Module 1: Activity 1

Harsh Siddhapura

Ira A. Fulton Schools of Engineering, Arizona State University

IFT 520: Advanced Information Systems Security

Dr. Jim Helm, Prof. Upakar Bhatta

August 24, 2023

“Solve the following three problems.”

1. **“Privacy and anonymity”**

“A service lets users browse the Internet anonymously through a network of proxy servers. To avoid having their identities compromised, users must install special software that modifies the way their web browsers behave.”

“In the absence of each change or action listed below, describe an attack that would reveal information about the user’s identity. Assume that the user regularly visits web sites under adversarial control, but that the adversary is physically separated from the user.”

“Changes that apply while the anonymous browsing service is enabled:”

(a) “Disable the HTTP Referer header”

HTTP referer header includes address of the source from which the site is requested. Referer header sometimes works as an optional header field and often ignored or blocked by firewalls for security purposes. As the referer header includes the source configuration, it is easy for a website to track down the user’s activity and other personal information.

However, referer header, no longer being an important header field, can be disabled in order to hide the source address. If the user fails to disable the http referer header, source address along with other confidential user data could be compromised [8] [9].

(b) “Disable Java and Flash”

Java (Object Oriented Programming Language) and Flash (multimedia software by Adobe) are the most vulnerable tools for cyber-attacks. Java and Flash are extensively used across many organizations and often remain unpatched due to high-maintenance cost and fear of website crash. The organizations using outdated versions of Java and Flash give the attacker a chance to sneak peek into their system and steal

confidential data. If the user is using Java and Flash both, they must patch them regularly to keep up to date with compatible and secure versions [10] [11].

(c) "Set the HTTP User-Agent header to a generic value"

Http User-Agent header is required to make the requesting site aware of the browser configuration. The website checks from which browser the request is coming and upon validation it sends the compatible response. User – Agent header helps to identify websites whether the requester is genuine or not. If the website doesn't recognize the User-Agent header, it does not provide proper response to the browser and sometimes blocks the traffic as well. It is always recommended to set the User – Agent header to a well-supported browser configuration so that the website trusts the request and responds accordingly [12] [13].

(d) "Resize browser windows to multiples of 35 pixels"

If the user is using TOR and tries resizing the window, there would be a potential that the user may be tracked. TOR browser opens a certain window size for many users to group together so that individual users are untraceable. Resizing the window creates unique web fingerprints for the user at that window size during the user's activity. This unique fingerprint will remain unaltered until the browser size is reset. This scenario allows attackers and advertisers to track a user's web activity [14] [15] [16] [17].

"Actions that occur each time the user enables or disables the anonymous browsing service:"

(e) "Close all open sites"

Any currently active sessions or cookies from the earlier browsing session can possibly be used to link the user's identity and actions if the user does not remove all open sites after enabling or disabling the anonymous browsing service [18].

(f) “Clear the history list”

Failing to remove the history list after enabling or stopping the anonymous browsing may allow an intruder to gain access to the user's device to view the user's browsing history, compromising their identity [19].

(g) “Clear the cookie store”

If the user does not clear the cookie store after setting or disabling the anonymous browsing service, tracking cookies from prior sessions may still be there, allowing websites to correlate the user's activities across sessions [20].

(h) “Clear the browser cache”

If the user does not clear the cookie store after setting or disabling the anonymous browsing service, tracking cookies from prior sessions may still be there, allowing websites to correlate the user's activities across sessions [21].

2. **“Secure programming. StackGuard is a mechanism for defending C programs against stack-based buffer overflows. It detects memory corruption using a canary, a known value stored in each function’s stack frame immediately before the return address. Before a function returns, it verifies that its canary value hasn’t changed; if it has, the program halts with a security error.”**

(a) “In some implementations, the canary value is a 64-bit integer that is randomly generated each time the program runs. Why does this prevent the basic buffer-overflow attack discussed in lecture?”

