ATHABASCA UNIVERSITY

EFFECTIVE SQL INJECTION ATTACK RECONSTRUCTION USING NETWORK RECORDING

BY

ALLEN POMEROY

A project submitted in partial fulfillment
Of the requirements for the degree of
MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta
June, 2010

DEDICATION

This work is dedicated to my partner Amanda for all her support and understanding, and to my parents Jan (1928 – 2010) and Bill (1925 – 2001) for teaching me to strive for the best.

ABSTRACT

This paper presents a network recording enhanced IDS system to capture SQL injection attacks. Using efficient network recording allows retrospective analysis of potentially malicious web application traffic, enabling a complete picture of an attacker's actions to be recorded. Web applications offer business and convenience services that society has become dependent on. These services have serious weaknesses that can be exploited by attackers. Success of these applications is dependent on end user trust, therefore application owners must take additional steps to ensure the security of customer data and integrity of the applications. Web applications are under siege from cyber criminals seeking to steal confidential information and disable or damage the services offered by these applications. Successful attacks have lead to some organizations experiencing financial difficulties or even being forced out of business, such as CardSystems Solutions, Inc. Organizations have insufficient tools to detect and respond to attacks on web applications, since traditional security logs have gaps that make attack reconstruction nearly impossible. This paper explores network recording challenges, benefits and possible future use. A network recording solution is proposed to detect and capture SQL injection attacks, resulting in the ability to successfully reconstruct SQL injection attacks.

ACKNOWLEDGEMENTS

I would like to acknowledge:

- My amazing partner Amanda for her unwavering support;
- My colleagues Jeff, Tom, Doug and Bob for numerous technical and grammatical reviews;
- Beverly for grammatical reviews;
- Seth Hall for Bro-IDS configuration help;
- The Bro-IDS and Time Machine project teams; and
- Dr. Qing Tan for countless reviews and much appreciated encouragement.

TABLE OF CONTENTS

LIST OF TABLES AND LISTINGS

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

Pervasive availability and convenience of services delivered through web application technologies has made society dependent on these services as business-to-business and business-to-consumer functions have made it easier than ever to communicate, research, learn, organize ourselves and conduct business. The dependence on web applications also increases our exposure to the effects of attacks that may directly impact individuals through compromise of confidential information, resulting in identity theft, or denial of services. This may further compound into lost revenue and possibly liabilities, as illustrated by attacks forcing companies out of business [1,2]. Continued success of these applications is completely dependent on trust placed in the applications by the users and this demands increased levels of security to protect these services from attackers seeking to obtain financial and identity information or disrupt services [3].

In August 2008, hostilities between Russia and Georgia resulted in devastating denial of service attacks that shut down Georgia news agencies and government functions [4]. A report prepared for US President Obama indicates over $1 trillion dollars of key intellectual property was stolen through cyber security breaches in 2008 alone [5]. Researchers have determined over 640,000 sites were successfully attacked in the third quarter of 2009 alone, with approximately 5.8 million successful attacks throughout 2009 [6]. The Verizon Business Risk Team determined there were 285 million records stolen in 2008 through only 90 confirmed breaches [6]. Of those breaches, SQL injection attacks were the among the top flaws exploited.

Attacks are focused not only on the web application technology used to serve customers, but also on the client side web browsers and the underlaying infrastructure where weaknesses also can lead to unauthorized control of client machines. Exacerbating the problem is the ineffectiveness of

traditional network security measures such as firewalls that block all traffic other than HTTP or HTTPS ports, since the majority of attacks are now targeted at the application layer [7,8].

Security of applications and services offered over the Internet has been proven to be a consistent and growing problem. Security concepts are often grouped into a triad including confidentiality, integrity and availability. Users must be able to trust the confidentiality and integrity of business transactions, since personal and financial information are exchanged with little knowledge of the user regarding the trustworthiness of the supporting infrastructure [3]. Attackers should not be able to view or modify confidential information. Unauthorized disclosure of information results in loss of confidentiality, which can be especially serious with sensitive information such as credit card numbers, personal identity information, medical records, private conversations, and corporate business information. Unauthorized alteration of information results in loss of integrity, such as alteration of a product price, quantity or shipping address. Availability of service is important to business success as well as to critical services, such as critical infrastructure and life safety systems.

Verizon notes “as the cybercrime market evolves, attackers, targets, and techniques do as well. As supply has increased and prices fallen, criminals have had to overhaul their processes and differentiate their products in order to maintain profitability. In 2008 this was accomplished by targeting points of data concentration or aggregation and acquiring more valuable sets of consumer information.” [6] Targets for criminals have shifted from credit card information to theft of Personal Identification Numbers (PIN) with associated account information. As theft of credit card information has dropped from $10-$16 per card number to less than $0.50, cyber-criminals have adapted by shifting their focus to PIN numbers and proprietary corporate information.

Web application security flaws are the primary exploit vector for attackers and have been shown by the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD)

to still be increasing in volume. Investigation of these application attacks are often hindered by incomplete forensic information. The sparse forensic information makes it difficult or even impossible to identify the exploited flaw or the perpetrators. Organizations are seeking ways to improve the reconstruction of web attacks in order to support criminal or civil legal proceedings as well as to understand and remedy web application vulnerabilities.

1.1 Statement of Purpose

The purpose of this paper is to show the value of employing network forensic techniques and technologies to improve investigation of SQL injection attacks on web applications. An overview of web application flaws that can be exploited to gain privilege escalation as well as logging and monitoring technology and techniques that can be used to correctly identify such attacks will be presented. Use of network based Intrusion Detection System (IDS) tools to trigger network recording of suspected application attacks will be compared to traditional logfile-only analysis to demonstrate the effectiveness of network recording in reconstruction of SQL injection attacks.

Known web attacks will be conducted against a lab environment that is representative of contemporary web application environments. The test environment will be successively configured with traditional network and operating system security capabilities (network firewall, standard operating system and web server logging), then enhanced network recording capabilities. An attack reconstruction will be attempted in both configurations, to prove the effectiveness of one approach over the other. The comparison of differing methods for reconstructing a SQL injection attack against a web application will illustrate the benefits of using network recording to enhance the reconstruction process.

Although current literature focuses on web application security, investigation of security breaches and network forensics, this paper highlights the effectiveness of network recording to determine the exact attack vector used and the extent of the intrusion.

1.2 Research Problem

Sophistication of web applications and user dependence on them continues to grow. Attackers are exploiting an ever increasing number of vulnerabilities and more powerful exploit frameworks to successfully attack web applications and services. Comprehensive new forensic tools are required that can definitively demonstrate how web applications have been attacked and what related activities were committed by the attacker.

This paper will discuss network forensic techniques and technologies that are available to be integrated with or deployed along side web server technology to improve the investigation of successful and unsuccessful attacks. Developers of web applications are facing increasing pressure to shorten the development cycle resulting in less quality assurance (QA) testing. This paper will answer the question "Can technologies and techniques used to support and perform network forensics be used to more effectively detect, recover from attacks, and simultaneously provide QA data useful to improve web application security?" Without an increase in development time to improve application QA or an alternative method to improve web application security, these applications will continue to provide an attack vector for criminals to exploit.

1.3 Organization of Paper

This paper attempts to illustrate the benefit of using network forensic techniques such as IDS triggered network recording to reveal the detailed exploit of a SQL flaw versus standard web server logging. The remainder of this paper is organized into the following chapters:

- Web Application Vulnerabilities, providing background depth into web application programming flaws and suggested mitigations; state of firewall, operating system, web server and web application logging; network recording techniques; and incident response (forensic) techniques.

- Attack Analysis, providing depth into anatomy of a SQL injection attack; challenges with traditional attack reconstruction using log files; and reconstruction of a successful attack with network recording capabilities.
- Recommendations and Conclusions, outlining the benefits and challenges to using network recording techniques as well as possible future work.

1.4 Related Work

Review of existing research found coverage of

- Web application and infrastructure vulnerabilities;
- Vulnerability mitigation techniques; and
- Attack detection techniques

that could be used to extend the concepts presented in this paper.

The Open Web Application Security Project (OWASP) categorizes web application vulnerabilities including both software and configuration flaws. Although this paper focuses on SQL injection flaws, OWASP shows there are several other types of web application vulnerabilities that also present threats to proper operation of web applications. [7] indicate SQL injection and cross-site scripting attacks are common and damaging. Broken access controls present opportunities for attackers to exploit weaknesses within web applications and [9] indicate that this flaw type can be more difficult to mitigate with web application firewalls than SQL injection or cross-site scripting errors. In addition to OWASP classified web application errors, web server software and other network daemons are subject to memory corruption attacks including buffer over/under flows, integer overflows and format string manipulation [10]. Client side (browser) attacks are discussed by [11] where a methodology to detect malicious servers is outlined. Even with known vulnerabilities successfully remediated, sites can

be successfully attacked due to configuration mistakes [12].

Mitigation techniques explored include use of web application firewalls [9] and SQL proxy systems [13] that strive to actively block SQL injection attacks.

Although Bro-IDS intrusion detection system (IDS) was chosen to detect SQL injection attempts within the research for this paper, [14] illustrate a SQL IDS system that could be used to extend the proposed solution to more accurately detect SQL injection attacks. Other detection techniques were proposed by [15], including agent based IDS sensing on web application servers.

We are unaware of existing projects or papers that parallel the objective of this paper. There are several related topics that we draw on heavily to produce the conclusions and in fact this work could not exist without these prior projects, including the Bro-IDS and the Time Machine (TM) network recording project.

There are commercial network recording products including NetWitness [16] and Niksun NetDetector [17], however they do not appear to offer tight integration with intrusion detection systems. Based on these facts, we believe the ideas expressed in this paper are new work.

1.5 Notice to Reader

Throughout this paper the term 'attacker' designates individuals or organizations who are attempting to gain unauthorized access to web applications of an organization. These could be 'white-hat' individuals executing attacks without the intent to harm the organization such as information security professionals who work with the organization, or they may be 'black-hat' individuals who are attempting to commit some form of fraud.

Even though variable by jurisdiction, typically all access to computer systems and computer networks without express authorization of the owner is a criminal offense with serious legal

ramifications. All work performed for this paper was undertaken within virtual lab environments that were well isolated from external networks; in the case readers endeavor to reproduce or add to this body of work, they are strongly encouraged to do so on explicitly authorized private networks.

Although believed to be safe at the time this paper was researched, attack tool and information sites listed in this paper may contain malicious software. The reader is cautioned to take appropriate care when visiting these sites.

CHAPTER 2 - WEB APPLICATION VULNERABILITIES

2.1 Background

According to web security researcher company Dasient, there were over 640,000 Internet sites attacked successfully in the third quarter of 2009 alone, with approximately 5.8 million successful attacks throughout 2009 [18]. The frequency and severity of these attacks are growing due to the large number of vulnerabilities that exist at the application layer [19] in comparison to the other supporting infrastructure layers such as network devices or operating systems. There continues to be an alarming number of programming and implementation flaws in these critical applications [12,19], resulting in sustained successful high profile attacks. The number of attacks detected against web applications is considerable and includes successful attacks against recognizable organizations such as Apache.org, Embassy of India in Spain, Fox Sports, Indian Institute of Remote Sensing, Intel, Heartland Payment Systems [20], Monster.com, MSN Canada, NASA, Royal Bank of Scotland, TJX [21], Torrentreactor, UK Parliament, U.S. Army, U.S. House of Representatives, Wall Street Journal and Yahoo Careers.

Hacker websites such as www.zone-h.org list victims of successful exploits including:

- K-12 schools
- Universities
- Government agencies
- IT organizations, and
- Open source software projects.

Verizon research shows that over 60 percent of the confirmed breaches involved the retail and financial services sectors, with over 70 percent of the attack sources from external networks such as the Internet.

The same research also showed a multi-year trend of increasing attacks from external sources.

Due to the explosive popularity of web applications for offering services over the Internet, many applications are deployed with compressed timeframes between development and release into production. Shorter development cycles make it difficult to perform adequate quality and security testing, resulting in serious vulnerabilities in these web applications [22].Web applications can be successfully attacked where there are incorrectly chosen security controls, incorrectly configured or deployed controls, or through flaws in the controls themselves. Since SQL injection attacks rely on insufficient programming, this paper focuses on flawed controls.

Verizon found the majority of incidents investigated fell within the Financial Services sector, including financial services companies themselves as well as retail organizations. Often these industries will use commercial off the shelf (COTS) systems. Several attacker sites include exploits for popular COTS systems including Juniper, Citrix, Oracle and Websense. Exploits were found for:

- content management systems,
- blogs,
- ecommerce,
- games,
- web site management,
- real estate,
- car dealerships, and
- business-to-business portals.

A survey of several attack tool websites found 47% of the exploits were oriented to performing

SQL injection attacks with the goal of gaining unauthorized access to victim sites [23].

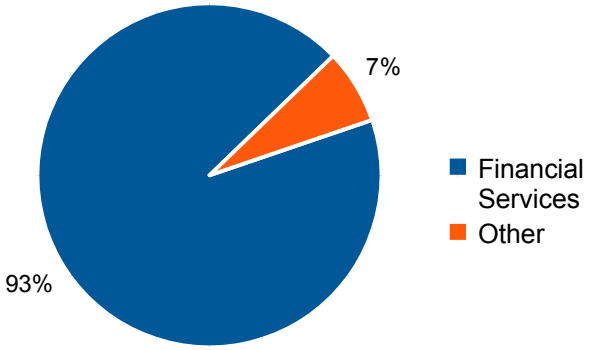

Figure 1: Attacked Industries

In Figure 2, Verizon clearly shows external attack sources are still the overwhelming majority compared to business partner and internal sources. Given many organizations use multiple-tier web sites backed with a SQL database, it stands to reason that these sites need to be protected from external attackers.

