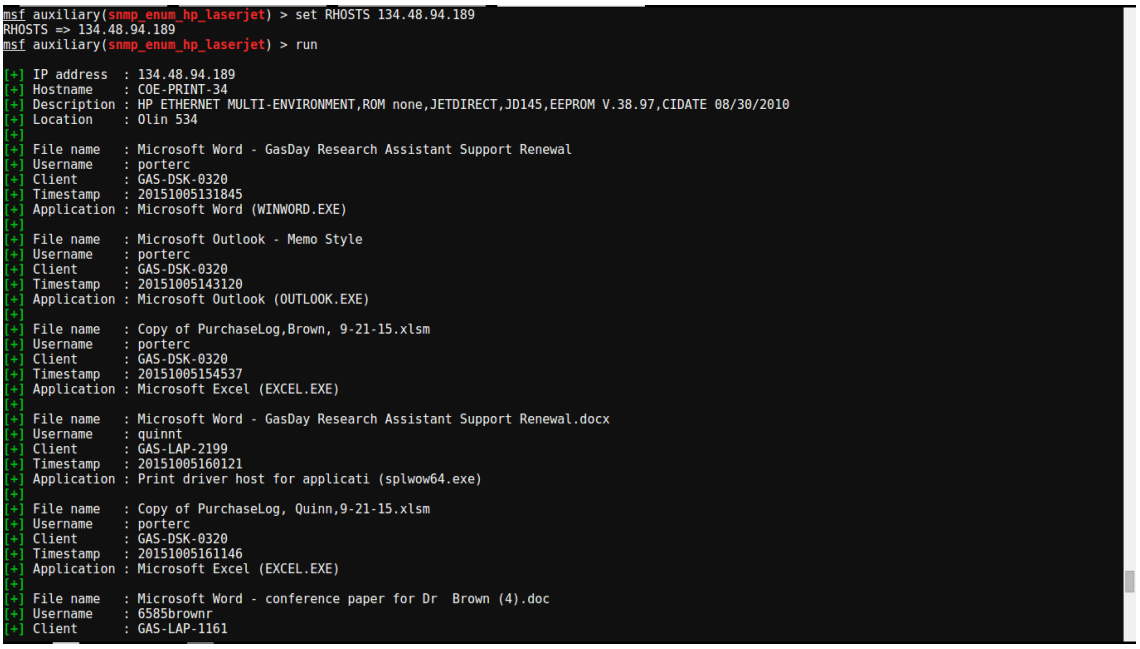

Figure 3: Truncated recent print job enumeration using SNMP Each of these documents could be downloaded from the printer. We also notice the username and application used to print the document

OpenVAS reported that the ESXI server was running an out of date third party library (in Section 3.1). One of these third party libraries was OpenSSL, running a version that was potentially vulnerable to CVE-2014-0160, also known as Heartbleed [20]. Heartbleed allows anyone to read the internal memory of the system running the vulnerable OpenSSL library. Figure 4 shows the result of running the Heartbleed exploit on the ESXI server.

Figure 4: Heartbleed exploit on ESXI server The exploit produces a large binary file that can be converted to readable text containing several kilobytes of memory data.

The server responds with a vulnerable heartbeat. The response is approximately 1KB and potentially not enough to gather any information. If enough queries are made, usernames and passwords, emails, and more can be extracted from the memory of the server. An intrusion detection system cannot differentiate the leaked heartbeat and a legitimate one because these queries do not leave any trace in the server logs. The server should be patched with the most recent version of the ESXI software.

OpenVAS also reported that one machine may be vulnerable to MS15-034 [25]. This is a vulnerability in the HTTP.sys library affecting Microsoft Internet Information Server (IIS). With a specially crafted HTTP header, an attacker is able to crash the server and cause it to occasionally leak its memory contents [26]. We can do this by setting the range property of the header in bytes such as,

curl -v 134.48.94.129 -h "Host: ms15034_test" -H "Range: bytes=X-18449744073709551615"

where X is a variable that we set based on the resource we are requesting. The largest number in the range is 2^{64}-1, which is the largest 64 bit number. Setting the first number to be greater than 0 but lower than the requested resource can leak memory. Setting it one byte less than the requested resource will cause the denial of service. The IIS welcome page is known to be 694 bytes, so a query that uses a range of 688 will leak approximately 1680Kb of memory. Setting the first range variable to be 693 will cause the IIS service to crash. ⌘ This vulnerability was fixed by applying the correct Windows Update to the machine. ⌘

In this section, we discussed looking for and confirming some vulnerabilities that exist within the GasDay network. Automated tools produce results that need to be confirmed through a manual assessment before being considered vulnerabilities. In the next section, we will discuss common vulnerabilities found within networks similar to GasDay and their status in the network.

4. Other Vulnerabilities

Attackers often follow the path of least resistance. They search for the easiest to exploit vulnerabilities and those that are most likely to work. Several vulnerabilities that have persisted for years despite multiple patches fixing them. This section discusses easy exploits that might exist in a network similar to GasDay's. This section also discusses exploits that would impact the integrity and confidentiality of customer data using knowledge of where the data is located.

4.1. Common Vulnerabilities

One of the common vulnerabilities found in legacy systems is MS08-067 [27]. This vulnerability affects Windows XP and Windows Vista devices. Windows XP is no longer supported as of April, 2014, but certain legacy systems still use it. There is a good chance that if a computer is still running XP, then it is not running a fully patched version of XP. MS08-067 affects all versions of XP and allows for the execution of code remotely. Even if the exploit fails, it has a chance of crashing the services on the

computer. Attempting to run the exploit on the XP machine in the lab fails and does not cause the services to crash.

Another set of common attacks are called null sessions. Windows can enable a user to connect via various protocols (SMB, RPC, etc.) using a null session - one with no username or password. If the null session is successful, then an attacker might be able to gain information based on the type of protocol used. Figure 5 shows the result of connecting to a computer using a null SMB session. We can see that the login was successful, but the information it returned was nothing more than what can be obtained via other methods (such as Nmap).

Figure 5: A null SMB session attack. The information given here is no more than what is acquired via a port scan. A successful SMB login would allow for FTP-like access to the computer.

A poor password policy can be worse than a poor network setup as it would allow users to remotely login to any system without the need for exploits. The GasDay administrator password is required to change regularly, meet a minimum length

requirement, and contain special characters. We analyzed a dataset of 42 million leaked passwords from sources including Facebook, Twitter, LinkedIn, and others. No GasDay usernames or passwords were found during this analysis. Passpal performed an analysis of 32.6 million passwords focusing on common characters and password structure. GasDay's passwords fall into the uncommon categories in both common characters and structure. We conclude that the combination of GasDay's uncommon password structure, complexity, and length give us confidence the password will not be brute forced.

4.2. Data Exploits

GasDay trains the models that forecast natural gas flow on a cluster system. Looking at the open ports on the cluster head node, port 27000 is open with the service name "flexlm0." This open port is the licensing server for the cluster head node and its connected worker nodes. Looking up "flexlm0" in Metasploit reveals a buffer overflow that targets several instances of the FlexNet licensing manager. This exploit would allow a remote user to execute code with the permissions of the logged in user. Attempting this specific exploit fails with multiple encoded payloads.

The GasDay software package that is delivered to our customers goes through a continuous review process. Any change in the code is peer reviewed before being accepted. Only the development teams have access to their respective code repositories. User accounts are pruned twice a year when the student developers leave (winter and spring semesters). Both the peer review platform and issue tracking platform are unavailable outside Marquette.

Although changes are peer reviewed and access to the code is controlled, it was discovered that one of the servers hosting code was accessible outside Marquette via a web page. Successfully exploiting this server would potentially allow an attacker to access the code. ⌘Although no vulnerabilities were discovered and attempting to brute force credentials resulted in an IP ban, GasDay decided that the best course of action was to block access to the server outside of Marquette. ⌘

5. Conclusion

In this chapter, we presented penetration testing as a method to evaluate the security and discover weaknesses within the GasDay network. GasDay is able to take advantage of the Marquette security controls already in place, which we have shown to be effective at detecting and stopping attacks that would otherwise go unidentified. Weaknesses that were discovered were corrected (noted by the ⌘ symbol). Penetration testing using a combination of manual analysis and automated tools were able to discover these weaknesses. These automated tools have also been configured to run automatically and report their results to make future testing easier.

We recommend that GasDay take the following actions to remedy existing vulnerabilities:

- Patch all existing assets with the most recent updates
- Remove or repurpose assets (software, passwords, hardware) that are no longer in use
- Continue using automated tools to catch vulnerabilities as they arrive

We recommend that GasDay consider the following actions to mitigate future vulnerabilities:

- Create an up-to-date inventory list and use automated tools to keep the list accurate with software and hardware changes.
- Before a new software tool is adopted or a hardware change is made, the risk added by the change should be understood and outlined with mitigation strategies.
- Implement an automated tool to manage software patches. This will help prevent computers from being unpatched due to their unknown location and status
- Evaluate internally hosted websites, login credentials, and network access using the principle of least privilege. Websites should not be external unless necessary. Network access and login credentials should be available only when necessary.
- Treat any non-standard interaction with a customer (testimony, certain presentations, etc.) the same as publicly releasing the data. Several items of GasDay are hosted on other web servers that contain personally identifiable information.

CHAPTER 3

Penetration Testing of GasDay Web: A Case Study

This chapter discusses finding vulnerabilities in a website, a group of web pages hosted on a server. Weaknesses are a result of a misconfigured server or coding mistakes by the developers. GasDay wants to deliver a new web product to our customers called GasDay Web. Creating a public website means that GasDay Web might become a target. We want to find potential vulnerabilities before someone else does. We present web application penetration testing as a method of identifying vulnerabilities within the code. We will step through the most common vulnerabilities, the tests for the vulnerabilities, and the GasDay Web response to testing. We also present additional tests for GasDay Web specific use cases. Throughout this chapter, the GasDay Web URL is represented as *gasdayweb.com*.

1. Web Application Penetration Testing

Web application penetration testing is the process of attempting to gain access to a website or its resources without valid credentials. This includes users gaining administrative privileges, unauthorized users gaining access, and valid users accessing other user's information [28]. Web application penetration testing is similar to network penetration testing because both mimic an attacker. The tester wants to find vulnerabilities in the system before the attacker. By finding the vulnerabilities first, the tester is able to fix the vulnerabilities before attackers have a chance to exploit them.

In 2013, at least 77% of all websites contained at least one vulnerability; in 2014, an estimated 76% of all websites contained at least one vulnerability [29], [30]. Over 90% of these vulnerabilities are the fault of the developers of the websites [31]. These vulnerabilities have been assembled into a Top 10 list that will help guide this assessment [32]. Many of these vulnerabilities allow an attacker to access information without knowledge of valid credentials. For GasDay and the customers of GasDay, data is extremely important to protect. Any weakness that might cause the loss of data could be worth millions of dollars.

We will use a tool called the Open Web Application Security Project Zed Application Proxy (ZAP) [33]. ZAP is an automated tool that provides a set of tools that support manual testing. It allows developers to test their website during development. The features of ZAP we use will be explained in the rest of this chapter.

The focus of this test is GasDay Web. GasDay Web will serve as a web platform for GasDay customers to be able to view their GasDay flow estimates and historical data. We want our customers to feel secure and comfortable transitioning to GasDay Web. We present this chapter as both a penetration test and an explanation of how penetration testing is integrated into the software development life cycle.

2. GasDay Web

In this section, we discuss GasDay Web and the third party services it uses. Customers can log into GasDay Web and view their GasDay flow estimates up to eight days in the future. GasDay Web also allows users to view historical data, similar days,

weather information, and more. We discuss Microsoft Azure, the cloud platform GasDay Web uses, and the ways that GasDay and our customers interact with Azure.

