The University of Southern Mississippi

Software Security

Securing Software to Prevent Threats

Kacheef K White

CSC 424 – Software Design

Dr. Michaela Elliot

March 21, 2021

Secure Software

A well-developed software involves efforts to keep codes and data secure from unauthorized users or breaches. This effort requires expertise, diligence, and many times money. The increase demand within the cybersecurity industry is proof of the importance of having good software security.

Secure Software is software created or engineered in a particular way where its operations and functionalities can operate normally even when attacked maliciously (4). The integrity of data and systems in its domain continue to be safe and the malicious attacks are found and removed. There can be many forms of malicious attacks including but not limited to viruses, worms, spywares and trojans. A computer virus modifies host files in a special way where when that file is executed, the virus becomes active and begins its destructive path. Computer viruses behave similarly to a biological virus. The first computer virus called the "Creeper system", was born in 1971. This virus was created and contained by BBN technologies in the US. In 1988, a graduate student, Robert Morris from Cornell University, created his own virus called "The Morris"(3). Despite the lack of creative thinking in the name, this virus spreads faster than the summer fires in California. The virus infected most of the computers in that time and since then, an exponential number of new viruses have been created each year.

Worms, similar to viruses, can have a severe impact to businesses. Worms have a special trait that allows them to duplicate without any action from the end user. This can be a nightmare for any company. If one person opened an email with a worm attached, all the other personnel within the company can be easily infected in a very short period of time. Computer worms can do a lot of damage by exploiting vulnerabilities in a system. They have the ability to duplicate themselves endlessly causing resource shortages. Worms also can be used to delete or modify files, causing a problem to data integrity. Trojans are a hacker go to weapon when trying to conduct an attack. They are hard to detect and remove as there are too many signatures to keep up with in the modern era.

Software developers have a huge responsibility to ensure that industry best practices are implemented when necessary. Minimizing how many bugs are in a software is essential for preventing or reducing exploitation. Any software defect could see negative results with buffer overflows and error handling. Software that uses the internet are most at risk with malicious attacks. With the increase complexity in the technology field, it becomes extremely difficult to prevent security holes in a software. A software covered with vulnerabilities can cause a business such as banks billions of dollars easily in just a short period of time.

Software Security Tools

It is not uncommon to see the news covering stories of data breach or a hacked website. With the advancement of technology and easy access to information online, it is absolutely necessary for software engineers to identify vulnerabilities as soon as possible. Thankfully, software security testing tools are available to use and no access to the source code is necessary for these tools to work. Security testing is an important phase for most software project. This ensures unauthorized users do not gain access to the system and the integrity of data remains protected. Some of the sections covered by security testing are integrity, non-repudiation, confidentiality, availability, authorization, and authentication. These testing with help organizations prevent data breach, increase stability and keeping their customers' trust.

One of the tools that is commonly used for security testing is Zed Attack Proxy (ZAP). This was created by OWASP and can be used across many platforms. This is an open-source web application security testing tool which makes it even more attractive for professionals to use (1). Zap is written in Java and can make it easy for a novice to use while providing tools like the command line for advanced users. ZAP provides great scanning capabilities and can also be used to intercept a proxy for manual testing. For brute-force testing web applications, Wfuzz is most commonly used. It has no GUI and was developed in Python (1).

Detecting database injection and testing error-handling is important to prevent catastrophic failure. If software security is not taken seriously, an end user could query the database and do unauthorized changes like dropping a table. SQLMap is a great free tool that supports "automating the process of detecting and utilizing SQL injection vulnerability in a website's database"(1). It supports databases such as PostgreSQL, Oracle, and MySQL. This tool test for 6 types of SQL injection techniques. This includes Boolean-based blind, error-based, out-of-band, stacked queries, time-based blind and last but not least UNION query.

Whether a vulnerability scanning tool is commercial or open source, the fact remains that there are many tools available for developers to use that will help create a peace of mind when their product deploys to production. Refusing to use these tools is not a wise choice as access to source code is not necessary for these scanning tools to work resulting in no big disadvantages from using them. Some of these tools are platform specific to SaaS, Windows, Linux or MacOS but some are also support all these platforms altogether. It is important to choose the right tool for the job and this will need experience and or research.

How to Make Code More Secure

When designing a software, security should always be in mind. Neglecting to put effort in securing your software can cause data breaches and long unplanned downtime. This can cause a lot of money for repair and reduced trust from customers. Knowing your framework and its common vulnerabilities is essentially for software security.

One of the most common cause of vulnerabilities in software is trusting that the end users will not try to cause harm to your application. Although most users do not have a malicious intent when using a software, there are many users that will try to break your software for their own selfish gain. These attacks usually come in the form of Cross Site Scripting, XML External Entities, Insecure Deserialization, and injection. To prevent these types of attacks, developers should validate every code before it is executed and. Validating a code is similar to a TSA agent's duty at an airport. It rejects suspicious or unsafe code and allow safe codes to be executed.

One of the hardest but most important concept to grasp about coding is that you should always try to keep it as simple as possible. The more complex a software design becomes, the harder it is to test for vulnerabilities to help keep it secure. The goal of a software design is not to show how creative and complex your code can be but to display the opposite. Always try to find the simplest way to accomplish business requirements. This helps with maintaining the software and reducing code rot.

Automating as much processes as possible is a great way to prevent dangerous queries from end users. Remove the ability of the end user to customize their request as they please

eliminate worrying about security as the software scales up. Consistency is software security best friend. Using cryptography correctly is almost a must have for modern software. Data is every company's money-making tool and not protecting this material can cost a company years and billions of dollars to recover. If an end user finds a way to breach security, cryptography becomes the last line of defense to secure data. Cryptography when done wrong can become useless and developers should be aware of how to use this tool correctly. A good attacker will always try to find as much vulnerabilities as possible and best practices are essentially to maintaining trust and data integrity.

References

1. https://hackr.io/blog/top-10-open-source-security-testing-tools-for-web-applications
2. https://resources.whitesourcesoftware.com/blog-whitesource/secure-coding
3. https://www.sentrian.com.au/blog/a-short-history-of-computer-viruses
4. https://study.com/academy/lesson/secure-software-definition-characteristics.html