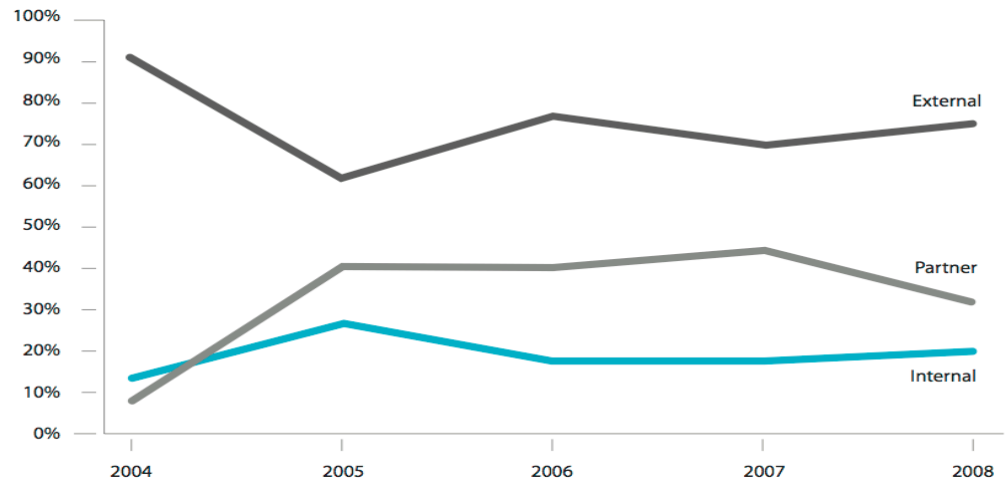

Figure 2: Verizon Breach Sources

In Figure 3, Verizon also illustrates successful attack vectors seen in over approximately 600 incident response cases. SQL injection attacks are shown to be the source of the majority of records

stolen (79%).

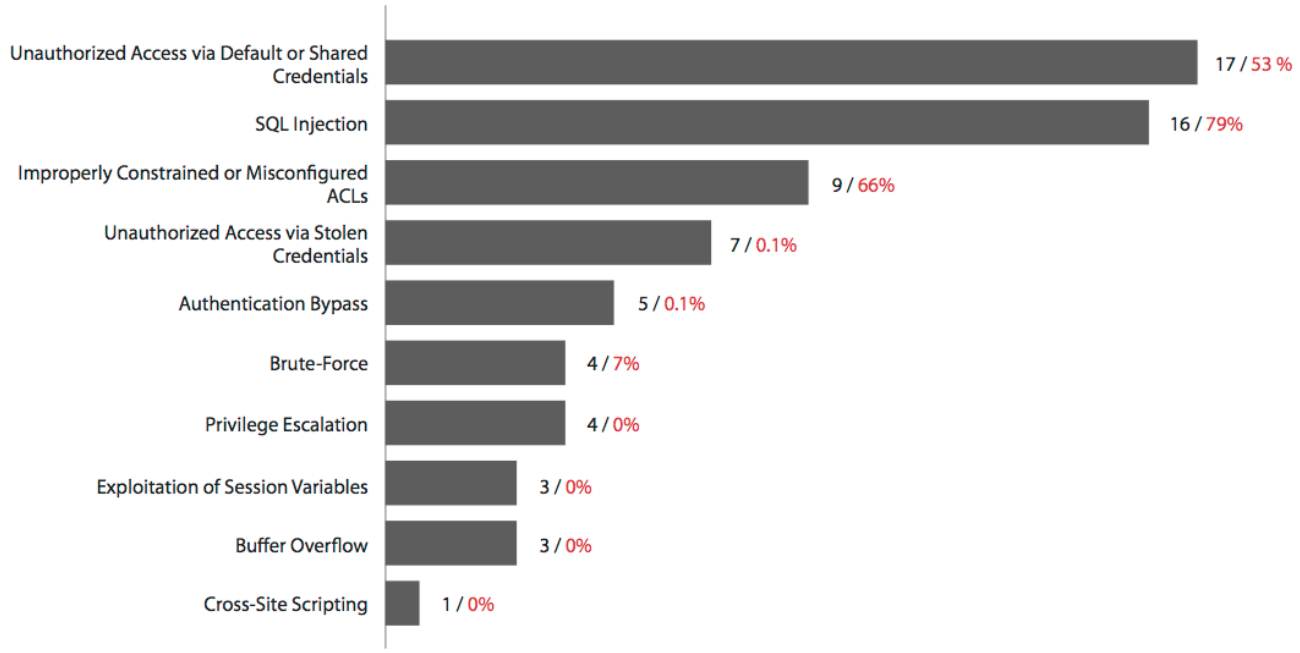

Figure 3: Verizon Attack Vector Summary – confirmed breaches (percent of total records stolen)

This paper shows the grave danger presented by web site flaws that allow SQL injection attacks to succeed.

2.2 Web Application Vulnerabilities

OWASP indicates that "insecure software is already undermining our financial, healthcare, defense, energy, and other critical infrastructure." [24] This stark warning is echoed by many industry experts including SysAdmin, Audit, Network, Security Institute (SANS) (www.sans.org), Mitre (cwe.mitre.org), NIST NVD (nvd.nist.gov), and in most of the security research we examined for this paper. [7] states up to 70% of the web applications responsible for handling much of the business conducted over the Internet today are affected by insecure software. Examples of impact caused by web application vulnerabilities, and SQL injection attacks in particular, include the demise of CardSystem Solutions, Inc., a credit card payment processor that handled $15 billion in transactions annually. CardSystems suffered a security breach that was discovered in May 2005, where attackers had

successfully executed a SQL injection compromise that resulted in the major credit card companies revoking CardSystems right to process consumer payments on behalf of merchants. In December 2005, CardSystems was sold to Pay By Touch for an undisclosed amount and eventually Pay By Touch ceased doing business in March 2008. Over 40 million credit card numbers were exposed during the attacker infiltration of the payment processor. During this time the payment processor was handling US government Electronic Benefit Transfer transactions that are used to pay social welfare and unemployment benefits [1,25].

Web application attacks occur via several different types of vulnerabilities such as:

- SQL and OS injection flaws
- Cross site scripting (XSS)
- Malicious code execution (through remote file inclusion)
- Direct object reference, and
- Cross site reference forgery buffer overflow.

OWASP and the Common Weakness Enumeration (CWE) project researched web application attacks and developed a list of the most common flaw types found in web applications. Those with the most severe impacts are shown in Table 1.

Table 1: Common Web Application Vulnerabilities

OWASP 2010 Top Ten Risks	**CWE/SANS 2010 Risks**	**Description**
A1 – Injection flaws	CWE-89 (SQL injection), CWE-78 (OS command injection)	Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
A2 - Cross Site Scripting (XSS)	CWE-79 (Cross-site scripting)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.
A3 -Broken Authentication and Session Management	CWE-306, CWE-307, CWE-798	Developers frequently build custom authentication and session management schemes, but building these correctly is hard. As a result, these custom schemes frequently have flaws in areas such as logout, password management, timeouts, remember me, secret question, account update, etc. Finding such flaws can sometimes be difficult, as each implementation is unique.
A4 - Insecure Direct Object Reference	CWE-285	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.
A5 - Cross Site Request Forgery (CSRF)	CWE-352	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.
A6 – Security misconfiguration	No direct mappings; CWE-209 is frequently the result of misconfiguration	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.

Security vendors also report specific vulnerabilities that affect the supporting infrastructure for web applications, such as the web servers, operating systems, and network equipment. These vulnerabilities enable exploitation of a web application or service even if the vulnerability does not exist in the web application itself [26]. Similar vulnerabilities can exist in other technologies, such as Web Services (WS), that are also subject to memory corruption attacks where components, such as XML parsers, are targeted versus web server binaries [27]. Although these infrastructure flaws are serious, we have chosen to focus on SQL injection attacks due to their alarming growth rate compared

to traditional flaws (shown in Figure 5).

Even so [10] indicate there is less attention from the research community on security problems at the web application level than on detection and prevention of these other more traditional memory corruption (buffer overflow, format string) type of attacks. This lack of focus on application level security is seen to further compound the problem as attackers have increasingly targeted application level vulnerabilities. As the lower level components have matured, they become harder to compromise.

OWASP illustrates the general severity of these application level weaknesses by a risk classification system shown in Figure 4. Each of these types of application flaws opens an application to differing types of exploits, with varying levels of impact and difficulty to accomplish.

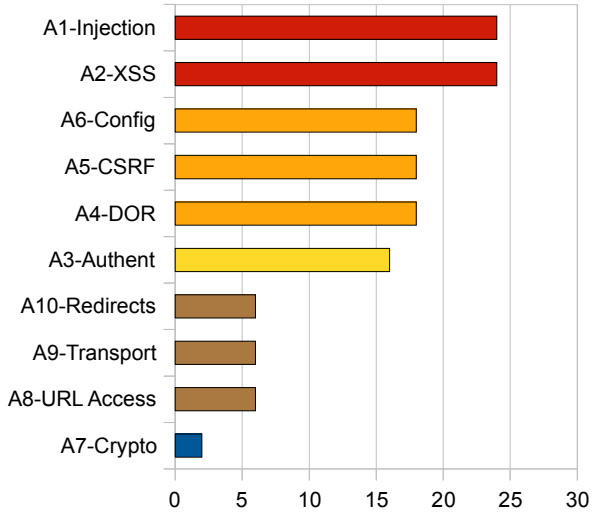

Figure 4: OWASP Vulnerability Risk Score

The OWASP ratings take into account several factors seen in typical threat risk assessment processes [24,28], including threat agent, exploitability, prevalence, detectability, and impact. This evaluation formula reveals where an organization should focus vulnerability discovery and remediation efforts, since it takes into account the value of a system to the business (impact) and ease of executing a specific attack. OWASP indicates “even egregious software weaknesses may not present a serious risk if there are no threat agents in a position to perform the necessary attack or the business impact is

negligible for the assets involved." [24] The focus of this paper is on assets that are deemed critical to an organization. Therefore, there needs to be some way to reduce the impact of SQL injection exploits and ultimately the vulnerabilities that are exploited.

Through the National Institute of Science and Technology (NIST) National Vulnerability Database (NVD), the US federal government offers statistics and information regarding known vulnerabilities in a standardized format to support vendor neutral vulnerability assessment, reporting, and management. Over the period from 2002 to 2008, the NVD showed a 1,561% increase in the number of known vulnerabilities classed as SQL injection flaws. Compared to buffer overflow vulnerability statistics that show a 563% increase from 1997 to 2008, the NVD data reveals a dangerous trend where SQL injection flaws have overtaken memory corruption flaws, such as buffer overflow and format string attacks, as a potential attack vector. See Figure 5.

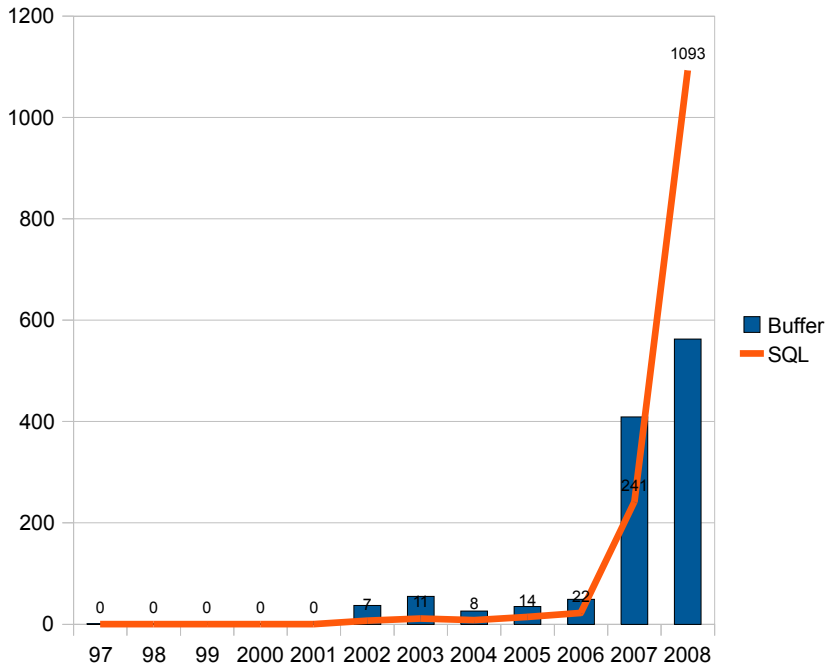

Figure 5: NVD SQL Injection vs Buffer Overflow Flaws

SQL injection flaws are one of the most serious threats to key services that are consumed over

the Internet today.

2.3 Exploit and Mitigation of Vulnerabilities

Although vulnerabilities may exist within a given computer system, a threat is not faced by an organization until the vulnerability is recognized and an exploit is available to take advantage of the vulnerability. Once a vulnerability is known, especially by malicious attacker communities, an exploit can be developed that enables an attacker to inject some form of malicious payload onto the vulnerable system. These payloads usually establish an unauthorized communication path or unintended access, allowing the attacker to control the victim system.