Microsoft Azure is a cloud computing platform that provides several solutions to its customers. GasDay uses three different Azure services: web app service, databases, and blob storage as shown in Figure 6. The web app service is the website for GasDay Web. Every GasDay customer has its own web service. These web services are logically separated so that no customer can access another customer's data. Each web service also uses its own database, which is the traditional GasDay database with additional tables to manage permissions for users. The blob storage is used to store images used by GasDay Web (such as loading icons, background images, etc.). Blob storage is independent of customer, so a single storage serves all GasDay web instances.

GasDay uses Azure to host the website and the services supporting the website. For most customers, the first interaction they have with GasDay Web is the login page (Figure 11) and the authentication of users.

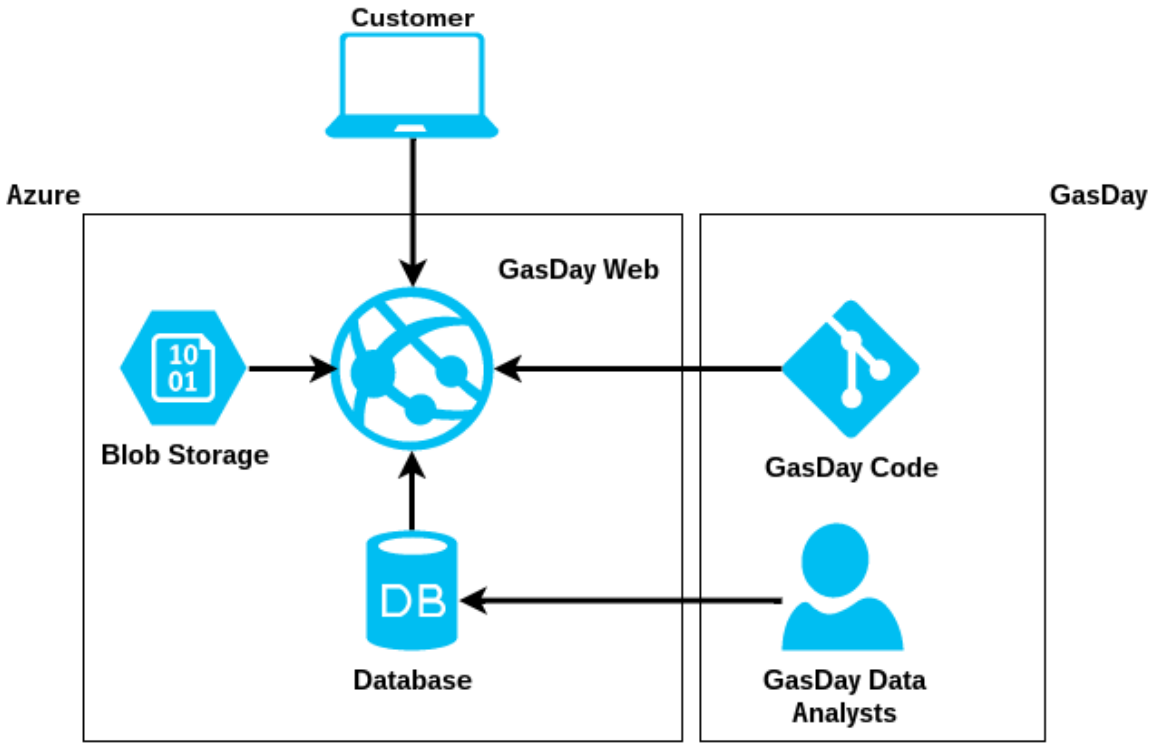

Figure 6: GasDay Web as a service. GasDay developers are able to push new changes directly to the website. GasDay analysts are able to access databases without modifying the web service.

3. Authentication

The opening page of GasDay Web is the login page (Figure 11). GasDay Web requires all users to log in with a username and password before they can view any information. The process of verifying that a user is who they claim to be is authentication. GasDay Web must have strong authentication guidelines to ensure that customers are the only ones who can access their data and that invalid users cannot bypass or guess GasDay Web's authentication. In this section, we discuss GasDay Web authentication bypasses and how GasDay Web protects valid authentication requests.

3.1. Bypass Authentication

When a user, Alice, attempts to log into GasDay Web, a message is sent to the Azure instance to validate her credentials. If Alice enters correct credentials, the server sends back an identification number that subsequent requests will use so that the server can identify Alice as valid. If an attacker, Bob, were able to intercept these messages, he would be able to log into GasDay Web with valid credentials. We test for this by deliberately intercepting the messages between the user Alice and the server.

```
POST http://gasdaytexasdemo.azurewebsites.net/ HTTP/1.1
Proxy-Connection: keep-alive
Content-Length: 27
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://gasdaytexasdemo.azurewebsites.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.73 Safari/537.36
Content-Type: application/x-www-form-urlencoded
DNT: 1
Referer: http://gasdaytexasdemo.azurewebsites.net/
Accept-Language: en-US,en;q=0.8
Cookie: ARRAffinity=6b11d2b1f9b5babe640c097cc322fa4270ea45ab58f4959881e9f4abf29115b6
Host: gasdaytexasdemo.azurewebsites.net

UserName=test&Password=test
```

Figure 7: The HTTP POST request for an example login

In Figure 7, we see that an example login sends a POST request to *http://gasdaytexasdemo.azurewebsites.net* over HTTP. This indicates that the credentials supplied are transmitted to the server in plain text. This would allow our malicious user, Bob, to sniff them off the network, or intercept them via a man-in-the-middle attack [34]. Bob would then have access to a username and password. Even if Bob is unable to intercept the initial login, Bob is able to intercept any communication between the user and GasDay Web, allowing him to access any customer data without needing to log in.

Bob is able to do this by using any network analyzer such as Wireshark [35]. To mitigate this, GasDay Web, should use HTTPS whenever sensitive information needs to be sent. GasDay Web should consider adopting HTTPS for every request and response.

If Bob were able to guess the ID the server sends back, he could create requests with the valid ID number and access information without logging in. Ten sequential login tokens were generated and analyzed for patterns. Each ID contained a unique string of characters; no pattern could be identified. GasDay Web uses a 288 digit hex string for each ID. This number is randomly generated, so it is improbable that Bob could predict it.

Another way to bypass the authentication schema is to access cached or historical versions of the webpage. These mechanisms are for Alice's convenience so that she can view previously displayed information without having to download it again. However, if confidential information were on these pages and stored as a cached/historical item, it could be viewed without visiting the webpage. To test this, we log into GasDay Web as a valid user and browse through the Summary Page for a few forecasts. Next, we logoff and verify that the browser no longer has an authentication token. By pressing the "Back" button on the browser, we are able to view every page that was visited as a valid user without needing to login. If Alice were to use GasDay Web and step away from the computer after logging out, anyone would be able to use the back button to view everything that Alice did. To fix this, GasDay Web should implement a no-cache, no-store, forced revalidation cache control [28]. This instructs the browser not to cache any web page and prompt Alice to re-enter credentials to reload the page.

3.2. Denial of Service

In a denial of service attack, Bob prevents Alice and other users from accessing the website. This usually is done by Bob sending requests to the server so that the server is busy trying to process invalid requests from Bob instead of valid requests from Alice. For GasDay Web, we are concerned about packet flood denial of service attacks.

Packet floods are known colloquially as denial of service or distributed denial of service attacks (DoS and DDoS), in which Bob sends large amounts of traffic-consuming resources to the server until it goes offline. Packet floods only deny others from using the service; they cannot access data that requires authorization. GasDay Web uses the technologies from Azure to protect customers from packet floods. Azure detects several kinds of packet floods (Layer 3 ICMP, Layer 4 SYN flood) and is able to rate-limit or block IP addresses that exceed the threshold for a specific type of attack [36].

In this section, we looked at how GasDay Web is accessed. GasDay Web authenticates users using IDs that cannot be predicted nor easily guessed. Any user wishing to access GasDay Web needs to be authenticated before any page beyond the login page is visible. GasDay Web takes advantage of the Azure cloud detection and prevention systems to prevent denial of service attacks. In the next, section we look at how GasDay Web differentiates valid users.

4. Sessions

GasDay Web differentiates between users using sessions [28]. HTTP is a stateless protocol, which means that the web server responds to requests from users without linking users together. Sessions are identification numbers associated with users so that the server is able to track individual users making multiple requests. Session ID numbers are different from the authentication number discussed in Section 3. Authentication numbers ensure that a user is authentic and that the user is only able to see information relevant to them (non-administrators cannot view administrative pages), while sessions keep track of a user's interaction state with the website (such as the view date on the Summary page). In this section, we discuss how GasDay Web stores session ID numbers and how we protect against attacks that target these IDs.

4.1. Cookies

GasDay Web stores the authentication and session IDs using cookies named ASPXAUTH and Session. Cookies are pieces of data that a website uses to keep track of information about the user browsing the website [37]. Cookies keep track of a shopping cart, authenticated users, or previously entered information on a page. They are returned to the server whenever the user makes a request (such as clicking a button) to help the server determine where the user should go next. However, a cookie represents a user's identity, so a stolen cookie represents a stolen identity [38].

Cookies are created and set when the server tells the user's browser to keep track of some data. GasDay Web creates an authentication and a session cookie when a user

supplies a valid username and password. GasDay creates these cookies with the HttpOnly attributes to prevent the cookie from being modified by scripts.

4.2. Session Fixation and Session Puzzling

The application is fixated on a specific session when the application does not renew the session cookie after a user successfully authenticates. Bob, an attacker, could use this by logging into the website and then sending his cookie to Alice, thereby stealing her session [28]. We test this by logging into the website with valid credentials and receiving a cookie as seen in Figure 8.

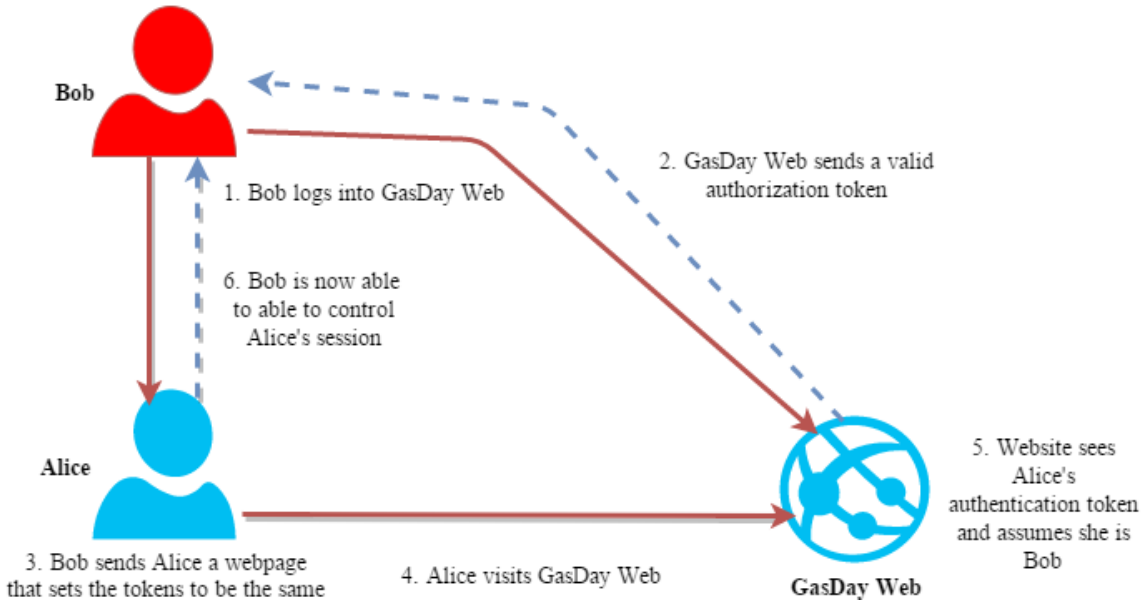

Figure 8: An example use case for session fixation. Bob is able to control what Alice sees on GasDay Web. Any action by Alice will also be seen by Bob.

