Vulnerability Document

30000737

SQL Injection

SQL injection is when an attacker is able to influence a Structured Query Language (SQL) queries that an application passes to a back-end database. If this vulnerability were to occur, the results could be devastating with the potential of usernames, passwords, credit card information, among other things, could be exposed. According to (Clarke-Salt, 2009) “By being able to influence what is being passed to the database, the attacker can leverage the syntax and capabilities of SQL itself” showing how much power the attacker is able to have over a database using this method.

SQL injection not only affects web applications, but rather any code that can accept input from an untrusted source which then uses that input to form dynamic SQL statements, making this method extremely dangerous. SQL injection works by an SQL code being inserted or appended into user input parameters which are then later passed to a back-end SQL server for parsing and execution. It is a main threat for military and banking systems, where some of the most private information is held.

One of the best ways to detect and prevent SQL injection is to use a combination of static and dynamic analysis, however this method is very complicated to implement. The work of Lee *et al* (2012) explains what each of these types of analysis are and how to implement them. When describing static analysis, they state that “the focus of the static analysis method is to validate the user input type in order to reduce the chances of SQL injection attacks rather than detect them”. This is useful when combined with other methods to help detect SQL injection, however they go on to state that static analysis cannot protect against SQL injection if “malicious data input has the correct syntax or type”. They then go onto describe how dynamic analysis works, stating that “dynamic analysis analyses the response from a web application after scanning it”. The scan will send every type of input to the server and receive the response. Comparing this to static analysis, it is more efficient as it can “locate the vulnerabilities from SQL injection attacks without having to make any modifications to the web application”. This is a benefit as it would give more realistic results and would be able to assist in helping eliminate the vulnerability, therefore strengthening the web application.

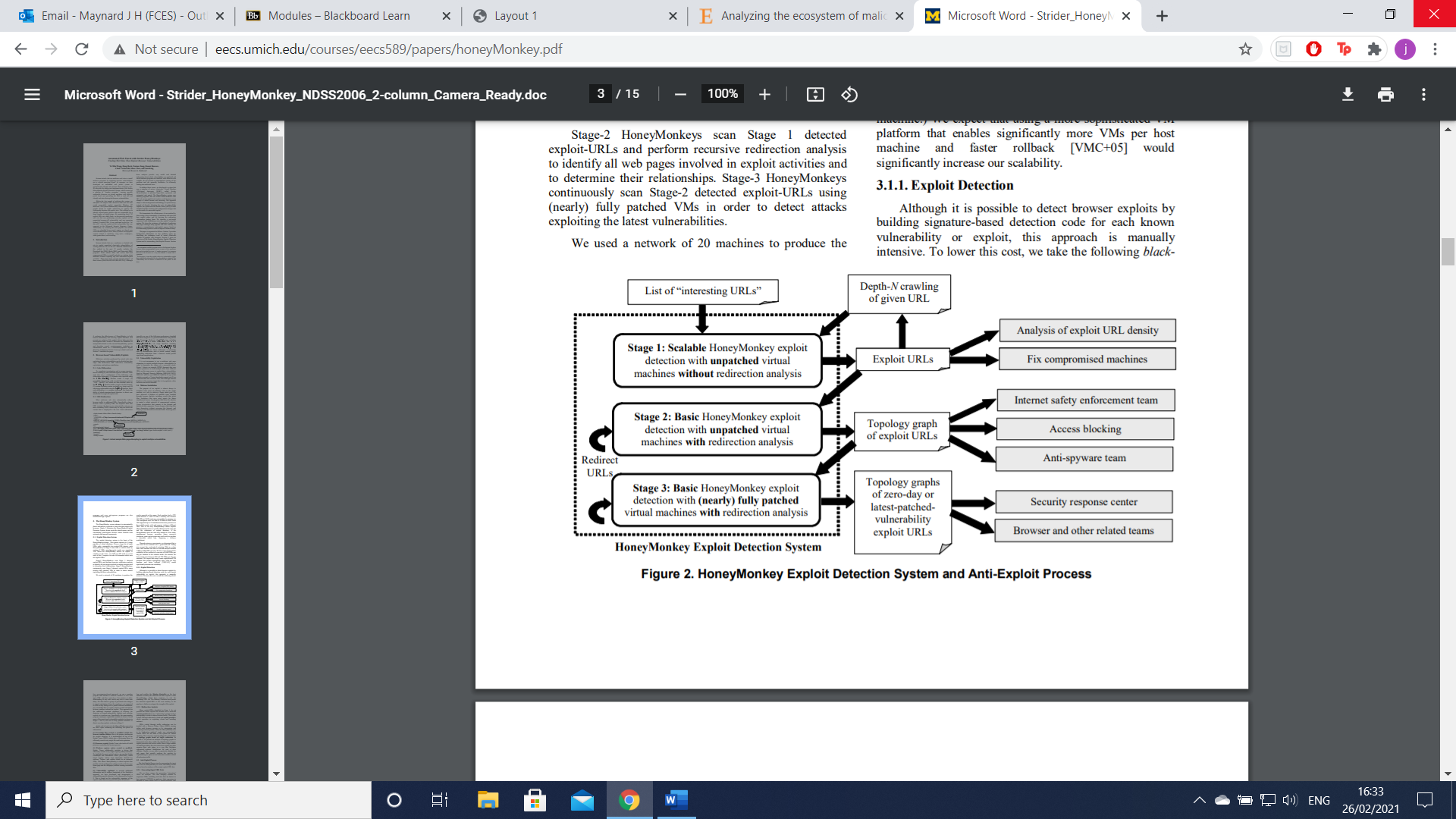
Combining both of the methods would allow for the advantages of the methods to be utilized to detect any SQL injection attacks. They go on to explain how by saying “it analyses web pages and simultaneously generates SQL queries to test the results”. This is one of the most effective methods of detecting and preventing SQL injection attacks available.