Often when an exploit allows loading and running a payload on a victim system, the payload will be running with the same credentials as the process that was running a vulnerable program. In cases where a system has been security hardened, this may be a user that has limited permissions and access on a system. Many systems are not hardened, and the exploit is especially serious if the vulnerable program is running with system administrative privileges. Gaining administrative access in this way is referred to as vertical privilege escalation, where a normal (or unauthorized) user has gained some advanced or higher level access. Where a normal or unauthorized user gains access to another normal or non-administrative user, it is referred to as a horizontal privilege escalation. Both types of unauthorized privilege give the attacker a greater foot-hold on the victim system from where additional probing and attacks can be launched.

Attackers can exploit both infrastructure and web application vulnerabilities by using powerful exploit frameworks such as Metasploit, CoreImpact, sqlninja, Immunity Canvas and Web Application Audit and Attack Framework (wa3f). Hackers use these sophisticated tool kits to discover and exploit weaknesses in web applications and the underlaying infrastructure. These frameworks are powerful enough to automatically discover web applications, assess their weaknesses, and launch attacks all with

a single mouse click. Some of these frameworks are especially dangerous, since they can allow a successful attacker to download advanced in-memory payloads that act as an agent, effectively positioning the attacker on the victim system, which is usually behind a network firewall. This foothold on an initial system can lead to the attacker successfully chaining subsequent attacks to gain access to increasingly more sensitive networks within an organization.

Specific exploits that target all layers of a web application, including network devices, server operating systems, web servers, application servers, databases, and especially web applications themselves, can be seen through security vulnerability reporting websites such as Bugtraq (www.securityfocus.com/archive/1), Secunia (www.secunia.com/advisories), Securiteam (www.securiteam.com) in addition to the centralized NVD (nvd.nist.gov).

Vulnerabilities found in web applications typically cannot be mitigated by using network firewalls since this level of firewall does not understand the application layer traffic. Network firewalls do little to stop attacker reconnaissance using well formed HTML, XML or SOAP requests, leaving web applications that are only protected by network firewalls vulnerable to attack at the HTTP application layer [9,12].

Figure 6 illustrates typical network traffic based filtering points protecting a web application with network firewalls that filter all traffic except for well known web application protocols (HTTP and/or HTTPS).

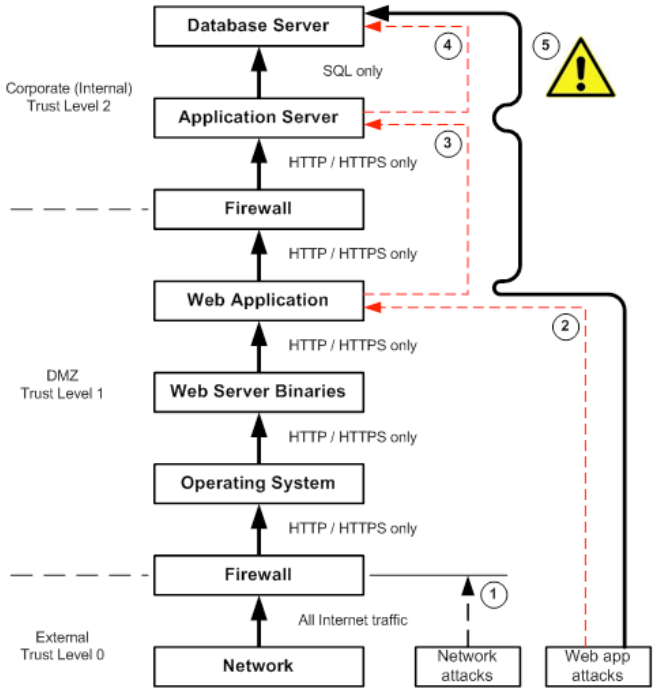

Figure 6: Web Application Network Stack

Case 1 in Figure 6 shows general network scans and attacks being blocked by the network firewall. Attacker reconnaissance such as port scanning and probing for listening network services (SSH, SNMP, telnet, ftp, etc) is blocked. Normal user access to web applications through HTTP or HTTPS traffic is allowed, since it traverses from the external network, through the firewall to the operating system on the server hosting the web server binaries, and finally to the web application itself. Weakness in the protection afforded by the firewall is seen in case 2, where there are attacks against the web application itself; allowed through the firewall as valid HTTP or HTTPS traffic. Once an attacker discovers a vulnerability in the web application, exploits can be run against the web application, resulting in unauthorized access, possibly to the web server or operating system. This foot hold allows the attacker to begin chaining attacks and compromises together, moving their point of presence to the newly exploited system. As additional vulnerabilities are discovered in more trusted systems, the attacker can move their foot print further into the organization, as shown in cases 3 and 4. Once a compromise in case 4 is completed, the attacker has a virtual direct connection through two network firewalls to

sensitive corporate resources, such as a SQL database server (case 5). Chaining attacks from exploited hosts and the attacker's memory based agents, such as Metasploit, Core Impact, or Immunity Canvas allow an attacker to compromise a web application, load an in-memory agent on the web server, then probe for unrelated weaknesses on the application server that can be exploited to push exploit agents further into the organization, eventually hitting the core network. Logs from the firewalls, operating systems, and web servers would be inadequate to understand the extent of the intrusion retrospectively.

Common practices to mitigate these web application vulnerabilities have been developed by industry professionals such as Mitre, SANS and OWASP. Both OWASP and CWE produce and maintain Top Ten style lists of the most widespread and dangerous programming errors that often lead to successful exploits by attackers, as well as recommended design patterns developers can use to prevent or mitigate the weaknesses that lead to exploitable vulnerabilities. Table 2 shows a summary of the mitigation techniques recommended by CWE.

Table 2: CWE Monster Mitigations

ID	Description
M1	Establish and maintain control over all of your inputs
M2	Establish and maintain control over all of your outputs
M3	Lock down your environment
M4	Assume that external components can be subverted, and your code can be read by anyone
M5	Use industry-accepted security features instead of inventing your own
GP1	(general) Use libraries and frameworks that make it easier to avoid introducing weaknesses
GP2	(general) Integrate security into the entire software development lifecycle
GP3	(general) Use a broad mix of methods to comprehensively find and prevent weaknesses
GP4	(general) Allow locked-down clients to interact with your software

Mitigation technique M1, “Establish and maintain control over all of your inputs”, identified by CWE can be used to guard against various type of injection attacks on web applications. Injection attacks, including SQL injection, are possible when the application developers do not sufficiently

inspect and filter out hazardous characters from input streams accepted from end users. These mitigations can be included in an organization's System Development Lifecycle (SDLC) through the adoption of standardized architectural security design patterns. Use of these design patterns can eliminate entire classes of vulnerabilities across all web applications deployed within an organization. Coordinated use of most, if not all, of the mitigations build up a defense in depth hardening of an application.

2.4 SQL Injection Attacks

[29] observe that web applications typically use relational databases to persist information that is used to dynamically build HTML pages that are presented to users. Typical web applications use a multi-tiered approach with a presentation tier (web server), business logic tier (application server) and a database tier. The web and/or the application server sends low level SQL queries to the database server in order to build the HTML needed for user presentation, typically through application API calls such as JDBC. Problems arise when programmers do not handle user input properly, resulting in unintended direct user interaction with the database. Since low level string manipulation is used to dynamically build database SQL queries based on user input, if that input is not handled correctly, unintended modification of the SQL queries can happen.

These security problems occur when application developers fail to perform sanity checks on inputs from untrusted sources, such as user input fields, that are eventually used as operands of security sensitive operations such as authentication or administrative tasks [10]. SQL injection attacks can take place when a user sends specially crafted input that results in query functions that are not intended by the programmer [29]. A clear example is given by [29]:

```
query = "SELECT * FROM accounts WHERE name='"
        + request.getParameter("name")
        + "' AND password='"
```

```
+ request.getParameter("pass") + "'";
```

This code is intended to authenticate a user, however it can be subverted when an attacker enters a value in the name field then `"'OR' a'='a'` into the password field. This specially crafted input generates the following query with the result of the conditional logic always returning true, allowing the attacker to bypass remaining authentication logic.

```
SELECT * FROM accounts WHERE name='admin' AND password='' OR 'a'='a'
```

Once an attacker confirms an application is vulnerable to SQL injection attacks, they can progress to enumerating the database schema, including probing to uncover columns in authentication tables.

Successfully modified SQL statements may simply generate an error message indicating that a syntactically incorrect query was passed to the database. Attackers will often use trial and error to reverse engineer the schema of a database table, allowing them to craft syntactically correct queries to either extract additional information or perform INSERT or UPDATE operations. Once an attacker can successfully alter or insert rows, it is often possible for the attacker to gain horizontal and/or vertical privilege escalation. Dependent on the application architecture, the attacker may only be able to execute queries as a normal user (horizontal escalation), however is often able to modify sufficient information in a user authentication table to gain administrative access (vertical escalation).

In some cases, the attacker may use INSERT or UPDATE capabilities to plant malware within the web site, infecting unsuspecting web clients in order to increase the volume of infected computers under control of the attacker. Alternatively, the attacker may use unauthorized SQL queries to extract sensitive information from the application, as was the case with the attackers which exploited web applications used by CardSystems to extract over 40 million credit card numbers.

Additional mitigation techniques that would make exploitation of SQL injection flaws more difficult include configuration of application servers, and web applications themselves, to avoid

displaying any system or database level error messages back to the client. These system error messages should always be captured and abstracted to an error message that is appropriate for the application context. Abstraction of messages in this manner substantially increases the difficulty faced by an attacker, since it deprives them from positive or negative feedback from their attack attempts.

CHAPTER 3 - ATTACK ANALYSIS

3.1 Background

Computer security incident response (CSIR) processes are used to prepare for and respond to various kinds of incidents including attacks. These processes include preparation, identification, containment, eradication and recovery stages [30]. Preparation may include: preventative controls such as implementation of network firewalls, operating system and web server hardening; detective controls such as enabling logging from network firewalls, operating systems, web servers, network intrusion detection systems; and corrective controls such as development of first responder capabilities and implementation of network intrusion prevention systems. These CSIR processes are intended to minimize the impact of an incident including detection and recovery when web applications have been attacked.

During or after the handling of an incident, digital forensics can be used to determine the extent to which the attack was successful [31]. After a web application has been attacked, investigators often begin by examining log files from security devices surrounding the web application and its supporting infrastructure. Traditional digital forensics [32] relies on log meta data relating to web transactions. These log sources typically include network and application firewalls, operating system and web server logs, application logs, and intrusion detection systems [33]. These logs are often insufficient for forensic purposes since they were only designed for system debugging purposes [34], even where the log data is intended for audit purposes. [15] indicate forensics and investigation always come after an attack, and as such, useful live evidence in the form of application network payload is lost [35], hindering the investigation. Where network recording capabilities do not exist, the investigator is forced to reconstruct events from incomplete data, often resulting in poor quality since they are left to deduce system or data changes and events from sparse forensic information [34]. With lack of detailed

information to illuminate the precise actions of the attacker, determination of exactly what the attacker did and on which systems is difficult, if not impossible.

```
1: Jun 10 14:00:07 fw2 kernel: [20567.917276] RULE 15 -- ACCEPT IN=eth1
   OUT=eth2 SRC=10.10.21.12 DST=10.10.22.12 LEN=60 TOS=0x00 PREC=0x00
   TTL=63 ID=55805 DF PROTO=TCP SPT=32965 DPT=80 SEQ=2928938648 ACK=0
   WINDOW=5840 RES=0x00 SYN URGP=0
2: Jun 10 14:40:34 fw2 kernel: [22994.577659] RULE 16 -- DENY IN=eth1
   OUT=eth2 SRC=10.10.21.12 DST=10.10.22.12 LEN=60 TOS=0x10 PREC=0x00
   TTL=63 ID=42817 DF PROTO=TCP SPT=33505 DPT=3306 SEQ=2188857246 ACK=0
   WINDOW=5840 RES=0x00 SYN URGP=0
```

Listing 1: Firewall log meta-data

Listing 1 shows logs from a firewall where `attacker (10.10.21.12)` was able to successfully get access to the web server and a subsequent denial for access to a MySQL service on `web11 (10.10.22.12)`. Nothing further is known from the firewall logs other than which hosts and services node `attacker` specifically tried to access. Even with the first entry showing a firewall accept of a TCP connection to node `web11`, there is no indication that a web server was running on that server or if the TCP connection was successful. This lack of clarity from a single log source highlights the forensic need to correlate many differing vantage points within the organization's network. Typical log data for a successful web application access is shown in Listing 2, where the investigator can see the firewall permitted the access (`ACCEPT 10.10.21.12 > 10.10.22.12 TCP port 80`) and the web server successfully served a request for a file named /etc/passwd123. The only reason we can deduce this was a malicious request is due to the contents of the HTTP GET URI. This example illustrates the difficulty of conducting forensics solely through use of meta-data; namely there is no clear indication if the web server completed the request and sent the requested data back to the attacker, despite the HTTP 200 success code. There is no way, through meta-data, to be sure what was the response of the web server. The response could have contained the unauthorized file or it could have been a properly formatted HTML page indicating the web server could not carry out the client request.

```
1: Jun 10 14:59:01 fw2 kernel: [24101.376743] RULE 11 -- ACCEPT IN=eth1
   OUT=eth2 SRC=10.10.21.12 DST=10.10.22.12 LEN=60 TOS=0x00 PREC=0x00
   TTL=63 ID=52256 DF PROTO=TCP SPT=33682 DPT=80 SEQ=2502871064 ACK=0
   WINDOW=5840 RES=0x00 SYN URGP=0

2: 10.10.21.12 - - [10/Jun/2010:14:59:01 -0600] "GET
   /mutillidae/index.php?page=../../../../etc/passwd123 HTTP/1.1" 200
   6576 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15)
   Gecko/2009102814 Ubuntu/8.10 (intrepid) Firefox/3.0.15"
```

Listing 2: Successful web access meta-data

Listing 3 further illustrates the dangerous gap between transaction meta data recorded by the web server followed by a HTTP network conversation payload capture, where the network recording shows the password variable containing a SQL injection attack `' or 1=1 --` that was not revealed in the web server transaction log. Without a network capture showing the HTTP POST request, the actual contents of the POST message is lost, leaving the investigator to guess as to the outcome of the transaction.

```
1: 10.10.21.12 - - [28/Apr/2010:13:17:14 -0600] "POST
   /mutillidae/index.php?page=login.php HTTP/1.1" 200 2237
   "http://web11/mutillidae/?page=login.php" "Mozilla/5.0 (X11; U; Linux
   i686; en-US; rv:1.9.0.15) Gecko/2009102814 Ubuntu/8.10 (intrepid)
   Firefox/3.0.15"

2: POST /mutillidae/index.php?page=login.php HTTP/1.1
   Host: web11
   User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15)
   Gecko/2009102814 Ubuntu/8.10 (intrepid) Firefox/3.0.15
   Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
   Referer: http://web11/mutillidae/?page=login.php
   Cookie: showhints=1
   Content-Type: application/x-www-form-urlencoded
   Content-Length: 68
   user_name=admin&password=%22%27+or+1%3D1+--+%22&Submit_button=Submit
```

Listing 3: Web server log meta-data and HTTP POST data

Using log meta-data alone to determine the attack vector and the extent of the intrusion would be extremely difficult, especially if the attacker had employed multiple systems, such as using a compromised web server as a stepping stone to access a corporate database server [36]. Traceability of

actions committed across several systems with only meta data is complicated by the termination of TCP communication streams on each system, where it would be difficult to establish a correlation between traffic terminating and originating on one system.