We can now take this cookie and send it to Alice via a simple web page that sets the cookie via JavaScript and redirects the user to GasDay Web. Alice will not need to

login and will see the Summary page for GasDay Web. Alice now is using our controlled session. If we reverse this by acquiring Alice's session token, we would be able to hijack Alice's session.

This could be prevented by using an additional token required for any state changing operation (such as changing the view date or creating a user) [39]. These tokens are also known as CSRF tokens because they also prevent another attack called cross site request forgery. These tokens are randomly generated and inserted into any URL on a webpage before being sent to Alice. Every time Alice clicks on a URL, the token is validated. If the token is different from what is expected, the server rejects the request and can invalidate the original authentication token. If Bob were to visit GasDay Web at the same time as Alice, his CSRF tokens will be different than Alice's, preventing Bob from viewing any information.

In Section 3.1 we showed that GasDay sends cookies over HTTP instead of HTTPS. Instead of fixating our session, we can set our own cookie to be one that we intercept from Alice. By visiting GasDay Web after setting our own cookie to Alice's, we are able to bypass the login and browse GasDay Web as Alice. We do not have a valid username and password, and our session would be terminated if Alice logged out manually. However, if Alice were an administrator, we would be able to create a new account with which we could later log in.

This could be prevented by sending the cookie with the "Secure" attribute to ensure the cookie is only sent using HTTPS. This prevents Bob from intercepting the token. The "Secure" attribute also would prevent an attacker from injecting code into Alice's browser that would force the browser to resend the token.

In this section, we looked at how GasDay Web keeps track of different user states. These user states could be hijacked if our malicious actor, Bob, were able to gain access to a token. GasDay Web should send every response with HTTPS to prevent Bob from intercepting the token. GasDay Web also should use the “Secure” attribute for cookies preventing the cookies from being read by external tools. These session attacks do not transfer between web instances, keeping GasDay customers safe from each other. In the next section, we will look at how GasDay Web authorizes different users to access the resources within each web instance. We will test if we can gain access to these resources as an unauthorized user.

5. Authorization

Authorization controls which users are able to view resources such as administrative pages or account information. Authorization comes after a user is authenticated. These tests are performed as a non-administrator attempting to use resources that they should not be able to authorize. First, we attempt to bypass authorization by trying to create or delete accounts as a user as shown in Figure 9. This action requires administrative privileges that our test account does not have. Visiting the settings page, we see settings that only apply to a non-administrator. We can craft a request to perform administrative actions (such as deleting a user). The server responds by redirecting our test account back to the login page. We verify that no user is deleted by viewing a list of usernames and finding our test account still active.

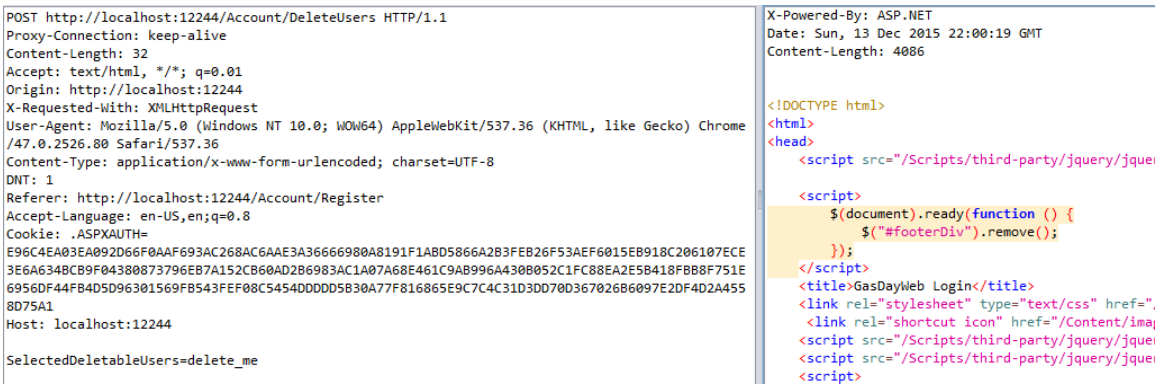

Figure 9: GasDay Web redirects an unauthorized user if they attempt to delete an account.

This same test is repeated for creating new accounts and changing account passwords. In both tests, the server redirects our test account to the login page without creating accounts or changing passwords. From this we can conclude that unauthorized users are unable to perform actions that require authorization such as changing settings or deleting accounts.

We also attempt to break out of the web application by attempting to access files not intended to be part of the website. A website usually confines all users to a single directory (or subdirectories) within a file system. By breaking out of the application, we would be able to escape this directory and access other parts of the file system. Resources that the website requires, such as images and audio files, or even sensitive server administration information, are stored within this directory. If the application did not validate requests for these images, we could be able to craft our own request to access files that are not part of the web service. For testing, we place a text file *loot.txt* one

directory higher than the web service. We try to access this file by manipulating requests as seen in Figure 10. We first try accessing the file via a URL such as

http://gasdayweb.com/../loot.txt.

The website blocks the request due to the double dot notation. By encoding the '.' to 2e (UTF-8), we change the URL to

http://gasdayweb.com/%2e%2e/loot%2etxt,

attempting to evade the website filter. We repeat the encoding for the entire URL and with different encodings. In every case, the server responds with an access denied page. The server responds with access denied if the requested resource does not exist (such as loot2.txt).

Figure 10: GasDay Web responds with a forbidden error when a request is made

After some manual tests with knowledge of the filesystem, we can use one of ZAP's features called active scan, an automated tool that sends requests to any input field set by the tester. We set up an active scan that attempts to traverse the underlying file system and local include files in requests. ZAP makes several thousand requests similar to the ones shown in Figure 9 and Figure 10. In several requests, ZAP made a request that

caused the server to return back a 500 internal server error. Internal server errors occur when the code running the website breaks. If the code breaks, the server might not be able to process that specific request, or it might have corrupted the entire website requiring a restart of the web service. For our results, from most errors, GasDay Web was able to recover. However, certain inputs crashed the entire website, requiring a restart. An example of one of these inputs is

%22%3E%3C%21--%23EXEC+cmd%3D%22dir+%5C%22--%3E%3C

When decoded from UTF-8, the command looks like,

"><!--#EXEC cmd="dir \"--><

which attempts to list the files in a directory on a Windows computer. These errors were the result of the Azure service (not the request filter) preventing the listing of files. By better capturing these errors and using another layer of request filtering, these crashes would be better mitigated.

In this section, we looked at how GasDay Web authorizes requests for valid and invalid resources on the website. However, a URL is not our only injection point. Any input field can be used to attack the website. In the next section, we will look at how GasDay Web validates other input fields.

6. Input Validation

The most common type of web vulnerability comes from input fields [32]. Figure 11 shows an example of input fields. These weaknesses are a result of not properly validating input data before using it. The first rule of dealing with handling external data

is “all input is evil” [40]. This section describes the different types of input and the tests to verify that GasDay Web properly validates all data before using it. Azure performs a second layer of validation for requests that interface with another service on Azure (such as the database). This layer of validation is to protect Azure from malicious requests that GasDay Web misses.

Figure 11: The login page for GasDay Web. The Username, Password, and Remember Me fields are all input fields subject to validation.

6.1. Cross Site Scripting

The first type of input we submit to the website are scripts. Cross site scripting (XSS) attacks occur when a malicious user, Bob, sends his own script to Alice through the web application. XSS allows an attacker to manipulate an input field to produce an

output that is under the attacker's control. This output could be stealing Alice's session or hijacking her webpage. XSS attacks are categorized into two methods: reflected and stored. Reflected XSS attacks, seen in Figure 12, bounce off the web server in the form of an error message, search result, redirect, or any response that the server sends back to the user. Stored XSS attacks are saved on the server and are sent to Alice whenever she requests the resource (such as an embedded script in an image that is displayed on a webpage).

6.1.1. Reflected Cross Site Scripting

In a reflected XSS attack, Bob sends his script to Alice through a URL on which Alice clicks. This sends a request to GasDay Web, which responds by sending a webpage containing Bob's script. Alice's browser trusts the webpage, and Bob's script is executed. Bob's URL does not have to exist on the website; his script could be executed as part of the error message returned by the website. Bob could also attach his script to any predefined value on the webpage displayed (e.g., radio buttons or pre-selected checkboxes).

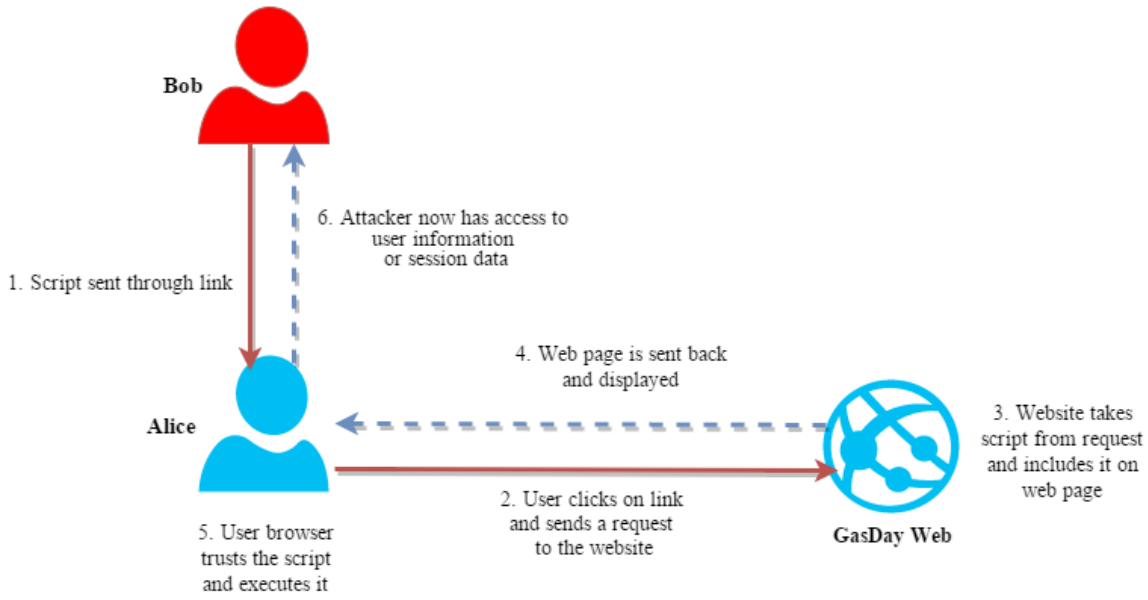

Figure 12: An example use case of a reflected cross site scripting attack. The attacker, Bob, does not directly interact with the website but is able to gain information about Alice.

We test for reflected XSS by injecting code into the browser's address bar. Throughout our tests, we use the JavaScript code *alert('Hello!')*. A successful XSS attack would display a popup that contains the text *Hello!*.

Our first test is to inject the scripts directly. The scripts we inject are not encoded and do not attempt to evade any filters the website might have. The first attempt is directly in the URL such as

http://gasdayweb.com/<script>alert('Hello!')</script>.

The website redirects to the home page of GasDay Web and does not display the *Hello!* message.

We also try encoding the URL using different formats. Encoding the script keeps the original functionality, but allows the script to evade certain filters. For example, we can encode the letter *a* to hexadecimal *&#x61*, decimal *&#97* or octal *0141*. Our previous request encoded to UTF8 now takes the form

http://gasdayweb.com/%3cscript%3Ealert%28%u2018Hello%21%u2019%29%3C/script%3E

This request also redirects to the home page of GasDay Web indicating that GasDay Web is successfully filtering out these requests.