URL Redirection

One way that is used to exploit a vulnerability is with the use of URL redirection. This method involves using an already compromised website to act as a stepping-stone to another website which will automatically download and install malware onto the users’ device.

According to Akiyama *et al* (2017) the definition of redirection is “automatically replacing access destinations, and it is generally controlled over an HTTP protocol on the web”. Once the redirection has occurred the web page will download and install malware such as a Trojan or other serious viruses. Internet Explorer is the main source of attack due to its popularity; however other browsers can be exploited.

One method of detecting and preventing a URL redirection is with the use of the HoneyMonkey system. Wang *et al* (2006) describes this system as “an attempt to automatically detect and analyse a network of websites that exploit web browsers”. The figure below illustrates the HoneyMonkey system and the surrounding anti-exploit process.



It is possible to detect browser exploits by building detection code for each vulnerability, however this is very intense and time consuming. They go on to discuss a simpler method of doing this, known as a *black box, non-signature based approach*. They explain it as “running a monkey program that launches a browser instance to visit each URL input and waits for a few minutes to allow downloading of any code”. The idea of this method is once the system has visited a website, it will produce a report detailing the following information:

* Executable files created or modified outside the browser sandbox folder.
* Processes created.
* Windows registry entries created or modified.
* Vulnerability exploited.
* Redirect-URLs visited.

With the above information, the system is then able to analyse the sites visited and can then run the anti-exploit process which will help deal with the redirected web pages.

Software that is already infected with virus.

According to Wikipedia, a computer virus is a type of computer program that, when executed, replicates itself by modifying other computer programs and inserting its own code. If this replication succeeds, the affected areas are then said to be "infected" with a computer virus. One of the most common software security vulnerabilities is the software that is already infected with a virus.

Software /Computer viruses are programs that must be triggered or somehow executed before they can infect your computer system and spread to others. Examples include opening a document infected with a “macro virus,” booting with a diskette infected with a “boot sector” virus or double-clicking on an infected program file. Viruses can then be spread by sharing infected files on a diskette, network drive, or other media, by exchanging infected files over the Internet via e-mail attachments, or by downloading questionable files from the Internet.

Software Virus can seriously affect our privacy and system’s security. For example, Virus /malware is known for relaying personal information to advertisers and other third parties without user consent. Some programs are also known for containing worms and viruses that cause a great deal of system damage. The terms “virus” and “malware” are often used interchangeably, but they are not the same thing. While a computer virus is a type of malware, not all malwares are computer viruses. In effort to protect software and systems, we can focus on taking effective measures that protect us from computer viruses. First and most essential protection is to install an antivirus software program. Second, update software / operating systems occasionally. Next, check for antivirus programs every period of time. Fourth, you should avoid websites that are not secure. Finally, visit only those websites for downloading that are reliable and worthy. In addition, for email, you can avoid messages from unknown senders and delete spam messages from your inbox. Thus, by following these measures we can avoid the virus next time completely. A security Anti-Virus Software program should be installed onto our computer, which helps protect our computer from viruses and also can scan, detect, and remove the viruses. It is supported by firewall software (which protects against hackers and some viruses) and can do several functions to protect our computer from viruses. The advantages of antivirus software are obvious as its name states. They not only protect our computer and our data, but also save our expenses and make us feel a lot safer. Continuing the virus analogy, if a given population stops receiving vaccinations for diseases thought to be eradicated, like the measles and polio, those diseases can and do come back. Likewise, it is important to be proactive about cybersecurity and take some basic protective measures against computer viruses. Otherwise, computer viruses could make a comeback.