[32] discuss methods to collect and analyze evidence of attacks, through network forensics, that could be used to reveal successful and unsuccessful attacks on web applications. [37] take this approach further, by incorporating intrusion detection system controlled application payload recording, preserving suspicious activity for inclusion in subsequent investigations. Use of application payload data enables an investigator to determine how a web application was attacked, the weakness that was exploited, and possibly subsequent activities by the attacker.

In summary, only using meta information leaves a gap where the investigator will be unlikely to complete a forensic analysis to determine how an attack was successful and what subsequent activities the attacker attempted and completed. Therefore, a method to incorporate web application HTTP transaction data is needed to reveal the entire attack and extent of intrusion.

3.2 Network Traffic Evidence

The increasing cost and complexity of investigating computer crimes including computer intrusions, intellectual property theft, and financial fraud [38] has pressed information security professionals to turn to computer networks for additional sources of evidence in addition to the traditional meta-data sources. [32] indicates a goal of intrusion detection systems is to provide digital evidence to support criminal and civil legal proceedings, however, like other meta-data, intrusion detection data also suffers from the inability to comprehensively rebuild a digital crime scene. Network recording is an archive of the raw traffic streams flowing across an organization's networks that represent transactions between clients and servers [37] and enables retrospective analysis of network transactions that have happened in the past. Where meta-data regarding these transactions is stored in

log files from firewalls, operating systems and application, the raw traffic stream is a real-time archive of the actual payload that traversed the organization's networks.

[38] indicates that captured network traffic is a compelling form of digital evidence since it can be used to show the actions of an offender like a surveillance video tape of a crime. Capture of network traffic also provides increased detective safeguard integrity, as it defeats attempts by an attacker to hide or destroy evidence of an intrusion since the attacker can only modify the victim system, not the network stream data that was recorded out-of-band from the victim system [38]. Although quality and admissibility of various forms of digital evidence is still variable [34,39], appropriate data must be available to reconstruct a crime scene such as a SQL injection attack. Since meta-data alone does not provide the ability for an investigator to answer the questions *who, what, where, when* and *why* [39], the increased level of data provided by network recordings is needed. Use of network recordings is key to improve the conclusions drawn and decisions made in response to an incident, since incomplete or inconclusive findings can result in either offenders not being held accountable or innocent individuals unjustly accused [34]. A substantial benefit from network recordings is the ability to fully understand a weakness that was attacked in order to determine the extent of a potential intrusion, answering the questions “What did the attacker do? What else did the attacker get access to?”. Additionally, understanding the exploited flaw enables the organization to mitigate the weakness in order to reduce or eliminate the continued threat of additional attacks. When an attack can be fully understood, the application owner can take affirmative steps to ensure the same attack does not succeed again; either by installing mitigating safeguards or by correcting the exploited flaw.

Using network traffic as digital evidence does have numerous drawbacks including possible errors in reconstruction of the traffic payload [38], and sheer volume of unconstrained network recording, both in terms of network traffic, required processing resources and disk space [40-42]. Tools

for managing network recordings are still in a nascent state where investigators need to apply specialized skills to understand and interpret the context of network flows, even assuming suspicious flows can be isolated from the library of recorded streams. Use of network recording needs to be coupled with the ability to selectively record only suspicious traffic as well as reduce the volume of traffic recorded to the maximum extent possible, while maintaining the forensic usefulness of the data [37,42,43].

Users of an organization's computer resources can be subject to an Acceptable Use Policy of those organizations that may indicate users have no reasonable expectation of privacy. Despite this, Elizabeth Denham, the Privacy Commissioner of Canada, points out the requirement to follow relevant privacy legislation in the jurisdiction in which the network recording is taking place. Ms. Denham gives the example of a complaint filed with the Privacy Commissioner's office where deep packet inspection (DPI) was in use at an Internet Service Provider (ISP). In this case, network IP addresses recorded by the ISP were found to be classed as personal information. The findings of the Privacy Commissioner were the use of DPI was justified since the ISP was using the inspections for managing their networks and taking precautions to protect the collected information only retaining it for the minimum period of time reasonable [44].

Bernhard Otupal, Assistant Director Financial and High Tech Crime, Interpol, indicates there is currently still a lack of internationally recognized standards for handling digital evidence such as network recordings. Where organizations are looking to use network recordings in civil cases or turn the evidence over to law enforcement agencies for possible criminal prosecution, there are no clear requirements for demonstrating the validity of IDS or network recording data for legal purposes [45].

The remainder of this section discusses the ability to selectively record traffic deemed suspicious and apply algorithms to determine which portions of the traffic to record while preserving

forensic value of the data.

3.3 Network Recording

[42] indicate pervasive network packet recording to capture packet headers as well as full packet payload can provide invaluable information for both troubleshooting and security investigations. However, integration of network recording capabilities into an infrastructure to support digital forensics presents substantial challenges that are compounded as the volume of network traffic increases [42]. Sites with even moderate Internet bandwidth and gigabit internal networks will not be able to brute-force record all network traffic practically and retain for a reasonable timeframe, given current commodity hardware sizes. Even recording only the headers and payloads of HTTP and HTTPS conversations, in order to investigate SQL injection attacks, will often require the storage of potentially massive data volumes. To address this limitation, the proposed solution leverages IDS functionality to selectively enable network recording of suspicious transactions and uses the high performance network recording Time Machine project that exploits the heavy-tail characteristic of TCP traffic [37,42] for maximum retention of network flows.

[42] show the bulk of traffic in high volume streams typically occurs with a relatively small number of connections. Most network connections are short, with a small number of large connections making up the majority of the network traffic [42]. During an investigation, the beginning of both short and long network connections is crucial to determining identifying information, such as source and destination IP address, protocol, source and destination port, and relevant application level protocol data. Sufficient data needs to be captured to understand the nature of the transaction, however the entire stream does not typically need to be captured. The Time Machine project implements an algorithm to capture sufficient detail at the beginning of network streams while discarding the heavy tail [42] or bulk of the transaction data, referred to as a *cut-off* point. The TM project is configurable enough,

however, to record substantial or complete streams that are deemed to be of high importance to an organization. This approach results in substantial data recording reduction while maintaining enough of the transaction for extended identification and to illustrate an attacker's actions. Since for forensic purposes, the packet headers and a relatively small amount of the payload from the beginning can accurately fingerprint connection patterns [43], the overall volume of traffic payload recorded can be reduced by as much as 96% [42]. An example of this effectiveness is a dramatic reduction of 32TB of raw 100Mbps Internet feed to a total of 1.3TB of network recordings covering an entire month. Selection of only suspicious traffic for recording can reduce the storage required even further.

The Bro-IDS intrusion detection system is used since it has built-in interfaces to connect to and leverage the recording resources of a TM [37]. This integration between Bro-IDS and TM is used to dynamically change recording parameters set within TM. Bro-IDS offers similar features to many commercial and well known open source IDS systems and can import signatures from the well known open source Snort IDS project. The policy language developed by the Bro-IDS project is geared to the specialized task of analyzing and monitoring network traffic, and includes event level linkages to a TM, enabling retrospective analysis of suspicious traffic.

The proposed solution uses TM to record and manage network traces of both static traffic patterns predetermined to be of interest to security administrators, as well as dynamically captured traces determined to be of interest by the Bro-IDS system. Figure 7 illustrates the overall configuration of the proposed solution, with network traffic feeding both Bro-IDS and the TM.

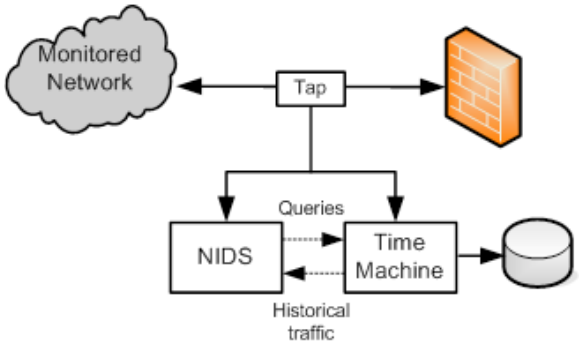

Figure 7: Coupling NIDS and TM

While IDS systems can capture and display slightly more detailed meta data regarding a particular application transaction, by themselves they suffer from an inability to retrospectively analyze network traffic that has already taken place [37,42,46]. As shown in Figure 8, the Bro-IDS system is configured to request the TM record specific traffic flows, as well as respond to queries regarding traffic that may be associated with a suspicious pattern. These selective or dynamic filters are used to reduce the data volume recorded while still capturing streams of interest. Use of Bro-IDS to trigger network recordings implements a dynamic filter capable of capturing sufficient information to reconstruct crucial data regarding a suspect transaction.

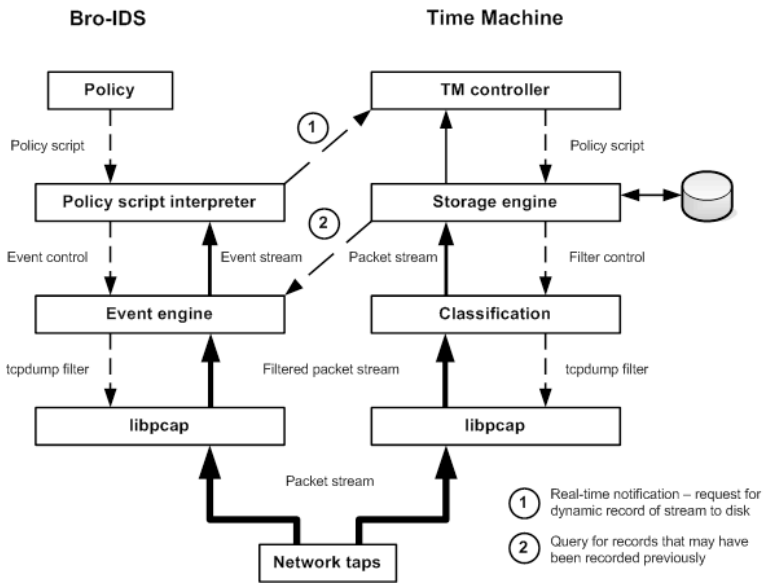

Figure 8: Bro-IDS and TimeMachine structure

Case 1 shown in Figure 8 illustrates Bro-IDS triggering TM to record network traffic of interest,

because Bro-IDS determined the traffic to be suspicious. TM may already be configured statically to record that pattern of traffic due to static filter policy set by security administrators. Case 2 shown in Figure 8 illustrates Bro-IDS querying the TM for traffic that needs further analysis, allowing the IDS to retrospectively analyze host traffic that has happened in the past, but still within the retention capabilities of the TM.

Policy statements were configured instructing Bro-IDS to trigger recording of all HTTP sessions where specific SQL injection attack patterns were seen. Initial stages of Bro-IDS and TM integration have been accomplished by cooperation of the respective open source project teams and the effect is for Bro-IDS policy statements to translate into tcpdump filters for both Bro-IDS and TM. The tcpdump filters act as the first level of traffic reduction, where the Bro-IDS event engine analyzes attributes within the incoming packets for a policy match. In the case where there is a policy match, Bro-IDS triggers the TM to begin recording the selected TCP/IP or UDP/IP stream. This selective filtering enables substantial data volume reduction, which can be further reduced by the TM through application of "heavy-tail" recording techniques. These recordings are what enables retrospective investigative ability, adding to the traditional log file entries.

3.4 Attack Lab Configuration

The web application attack lab was configured as shown in Figure 9. All nodes were created as virtual machines within a VMware Fusion 3.0 environment on an Apple MacBook Pro running Mac OS X 10.6. Attacks were run from the Attacker node (`attacker`), running Backtrack 4.0. The web server (`web11`) was configured with Apache 2.2, running the vulnerable Mutillidae PHP application that uses a MySQL database running on a database/application server (`app11`).