The inclusion of a randomly generated 64-bit integer as a canary value in the StackGuard method aids in the prevention of the basic buffer overflow attack by adding unpredictability and difficulty for attackers to exploit. This is how it works:

- Randomness: Each time the program is run, the canary value is created at random. This means that each time the program is executed, a fresh, distinct canary value is used. Because of this randomness, an attacker would have a very difficult time predicting the exact value of the canary ahead of time. Because the attacker does not know which canary value to overwrite, their attempts to overrun the buffer with malicious code or data are thwarted.
- Corruption Detection: Because the canary value is placed directly before the return address in the stack frame of a function, any attempt by an attacker to replace the canary value while overflowing the buffer will be detected during the function's return process. The function compares the current canary value to the original canary value as it prepares to return. If the canary is altered, suggesting a possible buffer overflow or corruption, the program is terminated with a security fault.
- Exploit Difficulty: Because the attacker does not know the exact canary value to utilize in their attack, they cannot properly estimate the offset in memory where the canary is placed. This means they won't be able to design a buffer overflow payload that will successfully evade the canary defense.

In essence, StackGuard's use of a randomly generated 64-bit integer canary avoids the fundamental buffer-overflow attack by making it extremely difficult for attackers to predict the canary value, limiting their capacity to create a successful exploit. This adds another layer of protection against memory corruption attacks [6].

(b) “What are the security drawbacks to choosing the canary value at compile time instead of at run time? Why do some implementations use 0 for the canary anyway?”

Choosing the canary value at compile time rather than run time has certain security implications in the context of stack protection methods such as StackGuard. Here are the main disadvantages:

- Predictability: If the canary value is chosen at build time and remains constant across all program executions, both legitimate users and possible attackers will be able to forecast it.

Attackers can find the canary value by analyzing the binary or executable, making it easier for them to design effective buffer overflow attacks. The canary's effectiveness as a defense mechanism is hampered by its lack of unpredictability.

- Compile-time canary values are incapable of adapting to the dynamic nature of runtime execution. In some circumstances, the identical binary may be loaded into various memory addresses throughout different runs, and an attacker may be able to take advantage of this lack of adaptation to bypass the static canary value.
- Reduced Resilience: Canary values at compile time are static and cannot be altered without recompiling the application. If a canary value is discovered or leaked, an attacker can reuse it over several executions of the program, decreasing the defense mechanism's overall resilience.

Concerning the usage of 0 as the canary value in some implementations, it is crucial to highlight that this is not a recommended practice due to its own set of security concerns:

- Zero Initialization: By default, some compilers set variables and memory addresses to 0. If a canary value of 0 is used, this default initialization behavior may mistakenly set the canary to 0, even if a different value was intended. As a result, attackers might more easily foresee and manipulate the canary, thereby rendering the defense system worthless.
- Bugs involving zero-checking: Certain programming faults may involve checks for zero values. If the canary is set to 0, it may interfere with these checks, resulting in vulnerabilities or unexpected behavior.

Overall, setting the canary value to 0 defeats the purpose of having a canary in the first place, as it increases predictability and potential conflicts with established programming methods. Instead, the canary value should be chosen with randomness and uniqueness in mind in order to effectively improve the program's security against buffer overflow attacks [7].

(c) “No matter how the canary is chosen, StackGuard cannot protect against all buffer overflow vulnerabilities. Describe two kinds of bugs that can corrupt the stack and allow the adversary to take control, even with StackGuard in place.”

StackGuard is present, but there are still two types of bugs that might damage the stack and let an enemy take over:

- Stack Smashing Attacks: In circumstances when the buffer overflow is severe enough, an attacker can overwrite not only the return address and canary but also the most crucial data that is present on the stack which includes function pointers or local variables. By changing these values, it allows an attacker to get beyond the canary protection and route the program's execution flow to malicious code they have injected into the compromised buffer.
- Return-Oriented Programming (ROP): Without adding new code, attackers can still create dangerous payloads by using Return-oriented Programming methods. They incorporate "gadgets" or small pieces of already-existing code which are stored in the program's memory. An attacker is able to chain the 'ret' instructions that are present at the end of devices by changing the return addresses which are present on stack. Although StackGuard's canary might prevent direct changes to the return address, an attacker can utilize return-oriented programming to carry out chains of already-existing instructions that serve the same purpose, thus taking over the program's control flow.