The use of the internet cannot be stopped, and we can also not determine if the information downloaded contained any viruses, it is advisable to use credible anti-virus software with latest updates. It warns us about any threats that a file or any information downloaded from the internet or a copied over infected file contains.

To conclude, software that is already infected with virus can cause a great harm to not only individual but the whole system.

Weak Password Vulnerability

As we are living in an information-driven world today that requires us to protect what and how we share the information. To protect and access the data or information we require either a password or a pin.

Passwords are the most common form of authentication used to control access to information, which is one of the most common software security vulnerabilities. In fact, weak passwords are the root cause of security threats.

According to **Wikipedia**, a **password**, sometimes called a passcode, is a memorized secret, typically a string of characters, usually used to confirm a user's identity.

In general, a password is an arbitrary string of characters including letters, digits, or other symbols. If the permissible characters are constrained to be numeric, the corresponding secret is sometimes called a Personal Identification Number (PIN).

In today’s generation, Passwords are used for access to personal phones, computers, online portals, websites and computer networks and databases for many software applications.  A weak password is short, common, a system default, or something that could be rapidly guessed by executing a brute force attack using a subset of all possible passwords, such as words in the dictionary, proper names, words based on the username or common variations on these themes.

However, ensuring that passwords are encrypted, and safe have become one of the greatest challenges for all individuals. With today's interconnected Internet, the problems are potentially devastating on an even larger scale; a skilful intruder may break into one system and never harm it, using it instead as a platform for attacks on a population of millions of targets. People should follow guidelines when creating a strong password.

Generally, a password should include alpha, numeric, and punctuation characters, and longer than 6 characters in length. Instead, we can use passphrases, applications that usually use shorter passwords reject real names and words, which are disreputable and unsafe. Rather, a password should often be an unsystematic sequence of characters. It is advisable to have a greater length of passphrase, which is easy to memorize, instead of a cryptic sequence of symbols, numbers, or letters that are short. Therefore, passphrases are more secure than cryptic passwords since they are longer and secure against password cracking.

One of the concerns that people often have when it comes to creating complex passwords is a fear of forgetting them, particularly when there are several to remember. Naturally, a person should try to think of something that will be easy for them to memorize. One way to do that is to turn a sentence or phrase into something that is not easily recognized by others. To do this, use the first letter of every word in the sentence, replacing certain words with numbers or symbols. For example, the word "for" may be replaced with the number 4 or the word "number" with the # symbol. With this method, a password such as "Save the number for later in the year" may read St#4LITY.

Buffer Overflow

In the recent years, buffer overflow vulnerability presents a high incidence, which is not optimistic. According to the latest release of CWE [5], the first most dangerous types of errors in 2019 can be seen, the type of improper operation restrictions within the memory buffer range (cwe – 119) ranks the first.

In the figure below is the statical analysis on the developmental trend of the number of security vulnerability and the proportion of buffer overflow vulnerabilities CVE [6] over the years from 1999 to 2018. It can be seen that in the past 20 years, the number of buffer vulnerabilities has increased year by year, and the number of vulnerabilities has increased sharply on the recent years. At the same time, the proportion of buffer overflow is growing rapidly.

**ESAET 2020 IOP Publishing ­­­­­­­Journal of physics: conference site 1549(2020) 022064 doi :** 10.1088/17426596/1549/2/022064

Chart

Description automatically generated

**Figure 1.** The number of security holes over the years and the proportion of buffer overflow vulnerability.

A major cause of the buffer overflow problem is that some of the string-manipulation functions and printing functions in the C library, such as strcpy(), streat(), sprint(), and gets (), can easily cause buffer overflow vulnerabilities if the boundary checking code is not included when the program calls them.

Static analysis method was adopted to detect buffer overflow caused by unsafe function innovation.

**Unsafe function innovation feature model**

*Unsafe Function Call Feature Extraction*

The lack of security checks on buffer boundaries in programs is the main cause of buffer overflow problems. According to library functions of unsafe functions, based on existing research results and the analysis of a large number of example, the characteristics of overflow vulnerability caused by the use of such functions are listed:

* Call the data copy class unsafe function.
* Buffer copy does not check input size.
* The source buffer data is not constant.
* Copy length is not constant.