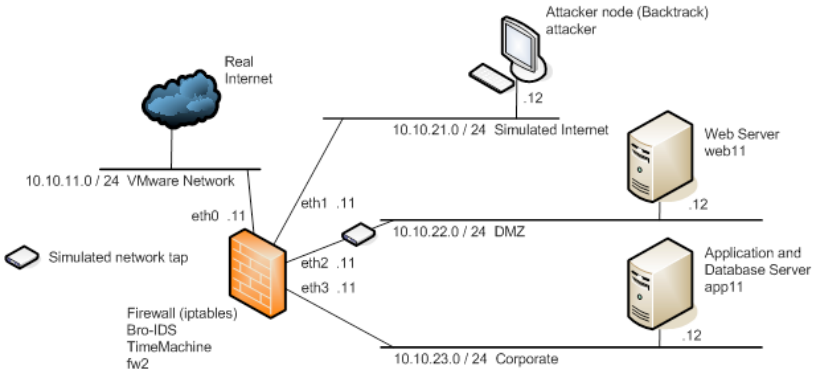

Figure 9: Attack lab network

As shown in Figure 10, the firewall was configured to use stateful iptables rules to provide access control typical for web applications from the simulated Internet to the DMZ and from the DMZ to the Corporate network.

fw2-internet-dmz (1 rules)							
net-SimulatedInternet	net-PublicDMZ	TCP http TCP https	All		Any		Access from simulated Internet to DMZ web server
fw2-dmz-core (2 rules)							
net-PublicDMZ	net-Corporate	TCP http TCP mysql	All		Any		Access from DMZ to application server on the inside
net-Corporate	net-PublicDMZ	TCP ssh	All		Any		Access to DMZ systems
vmnet-common (2 rules)							
net-freedom-vmnets	net-LocalVmNAT	DNS Useful_ICMP ICMP ping request	All		Any		Allow lab VMs ICMP and DNS to the VMware host
net-freedom-vmnets	net-freedom-vmnets	TCP http DNS UDP ntp TCP https Useful_ICMP ICMP ping request TCP ssh	All		Any		Allow lab VMs to get updates from the real Internet
default-drop (1 rules)							
Any	Any	Any	All		Any		

Figure 10: Attack Lab Firewall Rules

The Apache web server (`web11`) was configured to log at verbosity level `info`, capturing information about client HTTP requests including:

- remote host IP address

- remote user identification (if available)
- user ID of requestor as determined by any HTTP authentication
- date/time
- URI request line
- server status sent back to client, and
- size of the object returned to client in bytes.

This is typical of fields captured in web server logs, however the default verbosity level is `warn`, reducing the types of messages written to the web server access log. Information level logging on an energy trading exchange was found to generate approximately 162,000 log entries per trading day per web server. This is well within modern log management system capacities.

The iptables firewall was configured to log on all accesses as shown in Figure 10, where web requests from the Simulated-Internet network to the Public-DMZ network and database requests from the Public-DMZ to the Corporate network were recorded. Even with a simple three-tier web application, a single web server access can generate many requests from the application tier to the database tier, however these log volumes are still within the capacity of modern log management systems.

Bro-IDS was integrated with the TM as shown in Figure 8, with the simulated network tap implemented within the firewall (fw2) on the DMZ interface (eth2). For this paper, the TM was statically configured to capture all traffic that was seen using TCP port 80 (HTTP) and TCP port 443 (HTTPS), as well as respond to requests from the Bro-IDS system to record based on criteria provided by the IDS system.

Static Time Machine configuration consists of:

- filter statements to match network traffic
- a precedence level that removes ambiguity when matching classes with slightly overlapping criteria
- a cutoff that specifies how much of the payload to record
- how much system main memory to allocate
- maximum disk store size, and
- maximum size of all files with the disk store for that class.

Two static filters were established to ensure all relevant traffic was captured, where in practice, only high risk traffic would be statically recorded.

```
class "http" { filter "tcp port 80"; precedence 50;
   cutoff 20k; mem 150m;
   disk 10g; filesize 1m; }

class "https" { filter "tcp port 443"; precedence 50;
   cutoff 50k; mem 200m;
   disk 20g; filesize 1m; }
```

Listing 4: Time Machine configuration

Listing 4 shows static configuration of the Time Machine, where all traffic with a TCP destination port of 80 (HTTP) has the first 20kB recorded in pcap format. Up to 150MB of main memory will be allocated for handling this class of traffic, and through the use of sequential 1MB files, up to 10GB of disk will be used to store the traffic. Filter statement may be as complex as needed to define static conditions where traffic streams must be recorded. Recording class "https" specifies the first 50kB of each stream must be recorded, allows for more main memory, and additional total disk storage.

3.5 Reconstruction of SQL injection attack

The Mutillidae web application is a sample vulnerable PHP application that can be used to teach about security weaknesses common in web applications, and follows the OWASP Top 10 vulnerability list giving examples of the most common Top 10 programming errors. In the Mutillidae application, the HTML Login page contains code that is vulnerable to SQL injection attacks.

Following the general steps of an attack [47], we run a vulnerability assessment on the application as part of an attacker's reconnaissance of the application. Several web application vulnerability assessment and attack frameworks exist including:

- Metasploit
- SQL Ninja
- Core Impact
- Immunity Canvas
- IBM AppScan, and
- w3af (Web Application Attack and Audit Framework).

The w3af tool was chosen for ease of use, small footprint, and orientation to web testing. Weaknesses discovered by w3af include manipulation of the password parameter used in a POST operation from within of the Mutilladae forms, shown in Listing 5.

```
SQL injection in a MySQL database was found at:
"http://web11/mutillidae/index.php?page=user-info.php", using HTTP
method POST. The sent post-data was:
"view_user_name=LzGulbg&password=d'z"0&Submit_button=Submit". The
modified parameter was "password". This vulnerability was found in the
request with id 3034.
```

Listing 5: HTML POST vulnerability in Mutillidae

w3af used the POST request shown in Figure 11 to test the application input parsing of POST parameters. The results of this test, shown in Figure 12, confirmed the password parameter has been successfully modified to include characters that could be used in a SQL injection attack. Attackers will often inject characters into POST or GET parameters in hopes of inducing some form of SQL query error by the database that serves the application. If an application is not hardened properly, including performing user input validation and control of error messages, errors such as those shown in Figure 12 are transmitted back to the client. An attacker will use this error message to begin probing the application database to understand the schema and eventually attack the application.

Figure 11: w3af Assessment Request

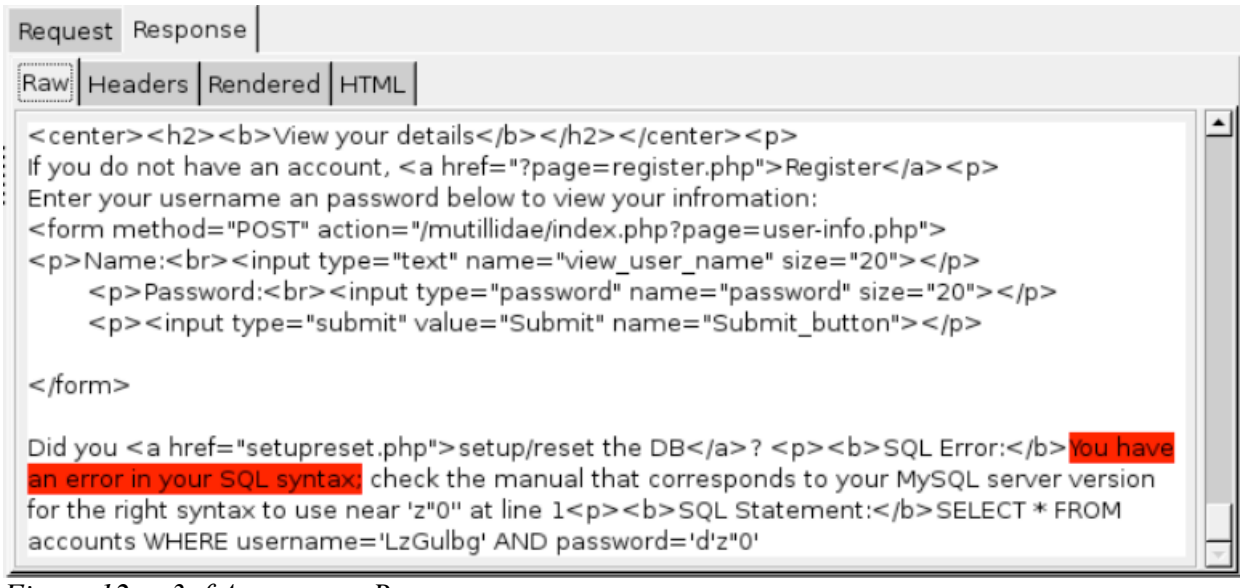

Figure 12: w3af Assessment Response

Attackers can send SQL queries through the vulnerable POST parameters to enumerate the database schema enabling more complex queries that will extract confidential information from the database at will, inject data into the database (such as unauthorized user credentials), or perform some malicious action, such as dropping or truncating tables. Executing arbitrary SQL queries against the database is accomplished through building additional components into an existing query, see Listing 6.

```
1: SQL Statement:SELECT * FROM accounts WHERE username='admin' AND
   password='' union select username,pass from core_user -- '

2: SQL Error:Table 'owasp10.core_user' doesn't exist

3: SQL Statement:SELECT * FROM accounts WHERE username='admin' AND
   password='' union select username,pass from accounts -- '

4: SQL Error:Unknown column 'userid' in 'field list'
```

Listing 6: SQL Injection queries enumerating the database schema

Where the proposed solution has been deployed, the Bro-IDS system will use network traffic payload analysis to discover signs of possible SQL injection attempts including when patterns from Table 3 are seen in HTTP GET or POST queries.

Table 3: SQL Injection Attack Patterns

Type	SQL Component
DDL	CREATE, DROP, TRUNCATE
DML	DELETE, INSERT, SELECT, UPDATE
Functions	AND, ASCII, CAST, CHAR, CONVERT, DECLARE, EXEC, LENGTH, OR, SUBSTRING, UNION, VERSION
Syntax	1=1 A single quote at the end of a URL with no other single quotes URLs with one single quote at the end of a normal GET value

Bro-IDS was configured to recognize password parameters within HTTP POST operations and both trigger a network recording of the HTTP session, as well as to write an alarm regarding the event. Although out of scope of this paper, the Bro-IDS policy language is powerful and granular enough to enable complex evaluation of network payload data. An alarm was raised by the IDS system indicating a possible SQL injection attack signs had been seen, shown in Listing 7.

```
1: t=1276192836.909579 no=HTTP_SQL_Injection_Attack
   na=NOTICE_ALARM_ALWAYS es=bro sa=10.10.21.12 sp=40303/tcp
   da=10.10.22.12 dp=80/tcp p=80/tcp method=POST
   url=/mutillidae/index.php?page\=login.php num=200
   msg=10.10.21.12/40303\ >\ 10.10.22.12/http\ %bro-14:\ POST\
   /mutillidae/index.php?page\=login.php\ (200\ "OK"\ [6324]\ web11)
   tag=@8e-6e14-7
```

Listing 7: Raw Bro-IDS alarm record indicating a possible SQL injection attack

Based on the alert from the IDS system, the TM can be queried (Listing 8) for network pcap data that would show the possible attack through the tm-query function. Interfaces are available to automatically display the network pcap data within a Bro-IDS alert, and could be readily integrated into other network security tools.

```
./tm-query --ip 10.10.22.12 localhost web11-20100609-1415.pcap --time 5m
```

Listing 8: TM Query to Extract pcap Data

The network data retrieved from the TM can show not only the suspicious traffic that triggered

the recording but all other systems with which the suspect interacted. The large reduction in traffic recording volume by using *heavy-tail* elimination enables a substantially larger number and type of network connections to be gathered for later analysis. The main limitation is sufficient placement of network taps to capture suspicious actions throughout the organization. As shown in Figure 9, in order for TM to capture network transactions, the source or destination must be in the DMZ. If there was suspicious traffic originating in the Corporate network, then flowing to the Internet, such as the case with an infected machine on the Corporate network, the single network tap on the DMZ network would not be able to intercept that traffic.

Once the TM has been queried to extract recordings of the suspicious transactions, the pcap data can be imported into a tool, such as WireShark, to reconstruct the HTTP payload including any GET or POST contents, as shown in Figure 13. Highlighted sections in the traffic reconstruction show the malicious attack, namely:

- Section 1 shows the HTTP POST query;
- Section 2 shows the attacker (src) and victim server (dst);
- Section 3 shows the TCP protocol used (HTTP);
- Section 4 shows the HTTP POST header; and
- Section 5 shows the actual SQL injection attack in the *password* parameter.

Analysis of subsequent HTTP/1.1 200 return HTML indicates the attacker succeeded in bypassing authentication and becoming administrator in the application.