We try evading filters using lesser known facts about HTML and JavaScript and combining previous approaches. For example, we include our script as part of an image and embed an encoded tab to split up the text changing

<script>alert('Hello!')</script>

*<IMG SRC="javasc	ript:alert('Hello!');">*

GasDay Web will redirect these requests to the home page of GasDay Web suggesting that GasDay Web is well defended against most reflected XSS attacks.

A code review reveals that GasDay Web uses ASP.NET request validation, which automatically blocks suspicious requests such as the two listed above. While an attacker could not review the code that GasDay Web uses, we can review the code to find vulnerabilities. Therefore, we need to evade the ASP request filter. The request validation filter by default blocks any request that contains a "<" followed by a character and a subset of special characters. Using this knowledge, we can encode our request using Unicode to bypass this filter. Our Hello! request now looks like:

%uff1cscript%uff1ealert('Hello!');%uff1c/script%uff1e .

By submitting this request into the username field, we see that we successfully evade the request filter as GasDay Web attempts to search for a user with the name of our *Hello!* request.

ZAP helps by creating approximately 12,000 unique encodings and permutations of our *Hello!* request. None of these requests the *Hello!* message to display, suggesting that these requests were rejected or handled by GasDay Web. From this, we can conclude that GasDay Web is well defended against reflected XSS attacks.

6.1.2. Stored Cross Site Scripting

Stored XSS attacks are saved on the server and are sent to Alice whenever she requests the resource (such as an embedded script in an image that is displayed on a webpage). The end result of these attacks is similar to reflected XSS, except that these attacks will impact Alice every time she accesses the resource. For GasDay Web, there are only two places where data is retrieved: when an account is created/modified and when forecast data is displayed to Alice. Both of these operations are validated before being stored into the database. Therefore, these types of XSS attacks do not succeed in GasDay Web.

6.2. SQL Injections

SQL injections are similar to cross site scripting. SQL injections also send custom data fragments to the server. These injections are targeted at the underlying database server directly instead of the users of the website. Other types of injections are possible, but are omitted because the GasDay Web is missing the component to be injected (for example, SMTP injection requires an email server, and buffer overflows require .NET to use unsafe code blocks, neither of which are found in GasDay Web).

SQL injections consist of entering a partial or complete SQL query via a request to view sensitive information in the database, modify data, or bypass authorization. SQL injection works when queries to the database that require user input are not validated before they are executed. For example, imagine a login page that grants access if a username and password match one in a database. The query might look like

*select * from users where username = $user and password = $pass*

If Bob were to enter his password as "*1' OR '1' == '1"* (preserving the single quotes and spaces), the database would search for a user whose password was 1 or a logical true. As long as Bob had a valid username, the system would log him in regardless of his password.

We test for this by identifying any SQL queries present in GasDay Web that require user input and attempting to exploit the input. GasDay Web interacts with the database on login and to grab data when Alice changes her view date. By putting in several SQL queries into the username and password field, we attempt to bypass the login page. Figure 8 shows the response from GasDay Web when a valid username is entered

with the encoded *1==1* query. The server does not set any authorization cookies, nor does it display any error. We again use ZAP to generate hundreds of these queries and inject them in the username, password, and remember me fields. All of these requests respond similar to the request in Figure 13. From this we can conclude that GasDay Web is well defended against SQL injection attacks.

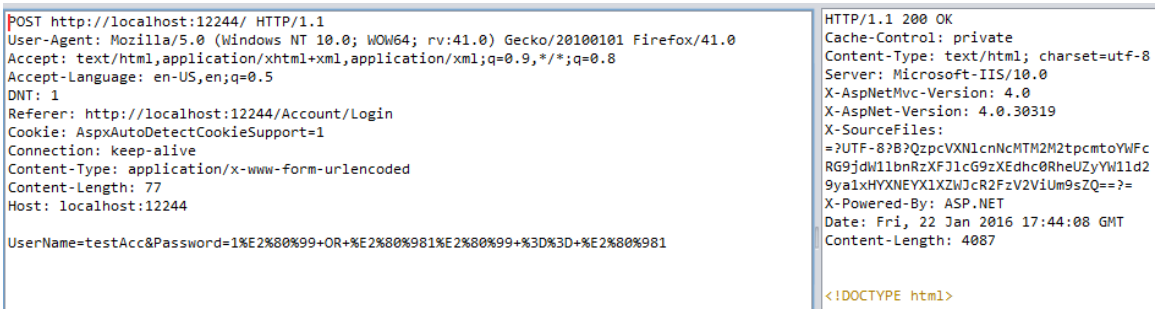

Figure 13: A SQL escaped character string is used as the password to GasDay Web. GasDay Web responds as if a user typed credentials incorrectly.

6.3. HTTP Request Tampering

When Alice performs an action on the website, an HTTP request is generated to satisfy the action. A request usually takes the form of a GET (getting a resource from the server) or POST (posting a resource to the server). However, there are several other HTTP verbs that can be used. The results of these verbs may produce unintended consequences. For example, the application may not let Alice (a non-admin) delete a user via POST, but if Alice tries to delete a user via PUT, the server might let her. We test for this by using every HTTP verb for groups of requests Alice can do on GasDay Web.

We try sending every verb to two specific URLs: one to retrieve data from the database and one that requires a user to be authorized to visit. For the URL that retrieves data, the server responds with a 404 page stating that the page was not found. This is a good response from GasDay Web because it does not give any information to Alice. Only when Alice sends the correct verb will she receive data. For the page that requires authorization, the server responds with a redirect to the login page. GasDay Web first checks if Alice is authorized to view the page. If she is not authorized, GasDay Web redirects her to the login page. If Alice were authorized, the server would respond as it did in Figure 14. From this we can conclude that GasDay Web is well defended against request tampering.

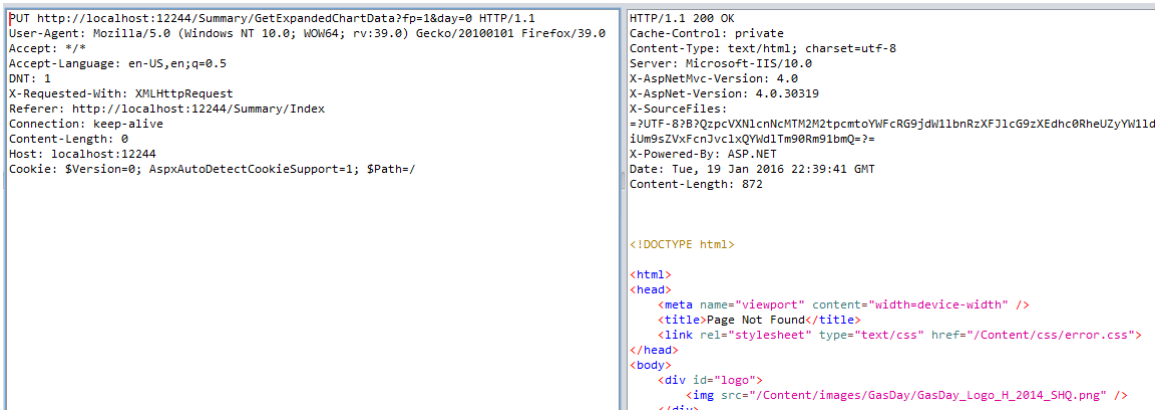

Figure 14: Sending a PUT Verb to retrieve data from the database results in a 404 page not found response from the server

In this section, we discussed validating all input that comes into GasDay Web. We injected several custom requests into every input field that GasDay Web currently has. We also injected custom requests into locations that GasDay Web does not normally

receive requests. We were unable to exploit GasDay Web using the most common vulnerabilities [32] using automated tools. This suggests GasDay Web is well defended against cross site scripting and request tampering.

In the next section, we will use the source code to search for vulnerabilities that automated tools can miss.

7. Code Review

The previously mentioned vulnerabilities can be discovered via an automated tool by a determined individual. There are often vulnerabilities that cannot be discovered with an automated tool and must be discovered by manually inspecting the code [28]. Reviewing the code is something an external attacker is not able to do, but it allows a tester to find vulnerabilities that an external attacker would have a hard time discovering through trial and error.

An easy thing to look for are functions that are public that should not be. We find one function titled *test_deleteUser*. The function exists to help validate deleting a user. However, it unintentionally allows anyone to call this function by going to *gasdayweb.com/Account/test_deleteUser*. We can further pass in the name of an account we want to delete such as

gasdayweb.com/Account/test_deleteUser?name=alice_account,

and *alice_account* will be deleted without requiring any authorization. This function should be removed, and a different tactic for testing (such as exposing private functions to the test class) should be used.

We also can look for functions that do not require authorization but should require authorization. For example, we find that the *GetAlert* function shown in Figure 15 does not require authorization.

```
[HttpGet]
3 references |          57 days ago | 7 authors, 14 changes
public ActionResult GetAlert(int fp, int day) {
    var fcstPt = ApplicationInfo.ForecastPoints.Single(f => f.Identifier == fp);
    var date = AppState.ViewDate.AddDays(day);
    var alert = AlertList.AsReadOnly().Where(a => a.Region.Equals(fcstPt) && a.Date
    try {
        return this.JsonNet(new { title = alert.DefaultMessage, message = alert.Mess
    }
    catch (NullReferenceException) {
        throw new ArgumentException("There is no alert for forecast point " + fp + "
    }
}
```

Figure 15: A GasDay Web function that will get any alerts for a forecast point. It does not require a user to be authorized explicitly, but assumes that any user that calls it is authorized.

This function gets any GasDay alerts for a specific forecast point (such as an unusual weather value). Attempting to call this function as an unauthorized user causes the web service to stop responding. The web service is similar to a program running on a computer. When the program stops responding, it must be manually restarted. When GasDay Web stops responding, or starts responding with invalid data, it must be manually restarted by a developer. We find several other functions to retrieve data from the database that also cause web service to stop responding when called as an unauthorized user. These functions could be executed non-maliciously if Alice were to become unauthorized (by having her session timeout, for example) and then attempt to send the request for data (by clicking on a forecast, for example). In this case, Alice

would unknowingly cause GasDay Web to stop responding, requiring a full restart of the web instance.

Code reviews also allow us to find input fields that are not properly checked before being used. Figure 16 shows an example of a function that uses input without validating it. This function is called directly from a browser to populate charts that appear on GasDay Web. Integers in C# are between -2,147,483,648 and 2,147,483,647 (-2^{31} and 2^{31}-1) [41].

```
public ActionResult GetExpandedChartData(int fp, int day) {
    Chart[] charts = model.GetExpandedCharts(fp, day);
    var expandedCharts = charts.Select(c => c.GenerateExpandedChart()).ToArray();
    return this.JsonNet(expandedCharts, JsonRequestBehavior.AllowGet);
}
```

Figure 16: A function in GasDay Web that uses input without validating it to be correct. Entering numbers greater than 231 or non-numbers results in undefined behavior.

By sending numbers outside of this range, we produce an undefined behavior. For GasDay Web, this produces an exception that stops the web service. We can also produce the same result by sending non-numbers (such as the letter ‘a’). These requests are unlikely to be sent by Alice and could not be sent by Alice if she was unauthorized. However, if Bob logged into GasDay Web and created a request such as:

gasdayweb.com/Summary/GetExpandedChartData?fp=a&day=999999999999,

he would be able to cause GasDay Web to stop responding. Each function that takes input from the client should validate it before it is used.

The results from this section demonstrate that automated tools cannot catch every vulnerability. By integrating code reviews into the development process, these vulnerabilities can easily be identified if the development team knows what the vulnerabilities look like. If these vulnerabilities are caught in the development phase, then they will be patched before they reach the live website.