# (Zaho, 2020) (Zaho, 2020)

Cross – Site Scripting (XSS)

Cross- Site Scripting (XSS) is a known web attack. It occurs when malicious code is sent or executed, usually in script form, from the browser on the victim’s computer, using their web applications. With this execution you could filter the personal information or use the user cookies to hijack the identity in a fraudulent session, so it offers the attackers the possibility of stealing sensitive data or even being able to take control of certain devices.

According to data from Imperva the XXS attacks represents the highest number of web application vulnerabilities in 2017. In fact, their number has doubled compared to 2016. And according to Imperva’s predictions, they will follow being the most frequent offensives in 2018.

Background of Cross-Site Scripting Attacks

Web applications are insecure by default. The main reason, their developers do not establish secure, this contributes to theft of personal and crucial information from users. This lack of good practices is considered a vulnerability.

According to the latest report on security statistics in web applications, the victim is the user and not the applications. XSS ranks second as one of the most serious vulnerabilities, with approximately 38% critical. However, what is worrying is that this type of attack has a very low solution and/ or meditation rate.

In summary, XSS is evaluated according to the following parameters: exploitable, business impacts, technical impacts, weakness detectability, and weakness prevalence.

Types of XXS Attacks and Exploitation Examples

According to the literature, there are 3 types of attacks: persistent XXS, non- persistent XSS and Document Object Model (DOM) XSS. The vulnerabilities are found in the software, the hardware and also in the users (developers) that are part of any computing environment.

**Scenario for reflected XXS attack**

The simplest scenario is when a web page is required to inject the malicious code through its search engine, Figs 1 and 2 shows the result of a real test performed within the pizza.com domain, where a malicious script was injected with the aim of showing an alert in the web browser.

A picture containing application

Description automatically generated

Fig 1. Scenario for a reflected or indirect XSS attack



Fig 2. Sample XSS code reflected injected in the pizza.com domain.

(German E. Rodiguez, 2020,)

**References**

[1] Cousot P, Giacobazzi R, Ranzato F. Program Analysis Is Harder Than Verification: A Computability Perspective [C] // International Conference on Computer Aided Verification. 2018.

[2] Ji T, Wu Y, Wang C, et al. The Coming Era of AlphaHacking?: A Survey of Automatic Software Vulnerability Detection, Exploitation and Patching Techniques [C] //2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE, 2018: 53-60.

[3] Ren J, Zheng Z, Liu Q, et al. A Buffer Overflow Prediction Approach Based on Software Metrics and Machine Learning [J]. Security and Communication Networks, 2019, 2019.

[4] Shao SH, Gao Q, Ma S, Duan FY, Ma X, Zhang SK, Hu JH. Progress in research on buffer overflow vulnerability analysis technologies. Ruan Jian Xue Bao/Journal of Software, 2018, 29 (5): 1179−1198 (in Chinese). http://www.jos.org.cn/1000- 9825/5504.htm

[5] 2019 CWE Top 25 Most Dangerous Software Errors [EB/OL]. [2019-09-18]. <https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html>.

[6] CVE [EB/OL]. [2019–07–16]. <https://cve.mitre.org/cve/>.

[7] Zhang J, Zhang C, Xuan JF, Xiong YF, Wang QX, Liang B, Li L, Dou WS, Chen ZB, Chen LQ, Cai Y. Recent progress in program analysis. Ruan Jian Xue Bao/Journal of Software, 2019, 30 (1): 80-109 (in Chinese). http://www.jos.org.cn/ 1000-9825/5651.htm.

[8] Andriesse D, Chen X, Veen V V D, et al. An In-Depth Analysis of Disassembly on Full-Scale x86/x64 Binaries [C] // USENIX Security Symposium. 2016. [9] CVE-2017-11882|Microsoft Office Memory Corruption Vulnerability [EB/OL]. [2017–11–29] <https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-11882>

Lee, I., Jeong, S., Yeo, S. And Moon, J. (2012) ‘A novel method for SQL injection attack detection based on removing SQL query attribute values’ *Mathematical and computer modelling,* 55(1-2), pp. 58-68

Clarke-Salt, J. (2009)  *SQL Injection Attacks and Defense.*  Elsevier.

Akiyama, M., Yagi, T., Yada, T., Mori, T. and Kadobayashi, Y. (2017) ‘Analyzing the ecosystem of malicious URL redirection through longitudinal observation from honeypots.’ *Computers and Security,* 69, pp. 155-173.

Wang, Y., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S. and King, S. (2006) *Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities.* http://www.eecs.umich.edu/courses/eecs589/papers/honeyMonkey.pdf