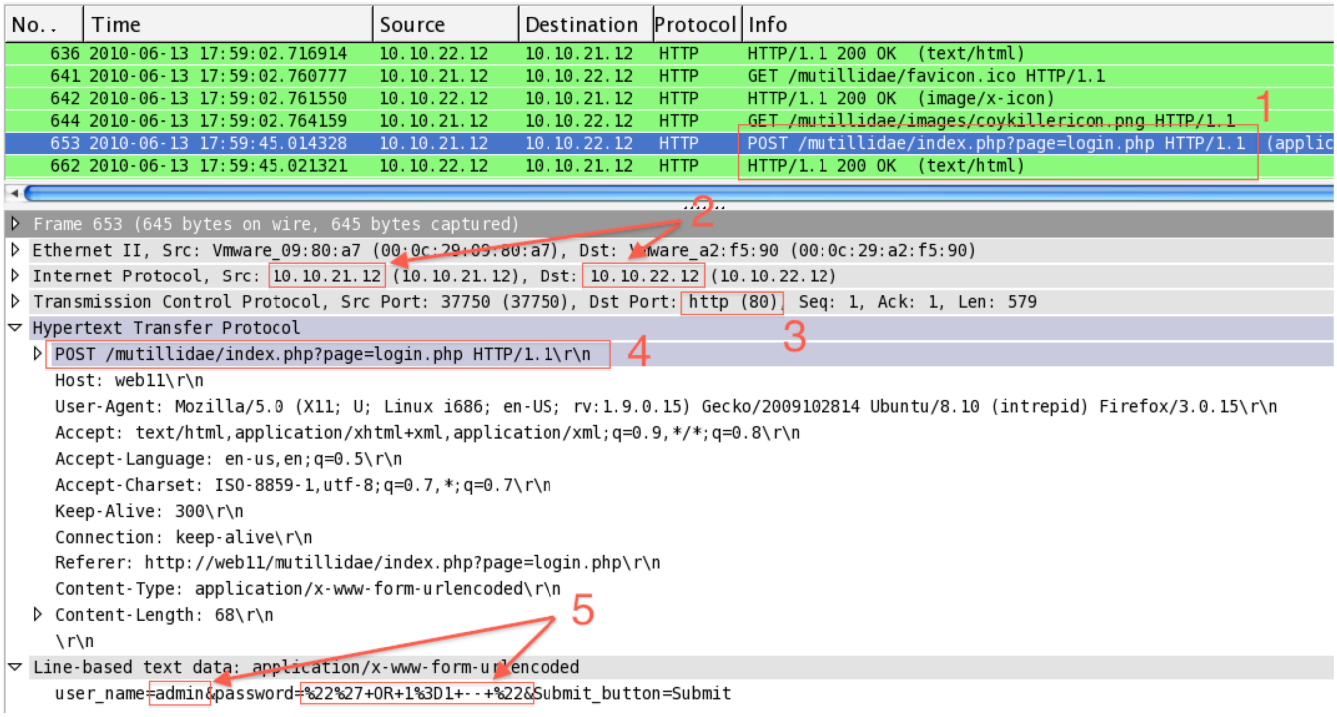

Figure 13: Wireshark analysis of Time Machine pcap data

Investigators attempting to reconstruct this attack from firewall and web server logs would have the entries shown in Listing 9 to attempt to deduce what had happened.

```
1: 10.10.21.12 - - [13/Jun/2010:17:59:45 -0600] "POST
   /mutillidae/index.php?page=login.php HTTP/1.1" 200 5988
   "http://web11/mutillidae/index.php?page=login.php" "Mozilla/5.0 (X11;
   U; Linux i686; en-US; rv:1.9.0.15) Gecko/2009102814 Ubuntu/8.10
   (intrepid) Firefox/3.0.15"

2: Jun 13 17:59:45 fw2 kernel: [24101.376743] RULE 11 -- ACCEPT IN=eth1
   OUT=eth2 SRC=10.10.21.12 DST=10.10.22.12 LEN=60 TOS=0x00 PREC=0x00
   TTL=63 ID=52256 DF PROTO=TCP SPT=33682 DPT=80 SEQ=2502871064 ACK=0
   WINDOW=5840 RES=0x00 SYN URGP=0
```

Listing 9: Web server log meta-data and HTTP POST data

As shown by the evidence captured by TM, in the actual SQL injection attack, the vulnerable password parameter was assigned a value as `password=%22%27+OR+1%3D1+--+%22` , resulting in the SQL injection attack `' OR 1=1 --` modifying the SQL query to `SELECT * FROM users WHERE username='admin' AND password='' OR 1=1 --`. This allows the attacker to successfully bypass authentication for the administrative user.

Wireshark was used to perform the HTTP protocol analysis - where this level of historical detail would not have been possible without the network recording capability, and the attack could have gone undetected. The forensic value of triggered and static TM recordings prove to be invaluable to substantially improving the reconstruction of SQL injection attacks.

CHAPTER 4 - RECOMMENDATIONS AND CONCLUSION

4.1 Recommendations

Deploy network recording infrastructure depicted in this paper to enable comprehensive analysis of SQL injection attacks against web applications. Positioning network taps as close as possible to external ingress and egress points as possible enables maximum coverage of suspicious external traffic. Leveraging Time Machine functionality enables organizations to capture significant volumes of network traffic that is forensically relevant.

Developers of web applications should follow attack mitigation advice given by industry subject matter experts, such as OWASP, CWE, and SANS. Prevention of attacker access to SQL databases supporting web applications is a powerful front line of defense.

4.2 Conclusion

Web applications are under attack at the network application layer where traditional network security is ineffective and cannot thwart SQL injection attacks. Despite the ability of web application firewalls to detect and prevent some types of application attacks, including some SQL injection attacks, web application owners need defensive tools to analyze successful attacks against the SQL databases that support these web applications. This paper has demonstrated techniques that can be used to clearly detect and analyze SQL injection attacks, and enables web application owners to more effectively protect critical web applications through

- Improved understanding of application vulnerabilities and extent of attacks, and
- Enabling continuous improvement of web applications once attack vector and pattern are known.

The network traffic recording techniques presented in this paper offer an effective method to

record and use high volumes of network traffic using commodity hardware, and enables levels of visibility into hostile actions within an organization not possible without these techniques.

Use of network recording presents possible challenges to an organization from handling perspectives. Collection and retention of network recordings may be subject to privacy protection legislation. Internationally accepted standards governing collection and handling of network recordings as forensic information do not currently exist. Where there is a need to present forensic information for support of criminal or civil legal action, organizations are advised to obtain guidance from law enforcement in the intended jurisdiction.

Application of the techniques illustrated in this paper to a variety of security tools, including intrusion detection and prevention, web application firewalls, security information and event management, and even web server software, could substantially improve the ability to proactively defend web applications. Coupled with current state of the art with reputation based web content filters, this technology could provide a positive confirmation of attempted exploits through detection of SQL injection attacks. Integration of Time Machine functionality with existing tools to provide deep near real-time retrospective analysis, enables a new class of attack identification and prevention.

The heavy-tail traffic reduction implemented by the Time Machine enables practical storage of forensic network data for relatively long periods of time. Use of artificial intelligence to expand on the reputation based decision making that is incorporated into state of the art web application filters could reduce the possibility of web exploits even further.

In summary, this paper has outlined new techniques that can be used to positively identify attacked weaknesses within web applications and enable deployment of compensating controls, like web application firewall configurations, to block additional attacks until the web application flaw can be remedied.

REFERENCES

[1] R. Lemos, “MasterCard warns of massive credit-card breach,” [Online document], 2005, [cited 2010 Mar 31], Available HTTP: http://www.securityfocus.com/news/11219

[2] T. R. Weiss, “Visa, Amex cut ties with processing firm hit by security breach,” [Online document], 2005, [cited 2010 Mar 31], Available HTTP: http://www.computerworld.com/s/article/103352/Visa_Amex_cut_ties_with_processing_firm_hit_by_security_breach

[3] I. Araujo and I. Araujo, “Developing trust in internet commerce,” Toronto, Ontario, Canada, 2003, IBM Press, pp. 1–15.

[4] J. Swaine, “Georgia: Russia ‘conducting cyber war’,” [Online document], 2008, [cited 2010 Mar 31], Available HTTP: http://www.telegraph.co.uk/news/worldnews/europe/georgia/2539157/Georgia-Russia-conducting-cyber-war.html

[5] “Cyberspace Policy Review - Assuring a Trusted and Resilient Information and Communications Infrastructure,” [Online document], 2010, [cited 2010 May 30], Available HTTP: http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf

[6] Verizon Business RISK Team, “2009 Data Breach Investigations Report,” [Online document], 2009, [cited 2010 Mar 31], Available HTTP: http://www.verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf

[7] A. Kieyzun, P. J. Guo, K. Jayarama and M. D. Ernst, “Automatic creation of SQL Injection and cross-site scripting attacks,” Washington, DC, USA, 2009, IEEE Computer Society, pp. 199–209.

[8] A. Anitha and V. Vaidehi, “Context based Application Level Intrusion Detection System,” Washington, DC, USA, 2006, IEEE Computer Society, pp. 16.

[9] L. Desmet, F. Piessens, W. Joosen and P. Verbaeten, “Bridging the gap between web application firewalls and web applications,” Alexandria, Virginia, USA New York, NY, USA, 2006, ACM, pp. 67–77.

[10] S. Nanda, L.-C. Lam and T.-c. Chiueh, “Dynamic multi-process information flow tracking for web application security,” Newport Beach, California New York, NY, USA, 2007, ACM, pp. 1–20.

[11] C. Seifert, I. Welch and P. Komisarczuk, “Application of divide-and-conquer algorithm paradigm to improve the detection speed of high interaction client honeypots SAC ‘08: Proceedings of the 2008 ACM symposium on Applied computing,” Fortaleza, Ceara, Brazil New York, NY, USA, 2008, ACM, pp. 1426–1432.

[12] B. Gehling and D. Stankard, “eCommerce security,” Kennesaw, Georgia New York, NY, USA, 2005, ACM, pp. 32–37.

[13] A. Liu et al., “SQLProb: a proxy-based architecture towards preventing SQL injection attacks SAC ‘09: Proceedings of the 2009 ACM symposium on Applied Computing,” Honolulu, Hawaii New York, NY, USA, 2009, ACM, pp. 2054–2061.

[14] K. Kemalis, and T. Tzouramanis, “SQL-IDS: a specification-based approach for SQL-injection detection SAC ‘08: Proceedings of the 2008 ACM symposium on Applied computing,” Fortaleza, Ceara, Brazil New York, NY, USA, 2008, ACM, pp. 2153–2158.

[15] W. Ren and H. Jin, “Distributed Agent-Based Real Time Network Intrusion Forensics System Architecture Design,” Washington, DC, USA, 2005, IEEE Computer Society, pp. 177–182.

[16] NetWitness, http://www.netwitness.com

[17] Niksun, http://www.niksun.com

[18] Dasient, “New Q3’09 malware data, and the Dasient Infection Library,” [Online document] 2009

Oct 27, cited [2010 Mar 31], Available HTTP: http://blog.dasient.com/2009/10/new-q309-malware-data-and-dasient.html
[19] SANS, "The Top Cyber Security Risks," [Online document], 2009, [cited 2010 Mar 31], Available HTTP: http://www.sans.org/top-cyber-security-risks
[20] B. Krebs, "Payment Processor Breach May Be Largest Ever," [Online document], 2009, [cited 2010 Mar 30], Available HTTP: http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html
[21] J. Vijayan, "TJX data breach: At 45.6M card numbers, it's the biggest ever," [Online document], 2007, [cited 2010 Mar 30], Available HTTP: http://www.computerworld.com/s/article/9014782/TJX_data_breach_At_45.6M_card_numbers_it_s_the_biggest_ever
[22] Y.-W. Huang, S.-K. Huang, T.-P. Lin, C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring," Budapest, Hungary New York, NY, USA, 2003, ACM, pp. 148–159.
[23] www.metasploit.com, milw0rm.com, www.packetstorm.nl, www.securityforest.com, www.securiteam.com, exploits.offensive-security.com
[24] Open Web Application Security Project (OWASP), http://www.owasp.org
[25] T. R. Weiss, "Visa, Amex cut ties with processing firm hit by security breach," [Online document], 2005 Jul 20, [cited 2010 Mar 31], Available HTTP: http://www.computerworld.com/s/article/103352/Visa_Amex_cut_ties_with_processing_firm_hit_by_security_breach
[26] Internet Security Systems, "Apache mod_rewrite off-by-one buffer overflow," [Online document], 2006 Jul 28, [cited 2010 Mar 31], Available HTTP: http://xforce.iss.net/xforce/xfdb/28063
[27] K. Bhargavan, C. Fournet, A. Gordon, G. O'Shea, "An advisor for web services security policies" Fairfax, VA, USA New York, NY, USA, 2005, ACM, pp. 1–9.
[28] G. Beauchemin and G. Dansereau, "Harmonized Threat and Risk Assessment Methodology," [Online document], 2007 Oct 23, [cited 2010 Mar 31], Available HTTP: http://www.rcmp-grc.gc.ca/ts-st/pubs/tra-emr/tra-emr-1-eng.pdf
[29] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," Charleston, South Carolina, USA New York, NY, USA, 2006, ACM, pp. 372–382.
[30] K. Scarfone, T. Grance and K. Masone, "NIST Special Publication 800-61 Computer Security Incident Handling Guide," [Online document], 2008 Mar, [cited 2010 Apr 11], Available HTTP: http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf
[31] K. Kent, S. Chevalier, T. Grance and H. Dang, "NIST Special Publication 800-86 - Guide to Integrating Forensic Techniques into Incident Response," [Online document], 2006 Aug, [cited 2010 Apr 11], Available HTTP: http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf
[32] L. Chen, Z. Li, C. Gao and Y. Liu, "Modeling and Analyzing Dynamic Forensics System Based on Intrusion Tolerance," Washington, DC, USA, 2009, IEEE Computer Society, pp. 230–235.
[33] K. Kent and M. Souppaya, "NIST Special Publication 800-92 Guide to Computer Security Log Management," [Online document], 2006 Sept, [cited 2010 Apr 11], Available HTTP: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf
[34] S. Peisert, M. Bishop and K. Marzullo, "Computer forensics in forensis," SIGOPS Oper. Syst. Rev., 42, 2008, ACM, pp. 112–122.
[35] R. G. Golden III and V. Roussev, "Next-generation digital forensics," Commun. ACM, 49, 2006, ACM, pp. 76–80.
[36] Y. Zhang and V. Paxson, "Detecting stepping stones." [Online document], 2000 Oct 18, [cited 2010 Mar 18], Available HTTP: http://www.icir.org/vern/papers/stepping