8. Conclusion

In this chapter, we presented website penetration testing as a method to discover vulnerabilities in GasDay Web. GasDay Web is able to take advantage of existing features within ASP.NET which have been shown to be effective at detecting and preventing malicious requests. However, we noted several places where these features could be evaded or where they were absent from certain sections of GasDay Web's code. While automated tools were able to discover some of these vulnerabilities, manual analysis yielded more results and allowed us to target the automated tools.

We note the following areas where GasDay Web performed well during these tests:

- Cross site scripting attacks and SQL injections were unsuccessful
- Denial of service attacks are mitigated by Azure
- Cross site request forgery, request tampering, and request splitting were unsuccessful
- Authentication tokens are extremely difficult to guess

We recommend GasDay takes the following actions to fix existing vulnerabilities:

- Force HTTPS for every interaction with GasDay Web
- Sanitize any input (using a third party tool) before using it
- Add an additional layer to catch any exceptions that might be thrown to protect the server

We recommend GasDay take the following actions to mitigate future vulnerabilities:

- Train developers in secure coding styles and secure code reviews
- Integrate malicious inputs as unit tests within GasDay Web code
- Integrate an automated tool, such as ZAP, as part of the build process to search for vulnerabilities
- Investigate a way to detect when a web instance requires a restart (potentially automatically) to reduce the impact of successful exploits

CHAPTER 4

Applying the NIST Cybersecurity Framework to GasDay

This chapter discusses the National Institute of Standards and Technology (NIST) Cybersecurity Framework and how it has been adapted for use at GasDay. Cybersecurity frameworks are a collection of standards and techniques that an organization can use to help guide cybersecurity decisions. Frameworks are not intended to be a step-by-step guide; instead they help an organization, such as GasDay, prioritize risks and mitigate anticipated future threats. The NIST Cybersecurity Framework (referred to as the Framework) is a collection of best practices created for critical infrastructure of the United States, such as the energy industry. We present the Framework and how it has been adapted to work within GasDay. We represent that GasDay implementation of the Framework as the GasDay Cybersecurity Framework. We present our initial assessment of GasDay in tandem with a maturity model.

1. NIST-Cybersecurity Framework

The National Institute of Standards and Technology (NIST) is an agency of the United States Department of Commerce. NIST is known primarily for creating standards for all facets of science and technology. NIST has created standards for measurements and weights, voting machines, antifreeze, and more. Within the context of cybersecurity, NIST developed the advanced encryption standard (AES), random number generators,

and the Cybersecurity Framework. NIST standards are considered best practice, often developed with the feedback of a public process.

The NIST Cybersecurity Framework was created as a result of Executive Order 13636, issued in 2013, to "enhance the security and resilience of the [United States] Nation's critical infrastructure" [42]. The Framework serves as a tool to enable organizations to describe their current and target state for cybersecurity, identify the gaps between these states, and communicate with stakeholders about risk [43]. The Framework is not intended to be a full cybersecurity program that manages risk (such as ISO 31000 [44], or NIST 800-37 [45]), but is instead to compliment them.

The Department of Energy recommends that every energy organization consider adopting the Framework. To aid in adoption, the Department of Energy released their own set of complementary guides to the Framework targeted at different energy providers [46]–[48]. Several GasDay customers submitted comments on the initial draft of the NIST Framework, while other customers have requested GasDay to demonstrate compliance with their own cybersecurity frameworks. GasDay decided to review the NIST Framework for adoption and has decided to implement the Framework alongside a Department of Energy complimentary guide.

The NIST Framework is composed of three parts: the Core, Implementation Tiers, and Profiles. The Core is a set of activities and desired outcomes that are common across all sectors of industry. It includes items such as identifying potential threats and protecting different types of data. Implementation Tiers describe how an organization views each category from the Core. Each Tier describes an increasing amount of sophistication of the cybersecurity practices. The Profiles describes how well an

organization's actual practices align with their targeted goals. The Profiles also contain a prioritization of which Core Categories are most important to GasDay and which implementation tiers are most cost effective to change.

1.1. NIST Framework Core

The Framework Core is composed of activities that are common across industries. It identifies actions that are helpful in mitigating risk in cybersecurity. These actions are grouped into five Functions: Identify, Protect, Detect, Respond, and Recover. Each Function is a high level organization of cybersecurity activities. Within each Function, there are Categories and Subcategories. Categories are a collection of activities within a Function. For example, a Category under the Protect Function is "Awareness and Training." Within each Category are Subcategories which are specific controls or actions that contribute to the category. For "Awareness and Training," an example Subcategory is "privileged users understand roles and responsibilities." Subcategories in the Framework also have a link to several other cybersecurity and risk management frameworks, which provide context on their implementation. Each Subcategory can have several controls that are evaluated using Implementation Tiers.

1.2. NIST Framework Implementation Tiers

The Department of Energy provides their own resource adapted for different areas in the energy industry. These resources are the Oil and Natural Gas Cybersecurity Capability Maturity Model (ONG-C2M2) and the Electricity Subsector Cybersecurity

Capability Maturity Model (ES-C2M2). The C2M2 allows GasDay to evaluate Tiers based on a standard used by other subsectors in the energy industry. The Tiers within the C2M2 range from maturity indicator level 0 (MIL0) to MIL3.

Each maturity level represents the degree of maturity for a specific practice. MIL0 represents no maturity. It could represent a practice that does not exist, or that the MIL1 goals are not met. MIL1 activities are performed in an ad-hoc manner. In this context, ad-hoc refers to activities depending on experience or initiative rather than a documented course of action. Within an organization, MIL1 activities can have varying amounts success depending on the experience of the practitioners. MIL2 represents an initial institutionalization of activities. Practices are documented, stakeholders are informed, standards guide implementation, and resources are provided to support cybersecurity. MIL2 performance is more stable and sustainable over time. MIL3 represents an organized and adaptive practice. MIL3 expands on MIL2 by reviewing activities, training personnel, and adapting the cybersecurity plan according to business needs. Each Subcategory of the Framework has a maturity indicator level. This allows GasDay to choose selectively which Subcategories are most important. The overall measure of Implementation Tiers is seen in the NIST Framework Profile.

1.3. NIST Framework Profile

The Framework Profile is the measure of alignment of guidelines of GasDay to the Framework. It represents the collective results of the Core and Implementation Tiers. GasDay can also build a Target Profile, which represents the ideal state for GasDay, given the constraints of the business (time, staff, and money). The Target Profile also can

be motivated by attack patterns that we identify now and those trends we anticipate in the future. GasDay can prioritize the gaps between the Current Profile and the Target Profile. The prioritization can be on specific Subcategories instead of an overall Implementation Tier.

Sections 2-6 consider each Function in the GasDay Framework. We discuss the major points for the Categories and Subcategories and we identify the Implementation Tier and present the Current Profile for that Function. Section 7 summarizes the Current Profile and suggests recommendations for GasDay to prioritize a target profile.

2. Identify Function

The Identify Function is focused primarily on the understanding of systems, assets, and data that are important to GasDay. By understanding the resources important to GasDay, they can be protected and recovered. The Categories and Subcategories focus on keeping up-to-date catalogues of resource configurations, regulatory requirements, and threat profiles. For a high maturity level, we should outline clearly all of GasDay's information assets and those assets required to achieve our goal of forecasting natural gas. We also should communicate our dependencies (tools and data we receive from a supplier) and the customers that depend on GasDay. A clear measure of risk assessment and risk management should be present, although evaluating these will come in a later section of the Framework.

One set of Subcategories deals with identifying the physical and electronic assets of GasDay, including data, personnel, software, and their requirements to keep functioning at operational capacity. We found several software configurations and virtual machines (VMs) missing from inventories. There is currently no way of tracking when VMs or a software configuration is added, removed, or changed. We also found that this problem extends to physical configurations. While systems were put in place to monitor and identify vulnerabilities, several identified systems are not present in inventories or should not be present within GasDay.

The lowest maturity levels come from a lack of written risk management strategies. Before the start of this work, there was no method to identify vulnerabilities or evaluate potential threats to GasDay. As a part of this thesis, automated vulnerability detection was implemented with a scoring system for each vulnerability. However, this is only one subcategory of the risk management portion of the Identify Function.

GasDay demonstrates a high level of maturity dealing with regulatory requirements. GasDay is a member of Marquette University and hires student workers; we are subject to FERPA regulations. FERPA training is mandatory for every employee.

GasDay also demonstrates maturity by cataloging our dependencies to deliver software to our customers. Each dependency that GasDay requires to deliver software is well documented in an easily accessible location. This documentation is also constantly updated whenever a customer wishes to change their configuration.

The overall maturity levels for Identify can be seen in Figure 17**Error! Reference source not found.**. While GasDay had several items contribute to maturity levels, ultimately out-of-date information and the lack of

documentation reduced the overall maturity level of several subcategories. The overall maturity level for Identify is scored as MIL1. This indicates that the practices are in place for most Categories, but the practices are primarily ad-hoc. For GasDay to become a MIL2, the practices should be well documented, and a risk management framework should be used to guide the practices.

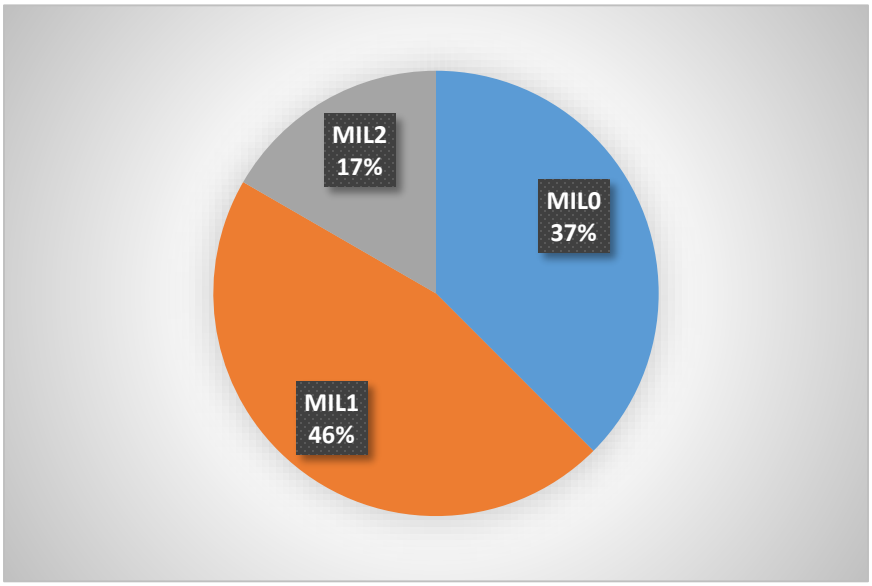

Figure 17: Maturity level breakdown for the NIST Identify Function

In this section, we considered the NIST Identify Function. GasDay's practices are largely ad-hoc and lack adequate documentation. We identified Subcategories that have improved over the past year and others that are worse. In the next section, we consider the Protect Function, which supports the ability to mitigate potential cybersecurity events. While Protect is a different Function from Identify, we will see that there are certain controls that are common to both Functions, but have different interpretations.

3. Protect Function

Protection ensures that GasDay can continue to deliver the software product while limiting or suppressing the impact of a cybersecurity event. These cybersecurity events can be internal (such as a disgruntled employee deleting data), external (a malicious individual attacking GasDay), or accidental (an employee unknowingly modifying code or data). Categories and Subcategories involve ensuring that each of these threats are mitigated through change policies, training, and the least privilege principle, which states that every asset in a system (users, software, administrators, etc.) should be able to access only the information required, and nothing more [49]. As we will see, GasDay starts with least privilege but is prone to privilege creep (assets acquiring privileges over time without losing old privileges).