3. **“Ethics and law. Consider the following scenario: A worm is infecting systems by exploiting a bug in a popular server program. It is spreading rapidly, and systems where it is deleted quickly become reinfected. A security researcher decides to launch a counterattack in the form of a defensive worm. Whenever a break-in attempt comes from a remote host, the defensive worm detects it, heads off the break-in, and exploits the same bug to spread to the attacking host. On that host, it deletes the original worm. It then waits until that system is attacked, and the cycle repeats.”**

(a) “Many people would claim that launching a counterattack in this scenario is ethically unacceptable. Give at least three arguments that support this view.”

The first argument against counter attacks is that they tend to affect benign third-parties involved. In such instances of the worm propagating through a popular server program, one can expect a few systems running this program to already be affected, and this worm pivots to other end user devices, thereby maintaining persistent access. Launching an active attack that will, for example, identify the source of the worm, reverse engineer its properties to purge at source folder will have widespread casualties beyond the original host that spread it initially. Also, say the worm was deleted from source during the counter attack, this will result in the loss of forensics data that would justify the counter attack in the first place.

Thirdly, attackers propagate worms using “seemingly harmless” files that download into an end user's workstation and launch from it. Sending a “delete-at-source” worm will inadvertently lead to data loss for the third-party user. There are much wider consequences if that worm-host turns out to be a primary domain container for a critical state entity (Gupte, P., 2019).

The fourth argument against counter attacks is the violation of end user privacy. If a user’s device is infected by the worm, and is now the primary broadcaster of the worm, launching a counter attack that also aims to analyze the behavior of the work at source is a serious violation of privacy, because the owner of the workstation’s activity and data will be exposed. This may also lead to legal challenges in various geographical regions in which the worm-infested device is located.

(b) “Are there circumstances or conditions under which a counterattack would be ethically justified? Explain your reasoning.”

Yes, provided there is sufficient evidence to support the counterattack. For example, a known persistent attacker, routinely trying to illicitly disrupt access to critical infrastructure systems and databases, such as hospitals, financial institutions and utility companies. In this regard, the consequences of worm

propagating into critical infrastructure i,e., loss of lives, endangerment to public services and more, far outweighs the risk of third-party casualties (Wolff, J., 2017).

“In 1988, a graduate student accidentally unleashed one of the first Internet worms. It deleted no files but caused widespread service interruptions. Cleaning it up and implementing defenses were costly. The student was convicted of violating the Computer Fraud and Abuse Act.
Despite demands that he be sent to prison for the maximum time possible (to make an example of him), the judge sentenced him to pay a fine and perform community service. Today that student is a professor at MIT.”

(c) “What factors do you think caused the judge to hand down this sentence? Do you believe the outcome was fair?”

The individual involved in this incident was Robert Tappan Morris, currently a professor at MIT and a notable contributor to the field of computer science. As a graduate student, he developed the Morris Worm to identify vulnerabilities in Unix systems. This worm inadvertently spread through the network by exploiting weak passwords. Notably, Morris did not foresee the worm's rapid replication. Several factors likely influenced the judge's decision not to impose a jail sentence:

- Limited Precedent and Knowledge in the Legal Sphere: Given the nascent state of computer-related cases, the judge grappled with an absence of established legal precedents and a clear understanding of the potential consequences, complicating the determination of an appropriate sentence.
- Unintended Impact: He acknowledged his lack of anticipation for the worm's swift propagation, potentially leading the judge to perceive the resulting damage as unintentional rather than malicious.
- Youthful Offender: Morris was a young graduate student without a history of criminal activity. His age and student status may have inclined the judge to lean towards a more lenient sentence, taking into account the potential for rehabilitation.

- Public Discourse: The case triggered widespread public debate on how to address computer-related offenses. Some voices advocated for a more lenient approach, emphasizing the educational and experimental aspects of Morris's actions alongside punitive measures. This ongoing discourse might have influenced the judge's considerations.

In light of these factors, the judge's decision not to impose jail time underscores the evolving nature of early computer-related offenses and the complexities surrounding their adjudication.

(d) "Would you expect a different outcome if a similar case happened today? Suppose you were the judge; what information would you take into account in making your decision?"