[37] G. Maier et al., “Enriching network security analysis with time travel,” Seattle, WA, USA New York, NY, USA, 2008, ACM, pp. 183–194.
[38] C. Eoghan, “Network traffic as a source of evidence: tool strengths, weaknesses, and future needs,” Digital Investigation, 1, 2004, pp. 28 - 43.
[39] D. K. Nilsson and U. E. Larson, “Conducting forensic investigations of cyber attacks on automobile in-vehicle networks,” Adelaide, Australia ICST, Brussels, Belgium, Belgium, 2008, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–6.
[40] A. Shabtai, D. Klimov, Y. Shahar and Y. Elovici, “An intelligent, interactive tool for exploration and visualization of time-oriented security data,” Alexandria, Virginia, USA New York, NY, USA, 2006, ACM, pp. 15–22.
[41] B. W. P. Hoelz, C. G. Ralha and R. Geeverghese, “Artificial intelligence applied to computer forensics,” Honolulu, Hawaii New York, NY, USA, 2009, ACM, pp. 883–888.
[42] S. Kornexl et al., “Building a time machine for efficient recording and retrieval of high-volume network traffic,” Berkeley, CA Berkeley, CA, USA, 2005, USENIX Association, pp. 23–23.
[43] M. Ponec et al., “Highly efficient techniques for network forensics,” Alexandria, Virginia, USA New York, NY, USA, 2007, ACM, pp. 150–160.
[44] E. Denham, “Privacy: The Intelligence Solution. Why Protecting Personal Information also Strengthens Security.”, [Online document], 2010 May 13, [cited 2010 May 13], Available HTTP: http://www.securitycanadaexpo.com/en//files/Denham_SecurityConferenceFINAL.pdf
[45] B. Otupal, “Global Police Response to the Global Cyber Threat”, Calgary, AB, 2010 May 13, Tri-Lateral Security Conference.
[46] V. Paxson, “Bro: A System for Detecting Network Intruders in Real-Time Computer Networks,” 1999, pp. 2435–2463.
[47] P. Ning and D. Xu, “Learning attack strategies from intrusion alerts,” Washington D.C., USA New York, NY, USA, 2003, ACM, pp. 200–209.

APPENDIX A – CONFERENCE SUBMISSION

The following paper is a condensed summary of this Master's integration essay and has been submitted to the Web and Pervasive Security (WPS) workshop track at IEEE GLOBECOM 2010 (Miami, Florida, December 2010) for publication consideration.

Effective SQL Injection Attack Reconstruction Using Network Recording

Allen Pomeroy Dr. Qing Tan

a@pomeroy.us qingt@athabascau.ca

Athabasca University, School of Computing and Information Systems

1 University Drive

Athabasca, Alberta, Canada T9S 3A3

Abstract – Web applications offer business and convenience services that society has become dependent on. These services have serious weaknesses that can be exploited by attackers. Success of these applications is dependent on end user trust, therefore application owners must take additional steps to ensure the security of customer data and integrity of the applications. Web applications are under siege from cyber criminals seeking to steal confidential information and disable or damage the services offered by these applications. Successful attacks have lead to some organizations experiencing financial difficulties or even being forced out of business. Organizations have insufficient tools to detect and respond to attacks on web applications, since traditional security logs have gaps that make attack reconstruction nearly impossible. This paper explores network recording challenges, benefits and possible future use. A network recording solution is proposed to detect and capture SQL injection attacks, resulting in the ability to successfully reconstruct SQL injection attacks.

Keywords: SQL injection attacks, network recording, intrusion detection, digital evidence, Bro-IDS, time machine

1. Introduction

Pervasive availability and convenience of services delivered through web application technologies has made society dependent on these services as business-to-business and business-to-consumer functions have made it easier than ever to communicate, research, learn, organize ourselves and conduct business. The dependence on web applications also increases our exposure to the effects of attacks that may directly impact individuals through compromise of confidential information, resulting in identity theft or denial of services. This may further compound into lost revenue and possibly liabilities, as illustrated by attacks forcing companies out of business [1,2]. Continued success of these applications is completely dependent on trust placed in the applications by the users and this demands increased levels of security to protect these services from attackers seeking to obtain financial and identity information or disrupt services [3].

In August 2008, hostilities between Russia and Georgia resulted in devastating denial of service attacks that shut down Georgia news agencies and government functions [4]. A cyber-security policy report prepared for US President Obama indicates over $1 trillion dollars of key intellectual property was stolen through cyber security breaches in 2008 alone [5]. Over 640,000 sites were successfully attacked in the third quarter of 2009, with approximately 5.8 million successful attacks throughout 2009 [6]. Victim sites included recognizable non-profit, government and business organizations [7,8].

The Verizon Business Risk Team determined there were 285 million records stolen in 2008 through only 90 confirmed breaches. Of those breaches, SQL injection attacks were the among the top flaws exploited. Exacerbating the problem is the ineffectiveness of traditional network security measures such as firewalls that block all traffic other than HTTP or HTTPS ports, since the majority of attacks are now targeted at the application layer [9-11]. Web application security flaws, including SQL injection attacks, are the primary exploit vector for attackers and have been shown by the NIST National Vulnerability Database to still be increasing in volume. [9] states up to 70% of the web applications responsible for handling much of the business conducted over the Internet today are affected by insecure software.

Attackers are using powerful exploit frameworks[1] to successfully attack web applications with point and click ease. Comprehensive new forensic tools are required that can definitively reveal how web applications have been attacked and what related activities were committed by the attacker.

Investigation of these application attacks are often hindered by incomplete forensic information. Sparse forensic information makes it difficult or even impossible to identify the exploited flaw or the perpetrators. Organizations are seeking ways to improve the reconstruction of web attacks in order to understand and remedy web application vulnerabilities as well as support criminal or civil legal proceedings.

1 Metasploit www.metasploit.com, CoreImpact www.coresecurity.com, sqlninja sqlninja.sourceforge.net, Immunity Canvas www.immunityinc.com and Web Application Audit and Attack Framework w3af.sourceforge.net

This paper shows the value of employing network forensic techniques and technologies to improve investigation of SQL injection attacks on web applications. Use of network based Intrusion Detection System (IDS) tools to trigger network recording of suspected application attacks can substantially improve the effectiveness of reconstructing SQL injection attacks. The remainder of this paper offers discussion and analysis of IDS triggered network recording and the use of recordings to reconstruct a simple SQL injection attack.

1.1 Related Work

We are unaware of existing projects or papers that parallel the objective of this paper. There are several related topics that we draw on heavily to produce the conclusions and in fact our body of work could not exist without these prior projects, including the Bro-IDS[2] and Time Machine[3] (TM) network recording project. There are commercial network recording products including NetWitness[4] and Niksun NetDetector[5], however they do not appear to offer tight integration with IDS. Although current literature focuses on web application security, investigation of security breaches and network forensics, this paper highlights the effectiveness of network recording to determine the exact attack vector used and the extent of the intrusion. Based on these facts, we believe the ideas expressed in this paper is new work.

1.2 Web Application Vulnerabilities and Exploitation

Web application attacks occur via several different types of vulnerabilities where Injection and Cross Site Scripting (XSS) flaws are the most serious [10,12,13]. From 2002 to 2008, the NVD showed a 1,561% increase in the number of known SQL injection flaws, compared to buffer overflow flaws, which showed a 563% increase from 1997 to 2008, revealing a dangerous trend where SQL injection flaws have overtaken memory corruption flaws. Verizon observed that SQL injection attacks are the source of the majority of records stolen (79%) over approximately 600 incident response cases. Several attacker sites include exploits for popular web applications, and a survey of several attack tool websites found 47% of the exploits were oriented to performing SQL injection attacks.

1.3 SQL Injection Attacks

Web application servers send low level SQL queries to a database in order to build the HTML needed for user presentation, typically through application API calls such as JDBC. Problems arise when programmers do not handle user input properly, resulting in unintended direct user interaction with the database. Since low level string manipulation is used to dynamically build database SQL queries based on user input, if that input is not handled correctly, unintended modification of the SQL queries can happen [14]. Results include unintended query functions [13,14]. For example [14]:

```
query = "SELECT * FROM accounts WHERE name='"
        + request.getParameter("name")
        + "' AND password='"
        + request.getParameter("pass") + "'";
```

This code is intended to authenticate a user, however can be subverted when an attacker enters `' OR ' a'='a'` into the password field. This specially crafted input generates the following query with the result of the conditional logic always returning true, allowing the attacker to bypass remaining authentication logic.

```
SELECT * FROM accounts WHERE name='admin' AND password='' OR 'a'='a'
```

Once an attacker confirms an application is vulnerable to SQL injection attacks, they can progress to enumerating the database schema and alteration or insert of rows, leading to horizontal and/or vertical privilege escalation.

2 Attack Analysis

Traditional digital forensics [15] relies on log meta data relating to web transactions, including network and application firewalls, IDS, operating system and web server logs, and application logs [16,17]. These logs are often insufficient for forensic purposes since they were only designed for system debugging purposes [18]. Where network

2 www.bro-ids.org
3 tracker.icir.org/time-machine
4 www.netwitness.com
5 www.niksun.com

recording capabilities do not exist, the investigator is forced to reconstruct events from incomplete data [19,20], often resulting in poor quality since they are left to deduce system or data changes and events from sparse forensic information [18]. With lack of detailed information to illuminate the precise actions of the attacker, determination of exactly what the attacker did and on which systems is difficult, if not impossible, therefore the increased level of data provided by network recordings are needed [21].

```
1: Jun 13 17:59:45 fw2 kernel: [24101.376743] RULE 11 -- ACCEPT IN=eth1 OUT=eth2
   SRC=10.10.21.12 DST=10.10.22.12 DF PROTO=TCP SPT=33682 DPT=80 SEQ=2502871064
   ACK=0 WINDOW=5840 RES=0x00 SYN URGP=0

2: 10.10.21.12 - - [13/Jun/2010:17:59:45 -0600] "POST /mutillidae/index.php?
   page=login.php HTTP/1.1" 200 5988 "http://web11/mutillidae/index.php?
   page=login.php" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15)
   Gecko/2009102814 Ubuntu/8.10 ...
```

Listing 10: Web server log meta-data and HTTP POST data

Listing 10 shows log data for a successful web application access, where the investigator can see `attacker (10.10.21.12)` was permitted access to the web server. Nothing further is known from the firewall logs other than which hosts and services node `attacker` specifically tried to access. Even with the first entry showing a firewall accept of a TCP connection to node `web11`, there is no indication that a web server was running on that server, if the TCP connection or HTTP transaction was successful. The web server log entry shows the web server successfully served a HTTP POST request. There is no way to deduce this was a malicious request through log data analysis alone.

```
1: 10.10.21.12 - - [28/Apr/2010:13:17:14 -0600] "POST /mutillidae/index.php?
   page=login.php HTTP/1.1" 200 2237 "http://web11/mutillidae/?page=login.php"
   "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15) Gecko/2009102814
   Ubuntu/8.10 (intrepid) ...

2: POST /mutillidae/index.php?page=login.php HTTP/1.1
   Host: web11
   User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0 ...
   Accept: text/html,application/xhtml+xml,application/xml ...
   Referer: http://web11/mutillidae/?page=login.php
   Cookie: showhints=1
   Content-Type: application/x-www-form-urlencoded
   Content-Length: 68
   user_name=admin&password=%22%27+or+1%3D1+--+%22&Submit_button=Submit
```

Listing 11: Web server log meta-data and HTTP POST data

Listing 11 further illustrates the dangerous gap between transaction meta data recorded by the web server followed by a HTTP network conversation payload capture, where the network recording shows the password variable containing a SQL injection attack `' or 1=1 --` that was not revealed in the web server transaction log.

2.1 Network Traffic Recording

[15] discuss methods to collect and analyze evidence of attacks that could be used to reveal successful and unsuccessful attacks on web applications. [22] take this approach further, by incorporating IDS controlled application payload recording, preserving suspicious activity for retrospective inclusion in subsequent investigations.

In addition to enabling reconstruction of SQL injection attacks, network recordings are a valuable source of compelling digital evidence since it can be used to show the actions of an offender like a surveillance video tape of a crime [15,18,23]. Using network traffic as digital evidence does have numerous drawbacks including possible errors in reconstruction of the traffic payload [23] and sheer volume of unconstrained network recording, in terms of network traffic, required processing resources and disk space [24,25].

To address this limitation, our solution uses the Bro-IDS project for the built-in interfaces and ability to leverage and control the recording resources of a TM [22]. We use this integration between Bro-IDS and TM to dynamically change recording parameters set within TM. The policy language developed by the Bro-IDS project is geared to the specialized task of analyzing and monitoring network traffic, and includes event level linkages to a TM, enabling retrospective analysis of

suspicious traffic by the IDS [22,25,26] and security analysts.