GasDay demonstrates a MIL1 level of maturity with controlling access to users. Critical data is accessible to only to those that require access. Every individual is also trained about the confidentiality of the data they access. Where GasDay suffers is the re-use and availability of privileged accounts. Many administrative passwords are posted on the internal wiki, and several unlisted administrative accounts use the same passwords listed on this page. Due to this, integrity checking software and configuration changes can be circumvented or otherwise untrackable. This results in Subcategories that would score at a MIL2 or MIL3 to be reduced to MIL1. The availability of administrative passwords also causes many systems that would otherwise follow the principle of least privilege to be questioned. For example, the GasDay ESXI server hosts several virtual machines (VMs) used for testing the software before delivery. The ESXI server tracks

which users log in and out, but the administrative password is posted on the internal wiki. Any user would be able to log in as the administrator and add/remove/modify VMs. The log would attribute the actions to the administrator with no other method of tracking the actual user (such as the IP of the computer that logged in).

Several response plans or protection processes exist, but are updated reactively to situations; they are only updated when an incident occurs. This causes the maturity levels for the respective Subcategories to be capped at MIL1 or MIL2. In addition, documentation exists in multiple places with each version differing. There are several recorded instances where an incorrect procedure was (unknowingly) followed as a result of using the wrong documentation. When wrong documentation is discovered, it is not always updated to reflect the correct procedure.

The Subcategory for remote maintenance is not scored because GasDay does not perform any remote maintenance. This helps to ensure that GasDay is focused on only improving areas that are currently in use.

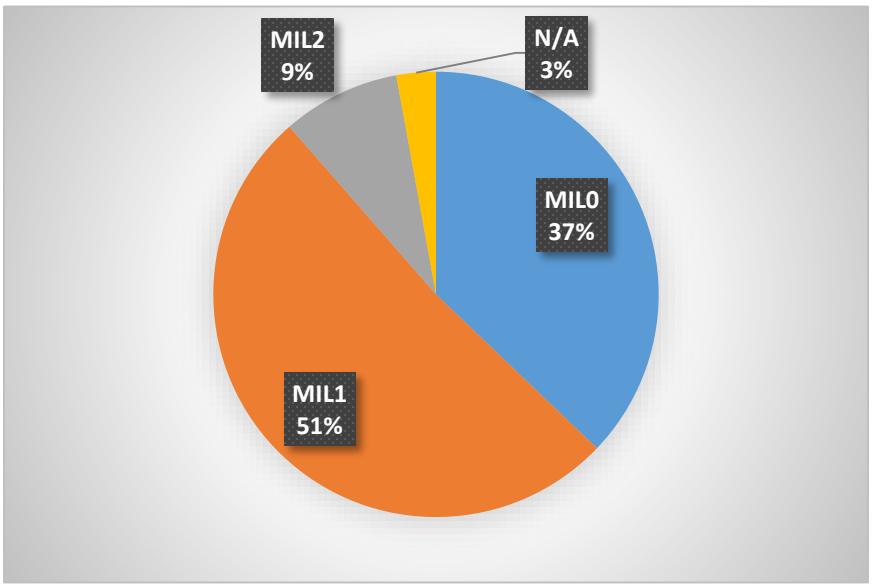

Figure 18: Maturity levels for the Protect Function. Subcategories that GasDay does not consider are marked by N/A

The overall maturity level for Protect is scored as MIL1. In Figure 18, we see the majority of Subcategories are scored as MIL0 or MIL1. GasDay can raise the maturity level of four subcategories by documenting the cybersecurity responsibilities of the leadership positions. GasDay can also raise the maturity levels of five Subcategories by periodically reviewing the access levels of identities. These action items are inexpensive to implement. Reviewing access levels is done for individuals who leave the lab. By additionally reviewing access levels for software, promoted and transferred employees, and external processes, GasDay would be better able to combat privilege creep.

Currently, GasDay is largely unaware of the cybersecurity threats facing the lab. While certain actions within Subcategories are performed (such as managing user identities), these actions would be much more efficient and effective if the threat landscape were clearly defined with a threat profile. A threat profile outlines all possible

threats that GasDay faces. In addition to identifying the threats, the vulnerabilities that the threat can exploit and the risk the threat creates should be identified. By identifying threats, GasDay would be able to prioritize and evaluate how different actions impact the threat landscape.

In this section, we discussed the controls associated with protecting the assets of GasDay. As we saw in Section 2, the primary problem is confusing and the absence of documentation. In addition, the undisciplined use of administrator passwords makes it difficult to verify and track changes made. The lack of a well-established threat profile makes it difficult for specific Subcategories to be effective. In the next section, we will see specifically how the lack of a threat profile impairs the ability to detect cybersecurity events.

4. Detect Function

The Detect Function evaluates the ability of GasDay to identify a cybersecurity event. The ability to discover a cybersecurity event should be within a (GasDay defined) timely window. Simply being able to detect a cybersecurity event represents a lower level of maturity than being able to detect an event quickly. Another important aspect to detection is the ability to analyze and improve the detection process. Threats (internal and external) evolve over time, and a detection system must be able to adapt with the changing environment. GasDay relies on Marquette University to monitor for external threats (such as physical environments) for GasDay. However, this assessment was

performed on GasDay as a standalone organization; therefore we do not consider the full protection received from Marquette. Several students also visit GasDay after hours to study or do homework. GasDay has accepted this as a potential risk. It is still important to detect cybersecurity events from accepted risks.

For GasDay, one of the largest threats is the possibility of an internal event where a student (currently or formerly employed) creates an event. Even though this is identified as a threat, the impact of such an event is not determined.

Consider the scenario where a student, Bob, seeks to create a disruption to the lab. There is currently no method to determine where and when Bob logged onto a lab computer. While there is a chance Bob's credentials are invalidated, Section 3 identified several username/password combinations that would allow Bob to log in. Bob then has access to network drives, confidential data, source code, and more. All of this data can be modified with no alarm raised. Bob can do this because there is no mechanism to determine when an incident is occurring.

GasDay now has to determine what to do in response to the event. There is no established criteria to determine when an issue should be escalated to Marquette or managed internally. This event probably would be handled using the quick decisions and would not be logged or tracked. Since no systems are in place to log activity or perform forensics, the event probably would be attributed to an anonymous actor.

The lack of a threat profile impacts other Subcategories as well. Automated vulnerability scans are performed weekly, but without a threat profile, identified vulnerabilities cannot be scored accurately for risk. Without a threat profile, the detection process cannot be tested realistically.

Shown in Figure 19 are the overall maturity levels for Detect. GasDay suffered from missing threat profiles and monitoring the environment. Some categories that are not missing any formal requirements for MIL1 were scored at MIL0 because the practice does not occur at GasDay. For example, GasDay does not analyze events to understand the target and the attacker's methods. Although there are no controls for this Subcategory, GasDay does not perform this action.

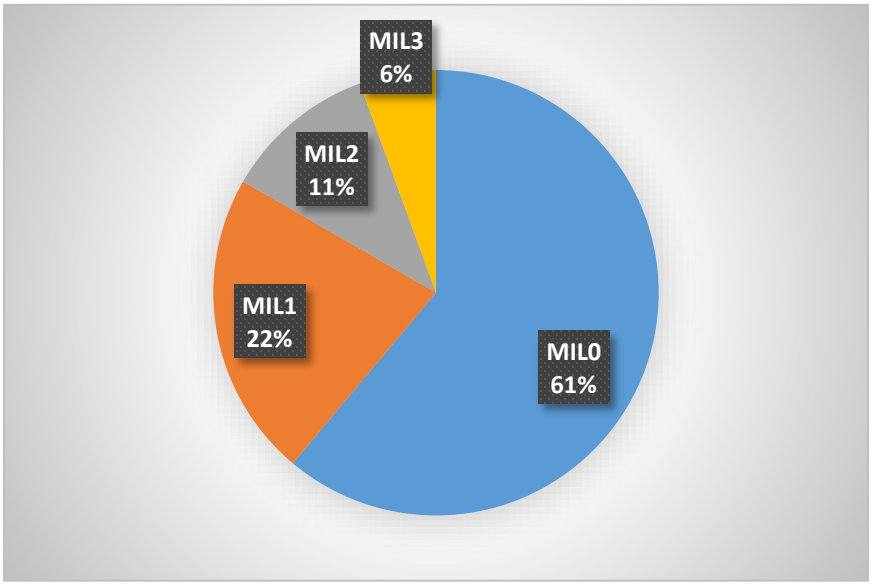

Figure 19: Maturity levels for the Detect function. Many items were scored as a MIL0 indicating no practice was in place.

GasDay heavily relies on Marquette University for protection, analysis, and notifications from external threats. GasDay cannot rely on Marquette for these same protections when the threats are internal – a far more probable threat for GasDay. Therefore, GasDay should consider these threats and establish ways of detecting events. GasDay has taken a step with the physical environment by installing doors that can be

unlocked with authorized student IDs (which are logged) and should implement similar measures for cybersecurity.

In this section, we discussed GasDay's ability to detect a cybersecurity event. GasDay relies heavily on Marquette for detection which does not address internal threats. By logging and monitoring the network and by establishing a threat profile, GasDay would be better armed to detect internal threats. In the next section, we will look at how GasDay responds to an event once identified. Again, the lack of documentation hinders GasDay's ability to respond to an event quickly.

5. Respond Function

Once a cybersecurity event is detected, it must be addressed. Responding to an event includes activities to restore functionality to affected assets, contain and mitigate the incident, and communicate information about the event to the stakeholders. For the Respond Function, we consider a detected cybersecurity event that requires a response; we do not consider undetected events or accepted risks. This is another category for which GasDay heavily relies on Marquette University. However, as with the Detect Function, we consider internal threats and GasDay's ability to respond to events without the aid of Marquette.

GasDay's ability to respond to a cybersecurity event (without the aid of Marquette) is very ad hoc. Response plans exist only for previous incidents which align with other functions of the lab (such as restoring an image to a computer). As we saw in

Section 4, without a formal definition of a cybersecurity event, the ability to respond and contain an event is limited.

In a previous event, a member of the lab had their laptop stolen. The reaction was to first change the passwords for Marquette services. Next, the stakeholders of GasDay (GasDay leaders and Marquette Engineering) were notified. The stakeholders then initiated the response plan for Marquette University. This response of GasDay was based on intuition rather than following a policy. As a result, Marquette University implemented a policy to encrypt all hard drives on laptops. Their response strategy was also updated to reflect the lessons learned from this incident. Marquette therefore exhibited a mature cybersecurity response. Conversely, the individual changed their personal policies by reducing and protecting sensitive information on their hard drive and using a password manager. These changes went unnoticed by GasDay, demonstrating an immature cybersecurity response.

Shown in Figure 20 are the maturity indicator levels for the Respond function. Some internal incidents (such as processes or automated tools) are mitigated by design of the system. When these tools fail, response plans are activated and often updated to reflect the changes to the systems. This process does not translate over to systems that are in use and controlled by employees. When one of these systems fail, the employee who knows the most is often the one to respond. The response is often based on domain knowledge and not a documented response plan. When updates occur, the expert is also unlikely to update documentation that might have existed. In addition, response plans for employee incidents often do not exist. This leads to a maturity rating of MIL1, where response plans are created and updated on an ad-hoc basis.