Given the advancements in technology and heightened cybersecurity awareness, were I to serve as a judge in a comparable case today, I would weigh several considerations to determine the most suitable course of action:

- Purpose and Drive: Carefully scrutinizing the defendant's motives—whether they originated from malevolent intent, monetary incentives, experimentation, or were inadvertent—would be of utmost importance.
- Degree of Damage: Assessing the breadth of harm suffered, spanning financial ramifications, data breaches, and violations of security and privacy, would heavily impact the decision regarding the fitting sentence.
- Cybersecurity Consciousness: Considering an individual's grasp of the repercussions stemming from their actions in the context of the present cybersecurity environment would hold significance.

Given the prevalence of numerous cyber attacks and advancements within the technology industry, I would ensure to incorporate these factors into the sentencing decision.

References

[1] https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2016/wp238_en.pdf

[2] https://csrc.nist.gov/publications/detail/nistir/7564/final

[3] https://www.pcworld.com/article/2013534/how-and-why-to-surf-the-web-in-secret.html#:~:text=Powerful%20options%20to%20browse%20the%20Web%20through%20a,info%20for%20the%20proxy%20server%2C%20then%20click%20OK

[4] https://www.wikihow.com/Surf-the-Web-Anonymously-with-Proxies (Links to an external site.)

[5] https://mcuoneclipse.com/2019/09/28/stack-canaries-with-gcc-checking-for-stack-overflow-at-runtime/

[6] Cowan, C., Wagle, P., Pu, C., Beattie, S., Kroah-Hartman, G., & Wright, C. (2003). StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. Proceedings of the 12th Conference on USENIX Security Symposium.

[7] Cowan, C., Beattie, S., & Johansen, J. D. (2005). PointGuard: Protecting Pointers from Buffer Overflow Vulnerabilities. Proceedings of the 14th USENIX Security Symposium.

[8] https://community.atlassian.com/t5/Answers-Developer-Questions/In-browsers-with-the-referrer-header-disabled-pages-error/qaq-p/571717

[9] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer

[10] https://www.beyondtrust.com/blog/entry/taking-the-fear-out-of-java-and-flash-vulnerabilities

[11] https://www.pcworld.com/article/432699/the-most-foolproof-java-and-flash-security-fix-is-the-one-thing-few-people-do.html

[12] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

[13] https://www.sistrix.com/ask-sistrix/getting-started-seo/what-is-a-user-agent/#:~:text=The%20user%2Dagent%20is%20an,are%20capable%20of%20managing%20them

[14] https://news.ycombinator.com/item?id=10833629#:~:text=Tor%20Anonymity%3A%20Things%20Not%20to%20Do%20%7C%20Hacker%20News&text=Here's%20one%20they%20missed%3A%20do,probability%20you%20can%20be%20tracked

[15] https://www.reddit.com/r/TOR/comments/7e1qd1/how_bad_is_resizing_the_tor_window_anonymitywise/?rdt=43238

[16] https://support.torproject.org/tbb/maximized-torbrowser-window/

[17] https://tor.stackexchange.com/questions/16111/is-manually-resizing-the-tor-window-dangerous/

[18] *How private browsing works in Chrome - Android - Google Chrome Help*. (n.d.). How Private Browsing Works in Chrome - Android - Google Chrome Help.

https://support.google.com/chrome/answer/7440301?sjid=6648538199600623585-NA

[19] *Chrome Browser Privacy Policy - Google Chrome. (n.d.)*. Chrome Browser Privacy Policy - Google Chrome. A https://www.google.com/chrome/privacy/

[20] *How to Enable and Disable Cookies on Every Web Browser [Guide]*. (2014, October 28). All About Cookies. https://allaboutcookies.org/how-to-manage-cookies

[21] *Clear cache & cookies - Computer - Google Account Help*. (n.d.). Clear Cache & Cookies - Computer - Google Account Help. https://support.google.com/accounts/answer/32050?sjid=6648538199600623585-NA

[22] Gupte, P. (2019). *Attacking the attacker: Analyzing the legality of hacking back*. LinkedIn. https://www.linkedin.com/pulse/attacking-attacker-analyzing-legality-hacking-ba-pranav-gupte

[23] Wolff, J. (2017, July 14). *When companies get hacked, should they be allowed to hack back?* The Atlantic. https://www.theatlantic.com/business/archive/2017/07/hacking-back-active-defense/533679/