The bulk of traffic in high volume streams typically occurs with a relatively small number of connections [22,25]. Most network connections are short, with a small number of large connections making up the majority of the network traffic [25]. The TM project implements an algorithm to capture sufficient detail at the beginning of network streams while discarding the heavy tail [25] or bulk of the transaction data, referred to as a cut-off point. This approach results in substantial data recording reduction while maintaining enough of the transaction for identification and to illustrate an attacker's actions. Since for forensic purposes [22,25,27], the packet headers and a relatively small amount of the payload from the beginning can accurately fingerprint connection patterns [27], the overall volume of traffic payload recorded can be reduced by as much as 96% [25].

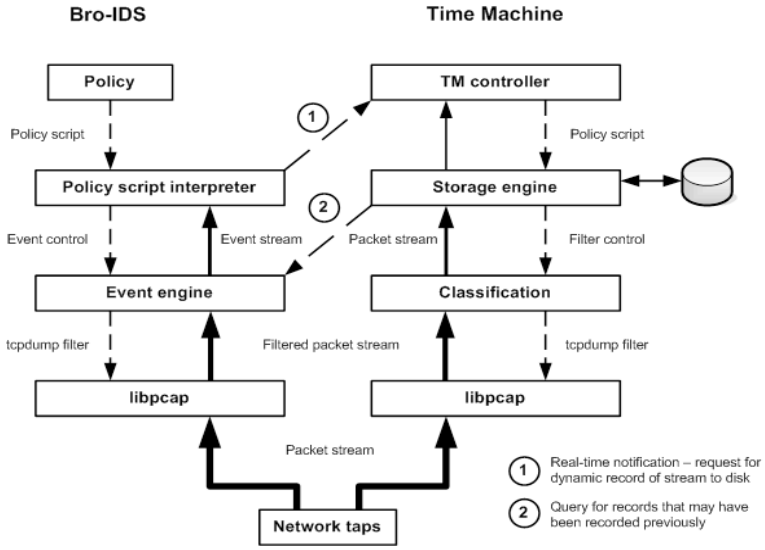

Figure 14: Bro-IDS and TimeMachine structure

Our solution records network traces of both static traffic patterns predetermined to be of interest to security administrators, as well as dynamically captured traces determined to be of interest by the Bro-IDS system. Figure 14 illustrates the overall solution configuration, with network traffic feeding both Bro-IDS and the TM. The IDS was configured to trigger recording of all HTTP sessions where specific SQL injection attack patterns were seen. Effectively the IDS policy statements translate into *tcpdump* filters for both the IDS and TM. The *tcpdump* filters act as the first level of traffic reduction, where the IDS event engine analyzes attributes within the incoming packets for a policy match. Matches trigger the TM to begin recording the selected TCP/IP or UDP/IP stream.

2.2 Reconstruction of SQL injection attack

A vulnerability assessment of the victim application[6] revealed manipulation of the *password* parameter used in a POST operation was possible (Listing 12).

```
SQL injection in a MySQL database was found at:
"http://web11/mutillidae/index.php?page=user-info.php", using HTTP method POST.
The sent post-data was:
"view_user_name=LzGulbg&password=d'z"0&Submit_button=Submit". The modified
parameter was "password".
```

Listing 12: HTML POST vulnerability in Mutillidae

Results of the test confirmed the password parameter has been successfully modified including characters that could be used in a SQL injection attack. TM queries were used to restore network *pcap* data showing the possible attack.

```
./tm-query --ip 10.10.22.12 localhost web11-20100609-1415.pcap --time 5m
```

WireShark was used to reconstruct the HTTP POST payload (Figure 15). Highlighted sections in the traffic

6 http://www.irongeek.com/i.php?page=security/mutillidae-deliberately-vulnerable-php-owasp-top-10

reconstruction show the malicious attack; section 1 shows the HTTP POST query; section 2 shows the attacker (src) and victim server (dst); section 3 shows the TCP protocol used (HTTP); section 4 shows the HTTP POST header; and section 5 shows the actual SQL injection attack in the *password* parameter. Analysis of subsequent HTTP/1.1 200 return HTML indicates the attacker succeeded in bypassing authentication and becoming Administrator in the application.

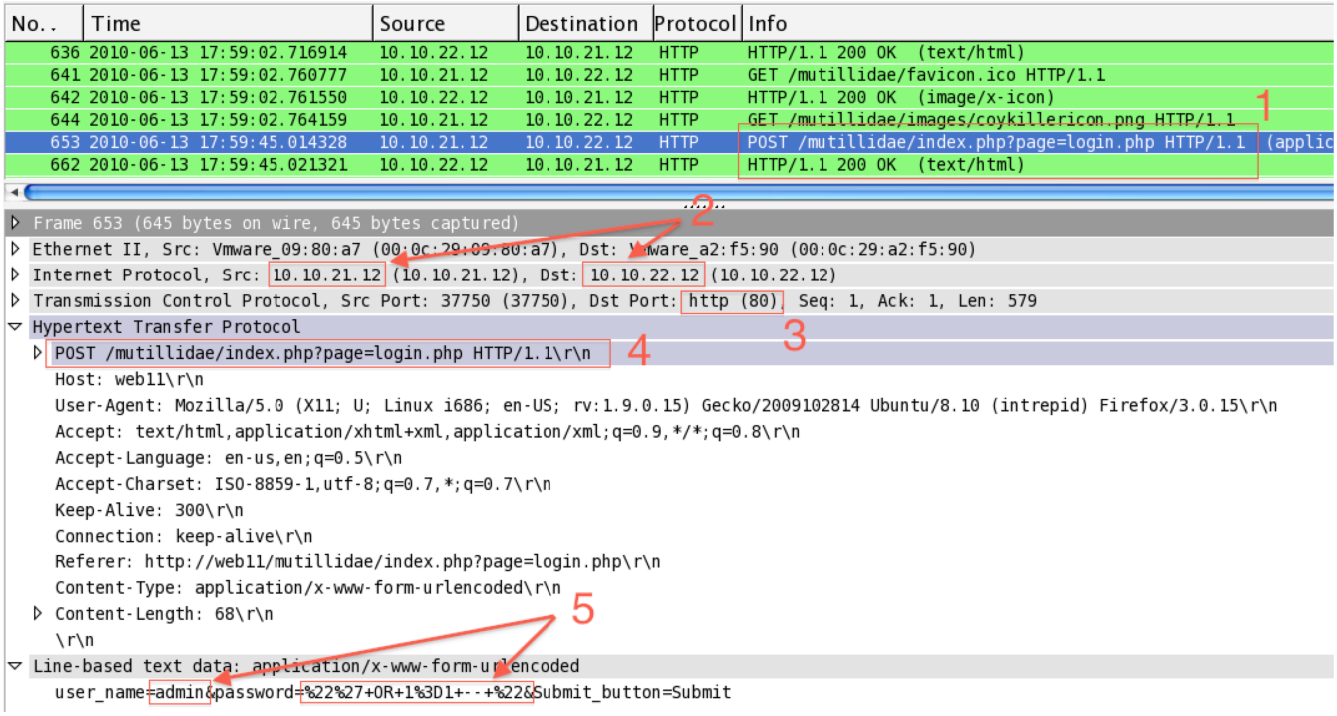

Figure 15: Wireshark analysis of Time Machine pcap data

As shown by the evidence captured by Time Machine, in the actual SQL injection attack, the vulnerable password parameter was assigned a value as `password=%22%27+OR+1%3D1+--+%22` , resulting in the SQL injection attack `' OR 1=1 --` modifying the SQL query to `SELECT * FROM users WHERE username='admin' AND password='' OR 1=1 --`. This allows the attacker to successfully bypass authentication for the administrative user.

3 Conclusion

A majority of web applications used for handling business over the Internet have serious security flaws, enabling successful attacks. Organizations have insufficient tools to detect and respond to attacks on web applications, since traditional security logs have gaps that make attack reconstruction nearly impossible. Traditionally, precise actions of the attackers is difficult to determine, therefore the increased level of data provided by network recordings are needed

Leveraging IDS controlled application payload recording allows for preservation of suspicious activity for retrospective inclusion in subsequent investigations. These recordings are a valuable source of compelling digital evidence since it can be used to show the actions of an offender like a surveillance video tape of a crime. Although high traffic volumes are difficult to indiscriminately record, use of heavy-tail and focused recording techniques enable effective SQL injection attack reconstruction that would have previously gone undetected.

References

[1] R. Lemos, “MasterCard warns of massive credit-card breach,” [Online document], 2005, [cited 2010 Mar 31], Available HTTP: http://www.securityfocus.com/news/11219

[2] T. R. Weiss, “Visa, Amex cut ties with processing firm hit by security breach,” [Online document], 2005, [cited 2010 Mar 31], Available HTTP: http://www.computerworld.com/s/article/103352/Visa_Amex_cut_ties_with_processing_firm_hit_by_security_breach

[3] I. Araujo and I. Araujo, “Developing trust in internet commerce,” Toronto, Ontario, Canada, 2003, IBM Press, pp. 1–15.

[4] J. Swaine, “Georgia: Russia ‘conducting cyber war’,” [Online document], 2008, [cited 2010 Mar 31], Available HTTP: http://www.telegraph.co.uk/news/worldnews/europe/georgia/2539157/Georgia-Russia-conducting-cyber-war.html
[5] “Cyberspace Policy Review - Assuring a Trusted and Resilient Information and Communications Infrastructure,” [Online document], 2010, [cited 2010 May 30], Available HTTP: http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf
[6] Verizon Business RISK Team, “2009 Data Breach Investigations Report,” [Online document], 2009, [cited 2010 Mar 31], Available HTTP: http://www.verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf
[7] B. Krebs, “Payment Processor Breach May Be Largest Ever,” [Online document], 2009, [cited 2010 Mar 30], Available HTTP: http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html
[8] J. Vijayan, “TJX data breach: At 45.6M card numbers, it’s the biggest ever,” [Online document], 2007, [cited 2010 Mar 30], Available HTTP: http://www.computerworld.com/s/article/9014782/TJX_data_breach_At_45.6M_card_numbers_it_s_the_biggest_eve
[9] A. Kieyzun et al., “Automatic creation of SQL Injection and cross-site scripting attacks,” Washington, DC, USA, 2009, IEEE Computer Society, pp. 199–209.
[10] B. Gehling and D. Stankard, “eCommerce security,” Kennesaw, Georgia New York, NY, USA, 2005, ACM, pp. 32–37.
[11] L. Desmet et al., “Bridging the gap between web application firewalls and web applications,” Alexandria, Virginia, USA New York, NY, USA, 2006, ACM, pp. 67–77.
[12] SANS, “The Top Cyber Security Risks,” [Online document], 2009, [cited 2010 Mar 31], Available HTTP: http://www.sans.org/top-cyber-security-risks
[13] S. Nanda, L.-C. Lam and T.-c. Chiueh, “Dynamic multi-process information flow tracking for web application security,” Newport Beach, California New York, NY, USA, 2007, ACM, pp. 1–20.
[14] Z. Su and G. Wassermann, “The essence of command injection attacks in web applications,” Charleston, South Carolina, USA New York, NY, USA, 2006, ACM, pp. 372–382.
[15] L. Chen et al., “Modeling and Analyzing Dynamic Forensics System Based on Intrusion Tolerance,” Washington, DC, USA, 2009, IEEE Computer Society, pp. 230–235.
[16] K. Kent et al., “NIST Special Publication 800-86 - Guide to Integrating Forensic Techniques into Incident Response,” [Online document], 2006, [cited 2010 Apr 11], Available HTTP: http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf
[17] K. Scarfone, T. Grance and K. Masone, “NIST Special Publication 800-61 Computer Security Incident Handling Guide,” [Online document], 2008, [cited 2010 Apr 11], Available HTTP: http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf
[18] S. Peisert, M. Bishop and K. Marzullo, “Computer forensics in forensis,” *SIGOPS Oper. Syst. Rev.*, 42, 2008, ACM, pp. 112–122.
[19] R. G. Golden III and V. Roussev, “Next-generation digital forensics,” *Commun. ACM*, 49, 2006, ACM, pp. 76–80.
[20] W. Ren and H. Jin, “Distributed Agent-Based Real Time Network Intrusion Forensics System Architecture Design,” Washington, DC, USA, 2005, IEEE Computer Society, pp. 177–182.
[21] D. K. Nilsson and U. E. Larson, “Conducting forensic investigations of cyber attacks on automobile in-vehicle networks,” Adelaide, Australia ICST, Brussels, Belgium, Belgium, 2008, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–6.
[22] G. Maier et al., “Enriching network security analysis with time travel,” Seattle, WA, USA New York, NY, USA, 2008, ACM, pp. 183–194.
[23] C. Eoghan, “Network traffic as a source of evidence: tool strengths, weaknesses, and future needs,” *Digital Investigation*, 1, 2004, pp. 28 - 43.
[24] B. W. P. Hoelz, C. G. Ralha and R. Geeverghese, “Artificial intelligence applied to computer forensics,” Honolulu, Hawaii New York, NY, USA, 2009, ACM, pp. 883–888.
[25] S. Kornexl et al., “Building a time machine for efficient recording and retrieval of high-volume network traffic,” Berkeley, CA Berkeley, CA, USA, 2005, USENIX Association, pp. 23–23.
[26] V. Paxson, “Bro: A System for Detecting Network Intruders in Real-Time Computer Networks,” 1999, pp. 2435–2463.
[27] M. Ponec et al., “Highly efficient techniques for network forensics,” Alexandria, Virginia, USA New York, NY, USA, 2007, ACM, pp. 150–160.