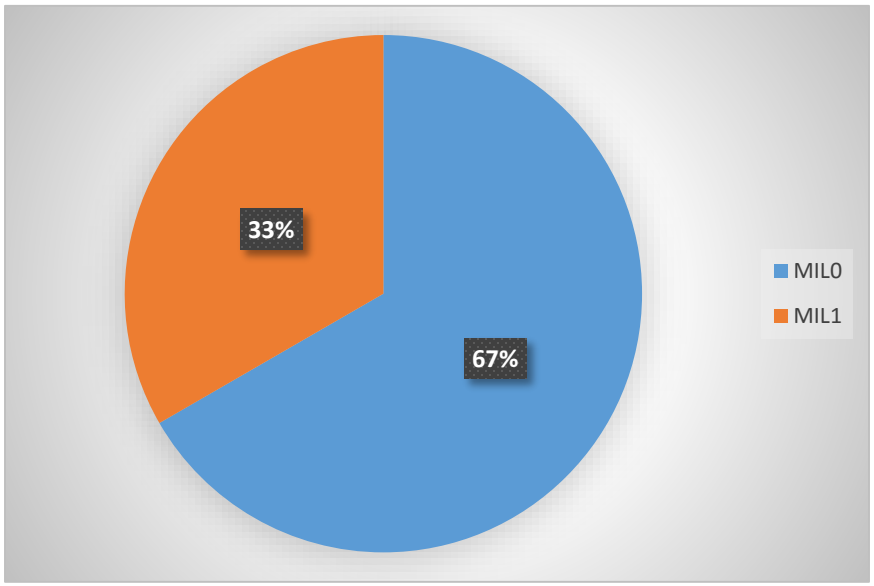

Figure 20: Maturity levels for the Respond Function. GasDay exhibits an ad hoc response level to any elevated cybersecurity event.

Overall, GasDay ranks at MIL0 for Respond. While several response plans exist that do get updated, this is the only response ability GasDay possesses. To achieve a higher maturity level, GasDay needs to outline the criteria for an incident to be classified as a cybersecurity event. It is not expected that GasDay create a response plan for every potential event. However, GasDay should create response plans for the most likely events and a template for incidents that do not have a response plan. GasDay would also benefit from sharing information with Marquette Information Technology, natural gas companies, and other research labs to achieve a broader awareness of the potential cybersecurity threats facing GasDay. To achieve these goals, GasDay should assign an individual (or group of individuals) to be in charge of these actions. By accomplishing these tasks, GasDay will be able to achieve a higher maturity level.

In this section, we discussed GasDay's ability to respond to an elevated (events that require a response) cybersecurity event. GasDay has fragmented documentation that leads to response plans being partially implemented. These plans can be implemented easily by sharing information with similar parties about likely events to occur. In the next section, we will discuss GasDay's ability to recover once an incident has taken place.

6. Recover Function

After a cybersecurity event ends, GasDay needs to recover from it. Recovery ensures that any systems and assets affected by the event are restored to normal operation. Recovery also includes communicating the event to the stakeholders in the project (internal and external) as well as managing the public relations and reputation of GasDay. A mature recovery plan enables all assets to be restored timely and accurately. The mature plan also includes a detailed list of stakeholders to contact after an incident is over.

GasDay has learned how to manage reputation and public relations effectively, but again it suffers from a lack of documentation. GasDay has reached out to customers and recovered from several incidents in the past. These incidents have ranged from mistrained models to third party patches breaking the software GasDay delivers. GasDay is able to anticipate customer and stakeholder needs. This allows GasDay to recover quickly and effectively from these incidents with a high level of maturity. As new issues with customers arise, steps to recover are documented and easily recallable in the event of a repeat incident.

Even though GasDay has become adept at dealing with any problem that might arise with customers, this is not the case with internal incident recovery. The ability to recover by restoring critical systems is sometimes documented, but the priority of these critical systems and which systems are dependent on others is obscured by confusing or non-existent documentation. As noted in previous sections, internal incidents (especially those involving personnel) often do not have a recovery plan, nor is one created when an incident occurs. If GasDay were to experience an event never seen before, GasDay would struggle to recover. If an event occurred when leaders of the lab were away, there would be no one to guide the ad-hoc response currently seen. As a result, the maturity levels are kept low.

Seen in Figure 21 are the overall maturity levels for Recover. GasDay scores a MIL2 for managing public relations after an incident occurs. GasDay could improve the maturity level of almost every Subcategory by adopting practices already in place for other types of cybersecurity events. The overall rating for GasDay's ability to recover is a MIL1. GasDay understands what it means to recover from an incident but would do well to establish and document recovery plans for the most likely risks that have not yet been encountered.

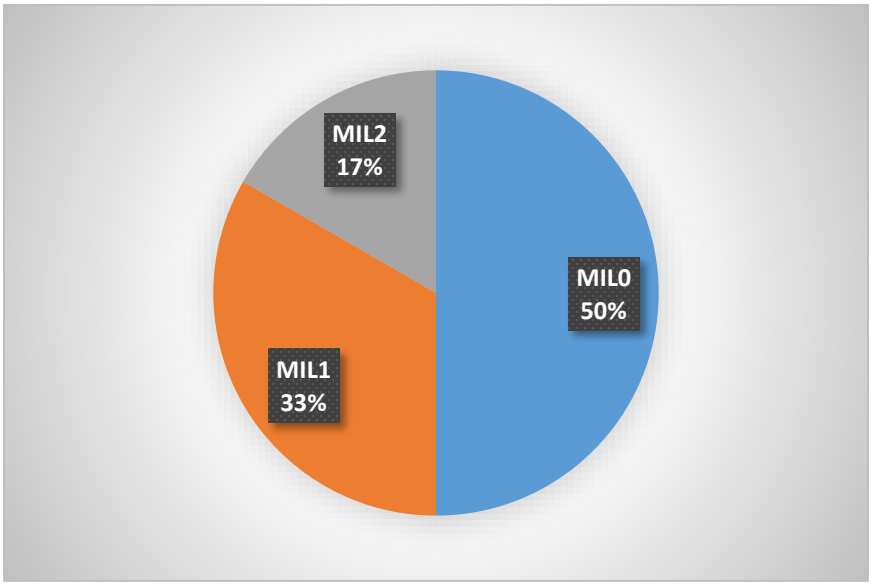

Figure 21: Maturity levels for the Recover Function. GasDay demonstrates a high level of maturity hindered by documentation.

In this section, we discussed GasDay's ability to recover from an elevated cybersecurity event. GasDay demonstrates a maturity level that is built from previous experience responding to several events but has not been documented as the experience has grown. In the next section, we will look at the overall maturity levels and present recommendations on which maturity levels GasDay should consider improving.

7. Current Profile

The current profile is the overall look at that maturity of GasDay. By identifying the overall maturity levels, GasDay can prioritize which actions should be preserved, improved, or relaxed. Shown in Table 2 are the total count of maturity indicator levels for the Subcategories of the NIST Framework. Almost half of all the Subcategories are rated

at MIL0 representing no practice or no maturity. As we noted throughout this chapter, one of the largest obstacles for GasDay is the inconsistency of documentation. We recommend that GasDay focus on documenting what it does and insuring that existing documentation is accurate.

Table 2: Complete count of Subcategory Maturity Indicator Levels for GasDay

Maturity Level	Number of Subcategories
MIL0	46
MIL1	40
MIL2	10
N/A	1
MIL3	1
Grand Total	**98**

Although GasDay is able to rely on Marquette University for several cybersecurity activities, the threat of an internal incident remains. GasDay should consider focusing on activities which help mitigate this threat. Activities such as privilege creep, activity logging, and configuration change control will help GasDay to protect, detect, and recover from an internal incident.

Continuing the activities presented in Chapter 2 and Chapter 3 of this thesis will allow GasDay to evaluate the effectiveness of any new activities. Before we begin adding or changing controls, GasDay should be understand how the control can be tested in addition to which threat or risk the control is addressing. The best way for GasDay to address this is by creating a threat profile and risk register to track all known threats and risks to GasDay.

By implementing these controls, GasDay will be able to raise the maturity level of several subcategories from a MIL0 to a MIL1. From here, GasDay will be able to prioritize and identify which Subcategories are necessary for GasDay to improve further. After a set period of time (regardless of the changes made), GasDay should reevaluate the cybersecurity of the lab using a method similar to the one presented in this chapter.

8. Conclusion

In this chapter we looked at the NIST Cybersecurity Framework. We applied the Framework in combination with the Oil and Natural Gas Cybersecurity Capability Maturity Model to evaluate the cybersecurity maturity of GasDay. We found that GasDay's performs many cybersecurity activities in an ad-hoc or unexperienced manner. GasDay is also lacking several critical items (threat profile, risk register) that guide cybersecurity decisions and evaluations. We presented several recommendations for GasDay to improve the maturity level of the lab.

We identified the following Subcategories where GasDay demonstrated maturity:

- Following FERPA regulations
- Responding to cybersecurity incidents involving customers
- Cataloging external information systems

We recommend GasDay take the following actions to increase the overall maturity level of the lab:

- Establish a threat profile that identifies the potential threats GasDay faces

- Establish a risk register which catalogues known risks and their mitigation strategies
- Document existing processes and remove documentation that is out of date or inaccurate
- Identify a target profile for cybersecurity and prioritize which Subcategories are important to improve first

CHAPTER 5

Conclusion

This chapter summarizes and draws conclusions from the work presented in this thesis. We discuss future work for GasDay and how this work could be applied in other research labs.

1. Contributions of Research

In this thesis, we demonstrated different methods to evaluate the cybersecurity of a research lab that functions like a small business. We showed how a research lab is prone to ad-hoc solutions towards cybersecurity. Small businesses and research labs tend to employee only a few employees; cybersecurity is either not considered or one of several roles for an employee. Research labs and small businesses often are unaware of the value of the data they possess. We showed how a mature research lab (GasDay) can possess critical vulnerabilities that could lead to a major cybersecurity event. We presented automated tools and a flexible cybersecurity framework which can help mitigate risks and identify threats.

2. Summary of Findings

In each chapter, we applied a different technique to evaluate different aspects of the cybersecurity of GasDay. As with software testing, our goal was to find

vulnerabilities and gaps in the security policy. Finding weaknesses indicates the test was successful and therefore vulnerabilities are a positive outcome. Chapter 2 used penetration testing to evaluate the security of GasDay's network and the devices that are connected to the network. Chapter 3 used a similar technique of penetration testing and applied it to the upcoming product, GasDay Web. Chapter 4 analyzed the security policy of GasDay and identified the overall maturity level of the lab.

In Chapter 2, we presented penetration testing as a tool that could be used to simulate an attacker targeting GasDay. We outlined different nodes on the network that were vulnerable to different types of attacks. Many vulnerabilities were remedied during the course of this work. We also presented OpenVAS, a tool that could be used to automatically detect many of these vulnerabilities. We demonstrated that several critical vulnerabilities that were not detected automatically.

Chapter 3 demonstrated how penetration testing was used to evaluate GasDay Web. Web vulnerabilities are often the direct result of programmer error. We demonstrated how a malicious actor could bypass authentication to gain access to GasDay Web. We also demonstrated how the same actor could perform actions, such as deleting a user, without being authorized. We presented a tool, the Zed Attack Proxy, to automatically detect web vulnerabilities. Again, we showed that the tool was able to detect several vulnerabilities, but was unable to capture all the critical vulnerabilities.

Chapter 4 discussed the overall cybersecurity policy at GasDay. We presented the NIST Cybersecurity Framework as a way for GasDay to evaluate the cybersecurity maturity of GasDay. The Framework also allows GasDay to self-prioritize which controls and aspects of cybersecurity are most important. We performed an evaluation of the

GasDay cybersecurity policy. We demonstrated that GasDay's approach to security is largely ad-hoc, which has often lead to slow responses to cybersecurity events. We presented several areas where GasDay could improve and use as a starting point as a larger cybersecurity risk management plan.

In the next section, we further prioritize our findings into a high level recommendation for GasDay. We propose items that will give GasDay the greatest return of investment compared to the resources required to implement the items.

3. Overall Recommendations

Throughout this thesis, we noted vulnerabilities of GasDay and of GasDay Web. Several of these have been fixed since the start of this work, while others still remain. An important next step for GasDay is to establish a risk register and threat profile. A risk register is a documented list of known risks for GasDay and their mitigation strategies. An example entry might look like:

Risk: GasDay employees student workers of Marquette

Mitigation: All employees must undergo FERPA training

A threat profile contains a list of threats to GasDay. Threats cannot be controlled, while risk can be mitigated. The threat profile should also identify the likelihood of a threat occurring. An example entry might look like:

Threat: Disgruntled student seeking to corrupt data

Probability: Low

Impact: Medium – no method to track data changes. Easy to replace

Both the risk register and threat profile should be updated regularly as new threats/risk arise and to re-evaluate already entered items. These items will help guide which items in the NIST Framework should be improved, as well as which assets within the network are important to protect. These items should be updated around the same time further evaluations or tests are being conducted. Cybersecurity is an ongoing work; penetration tests and evaluations using the NIST Framework should be performed regularly.

We made the following recommendations from other chapters that are still pending or have not been addressed yet:

- Assets (software, hardware) should be patched on a regular schedule. Several computers are months to years out of date.
- GasDay Web should force HTTPS at all times
- GasDay Web should sanitize all input before using it
- Establish a threat profile to identify potential threats
- Establish a risk register to catalogue known risks and mitigation strategy
- Document existing processes (not limited to cybersecurity) and remove out of date or inaccurate documentation

We also made the following recommendations thinking ahead for what GasDay should aim for after the above recommendations have been considered:

- Automate patching and inventory to keep assets accurate and consistent
- Consider cybersecurity risks when making changes to the environment. This includes asset (hardware, software, personnel) changes, customer changes, and changes to third parties
- Adopt the principle of least privilege everywhere instead of in choice areas

- Continue using (and consider automating) vulnerability detection for the network and GasDay Web to reduce detection time
- Train developers, support staff, team leaders, and management on cybersecurity

Cybersecurity should be integrated into the life cycle for every facet of GasDay. Developers, support staff, and business staff, should understand their role in keeping GasDay secure. Developers should be trained in secure coding if they are working on GasDay Web. Support staff should be trained on setting up systems securely. Business staff should consider the potential risks or threats when interacting with new customers. As the maturity level of the lab grows, GasDay could be in a position to help raise awareness and maturity levels of our customers.

4. Conclusion

This thesis tested how cybersecurity is viewed and implemented at GasDay. Since the start of this work, GasDay has learned how cybersecurity integrates into life cycle of a software product. GasDay has several cybersecurity goals as well as examples on how evaluations can take place in the future. GasDay is better prepared to handle potential cybersecurity events in the future.

BIBLIOGRAPHY

[1] R. Massie, “Cyber Security Primer,” 2014. [Online]. Available: http://www.umuc.edu/cybersecurity/about/cybersecurity-basics.cfm. [Accessed: 18-Jan-2016].

[2] M. Stamp, *Information Security: Principles and Practice*. John Wiley & Sons, 2011.

[3] Symantec, “Internet Security Threat Report 2014: Volume 19,” *2013 Trends*, Apr-2014. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf. [Accessed: 20-Oct-2015].

[4] C. Wueest, “Targeted Attacks Against the Energy Sector,” *Symantec Security Response, Mountain View, CA*, 2014. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/targeted_attacks_against_the_energy_sector.pdf. [Accessed: 21-Oct-2015].

[5] T. Bolt, “Business Blackout: The Insurance Implications of a Cyber Attack on the US Power Grid,” *LLoyds*, 2015.

[6] N. Falliere, L. O. Murchu, and E. Chien, “W32.Stuxnet Dossier,” Feb-2011. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf. [Accessed: 13-Oct-2015].

[7] Symantec, “W32.Duqu The precursor to the next Stuxnet,” *Symantec Security Response*, 2011. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf. [Accessed: 14-Oct-2015].

[8] Mandiant, “APT1 Exposing One of China’s Cyber Espionage Units,” *Report*, 2013. [Online]. Available: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf. [Accessed: 07-Oct-2015].

[9] “Family Educational Rights and Privacy Act (FERPA).,” *Journal of Empirical Research on Human Research Ethics : JERHRE*, vol. 2, p. 101, 2007.

[10] Penetration Test Guidance Special Interest Group and PCI Security Standards Council, “Penetration Testing Guidance,” Mar-2015.

[11] R. H. Brown, “Prefiled Direct Testimony of Ronald H. Brown,” 2014. [Online]. Available: https://enstarnaturalgas.com/wp-content/uploads/2014/09/Brown-Testimony-With-Exhibits-00218228xB615B.pdf. [Accessed: 19-Sep-2015].

[12] T. H. Sanders, “Direct Testimony and Exhibits of Tommy H. Sanders,” 2012. [Online]. Available: https://www.nmgco.com/PDF/Sanders Direct.pdf. [Accessed: 19-Sep-2015].

[13] Office of Research And Sponsored Projects, “A Century of Scholarship,” 2004. [Online]. Available: http://www.marquette.edu/orsp/documents/CenturyOfScholarship.pdf. [Accessed: 11-Dec-2015].

[14] “User:Streeter8 - MSCS REU.” [Online]. Available: http://acm.mscs.mu.edu/wiki-reu/index.php/User:Streeter8. [Accessed: 04-Oct-2015].

[15] “Multidisciplinary Capstone Design Projects.” [Online]. Available: http://gasdayweb.com/mumcd/12assets/course.css. [Accessed: 15-Sep-2015] [Removed].

[16] “MX Lookup Tool,” *MxToolbox*. [Online]. Available: http://mxtoolbox.com/. [Accessed: 18-Aug-2015].

[17] N. Chris, K. David, and C. J. R. Reil, “The Penetration Testing Execution Standard,” *2014*, 2009. [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page. [Accessed: 17-Oct-2015].

[18] G. Lyon, “Nmap: The Network Mapper,” 2015. [Online]. Available: https://nmap.org/. [Accessed: 18-Sep-2015].

[19] B. Fenner and J. Flick, “Management Information Base for the User Datagram Protocol (UDP),” 2005.

[20] Codenomicon Defensics, “Heartbleed Bug,” 2014. [Online]. Available: http://heartbleed.com/. [Accessed: 26-Oct-2015].

[21] “OpenVAS - NVT Feed.” [Online]. Available: http://www.openvas.org/openvas-nvt-feed.html. [Accessed: 15-Oct-2015].

[22] Rapid7, “Penetration Testing Software, Pen Testing Security | Metasploit,” *Metasploit.com*, 2015. [Online]. Available: http://www.metasploit.com/. [Accessed: 18-Aug-2015].

[23] Rapid7, “Vulnerability & Exploit Database | Rapid7.” [Online]. Available: www.rapid7.com/db/. [Accessed: 01-Aug-2015].

[24] HP, “Printer Job Language Technical Reference Manual,” 2003. [Online]. Available: http://h20565.www2.hp.com/hpsc/doc/public/display?docId=emr_na-bpl13207. [Accessed: 21-Sep-2015].

[25] “Microsoft Security Bulletin MS15-034 - Critical,” *Microsoft Technet*, 2011. [Online]. Available: https://technet.microsoft.com/en-us/library/security/ms15-034.aspx. [Accessed: 14-Sep-2015].

[26] J. Tang, “IIS at Risk: An In-depth Look into CVE-2015-1635.” [Online]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/iis-at-risk-an-in-depth-look-into-cve-2015-1635. [Accessed: 17-Oct-2015].

[27] Microsoft Technet, “MS08-067: Vulnerability in Server service could allow remote code execution.” [Online]. Available: https://support.microsoft.com/en-us/kb/958644. [Accessed: 15-Oct-2015].

[28] A. Muller and M. Meucci, “OWASP Testing Guide v4,” 2014. [Online]. Available: https://www.owasp.org/images/5/52/OWASP_Testing_Guide_v4.pdf. [Accessed: 03-Dec-2015].

[29] Symantec, “Internet Security Threat Report,” 2015. [Online]. Available: https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932_GA-internet-security-threat-report-volume-20-2015-social_v2.pdf. [Accessed: 15-Jul-2015].

[30] Security Aspect, “2013 Global Application Security Risk Report,” 2013. [Online]. Available: http://www.aspectsecurity.com/the-2013-global-application-security-risk-report. [Accessed: 11-Nov-2015].

[31] C. Wysopal, “State of Software Security Report,” 2011. [Online]. Available: http://info.veracode.com/rs/veracode/images/VERACODE-SOSS-V4.PDF. [Accessed: 05-Dec-2015].

[32] OWASP, “OWASP Top 10 - 2013.” [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10. [Accessed: 07-Dec-2015].

[33] S. Bennetts, “OWASP Zed Attack Proxy Project,” 2012. [Online]. Available: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project. [Accessed: 13-Dec-2015].

[34] OWASP, “Man-in-the-middle attack - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Man-in-the-middle_attack. [Accessed: 02-Feb-2016].

[35] “Wireshark.” [Online]. Available: https://www.wireshark.org/. [Accessed: 07-Feb-

2016].

[36] Microsoft, “Microsoft Azure Network Security,” Feb-2015. [Online]. Available: http://download.microsoft.com/download/C/A/3/CA3FC5C0-ECE0-4F87-BF4B-D74064A00846/AzureNetworkSecurity_v3_Feb2015.pdf. [Accessed: 23-Jul-2015].

[37] A. Barth, “HTTP State Management Mechanism,” *The Internet Engineering Task Force (IETF)*, 2011. [Online]. Available: http://tools.ietf.org/html/rfc6265#section-3. [Accessed: 03-Oct-2015].

[38] “HTTP Cookies,” *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. [Accessed: 26-Jan-2016].

[39] J. Blatz, “CSRF: Attack and Defense,” *McAfee Foundstone Professional Services*, 2007. [Online]. Available: http://www.foundstone.com.au/uk/resources/white-papers/wp-csrf-attack-defense.pdf. [Accessed: 20-Nov-2015].

[40] M. Howard and D. LeBlanc, *Writing Secure Code*. Pearson Education, 2006.

[41] Microsoft, “int (C# Reference).” [Online]. Available: https://msdn.microsoft.com/en-us/library/5kzh1b5w.aspx. [Accessed: 27-Jan-2016].

[42] B. Obama, “Improving Critical Infrastructure Cybersecurity,” *Executive Order 13366*, 2013. [Online]. Available: https://www.whitehouse.gov/the-press-office/2013/02/12/executive-order-improving-critical-infrastructure-cybersecurity. [Accessed: 03-Jan-2015].

[43] NIST, “Framework for Improving Critical Infrastructure Cybersecurity,” *National Institute of S*, 2014. [Online]. Available: http://www.nist.gov/cyberframework/upload/cybersecurity-framework-021214.pdf. [Accessed: 04-Jun-2015].

[44] International Organization for Standardization, “ISO/FDIS 31000:2009: Risk management - Principles and guidelines,” 2009.

[45] NIST, “Guide for Applying the Risk Management Framework to Federal Information Systems,” *NIST Special Publication*, vol. 800, no. 37, p. 93, Feb. 2010.

[46] “Reducing Cyber Risk to Critical Infrastructure: NIST Framework.” [Online]. Available: http://energy.gov/oe/services/cybersecurity/reducing-cyber-risk-critical-infrastructure-nist-framework. [Accessed: 27-Jul-2015].

[47] U.S. Department of Energy, “Electricity Subsector Cybersecurity Capability Maturity Model (ES-C2M2),” Feb-2014. [Online]. Available: http://energy.gov/sites/prod/files/2014/02/f7/ES-C2M2-v1-1-Feb2014.pdf. [Accessed: 02-Jan-2015].

[48] U.S. Department of Energy, “Oil and Natural Gas Subsector Cybersecurity Capability Maturity Model (ONG-C2M2),” 2014. [Online]. Available: http://energy.gov/sites/prod/files/2014/03/f13/ONG-C2M2-v1-1_cor.pdf. [Accessed: 02-Jan-2015].

[49] F. B. Schneider, “Least Privilege and More,” *Computer Systems*, pp. 209–213, 1972.