Out-of-band Application Security Testing: Significance and Modern Detection Approaches

CS4991 Capstone Report, 2024

Nicholas Tung
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ddn9nb@virginia.edu

ABSTRACT
A software-as-a-service company's product security team decided to stand up its own infrastructure for performing regular out-of-band application security testing. I brought this system from non-existent to a validated prototype by hosting an instance of Interact.sh, a beacon detection service, and using it for various methods of security testing. The first step was to work locally and show that I could positively identify a vulnerable service using Interact.sh. After local validation, I performed similar steps and configuration for an instance of Interact.sh available on the Internet, a requirement for a production instance. With a demonstrably effective instance running, I moved on to integrating Interact.sh with a variety of security testing tools, both automated and manual. I was able to use Interact.sh with three different tools on production systems, all while learning about the nuances and significance of out-of-band vulnerabilities. With this infrastructure in place, the team can move toward more advanced security testing, both automated and manual, and know that any relevant data is under the company's control.

1. INTRODUCTION

On December 10, 2021, the vulnerability known as "Log4Shell" was disclosed to the public. A misconfiguration in the popular Java logging library Log4j enabled substitution of a template string with a Java object when the template string was logged by Log4j. Crucially, substitution of this template string could, depending on the particular template string, trigger the lookup and execution of remotely-hosted code. At the time of public disclosure, Log4Shell affected numerous cloud services, as well as popular applications like Minecraft (Goodin, 2021). Log4Shell shook the world as security professionals rushed to determine if and how they were affected and patch their systems.

Log4Shell was a vulnerability of unique severity, but similar vulnerabilities are discovered and disclosed every year. Vulnerabilities that have similar communication patterns as Log4Shell are known as out-of-band vulnerabilities. The importance of both accuracy and precision in detection has led to various innovative strategies being developed for out-of-band vulnerabilities.

2. RELATED WORKS

Tracking DNS lookups is a versatile method for detecting out-of-band vulnerabilities. DNS-based detection is effective for finding so many types of communication that the likelihood of a false negative for some kind of vulnerability existing is limited (*Out of Band Exploitation (OOB) CheatSheet*, 2018). The tradeoff comes in precision. When a DNS lookup comes in, the detecting party has no information about what type of communication is being initiated. It could range from attempting and failing to send an email (probably not a big deal) to downloading and executing code (definitely a big deal).

To address the precision constraints of tools based solely on tracking DNS lookups, detection tools implement additional functionality. One example of a tool that implements additional functionality is BOAST. It is a “server built to receive and report Out-of-Band Application Security Testing interactions”. It integrates with ZAP, an open source attack proxy used for penetration testing, and implements receivers for not only the DNS protocol, but also the HTTP and HTTPS protocols (*ZAP – BOAST*, n.d.). HTTP and HTTPS are incredibly common protocols and are used for a significant portion of web-based communication, making their implementation a very pragmatic choice. Using BOAST will provide additional information about a detected vulnerability, but only if that vulnerability goes on to initiate HTTP or HTTPS communication. Many variations of Log4Shell use the LDAP protocol, so communications triggered by Log4Shell will stop at DNS lookups when using BOAST. The information provided by BOAST is useful, but even better tools exist.

3. PROJECT DESIGN

This project leverages Interact.sh, a cutting-edge tool for detecting various types of out-of-band vulnerabilities. To understand the importance of Interact.sh features and how this project leverages the tool, a technical knowledge base must be established.

3.1 Out-of-band vulnerability components

Multiple types of exploits can be used to compromise networked applications. Many follow a similar pattern: send an exploit payload to a service (ex. submit a username and password to a website’s login form) and see if the service responds with something interesting (ex. does the exploit enable logging into someone else’s account?).

However, the communication involved with Log4Shell followed a different pattern. The first step stayed the same: send an exploit payload to a service. However, there was not typically any more significant communication along that channel. In the service login example, the result of username and password submission is relatively inconsequential. The real goal is getting the username or password processed and logged by a vulnerable Log4J. Causing remote code lookup via template string substitution triggers outbound communication from the vulnerable service to some malicious host. Crucially, this communication is on a different channel: it is not related to the method of sending the exploit payload and could occur far after the exploit is sent. Vulnerabilities that involve this pattern of multi-channel communication are known as out-of-band vulnerabilities or second order vulnerabilities.

Detecting out-of-band vulnerabilities and gleaning useful information about their presence is a tricky problem. There are various scenarios where the location of the vulnerability is unclear. It is not necessarily the original target of the exploit payload, and the consequence of the exploit may be something unexpected. There is a tradeoff when making additional assumptions about what an exploit will do to the target system (if anything). Assumptions can help guide the detection strategy and provide precision when reasonable, but they could also cause one to narrow their search too far and miss an important signal.

3.2 Out-of-band vulnerability detection

Traditionally, out-of-band vulnerabilities are detected by tracking DNS lookups. To provide a brief overview of networked communication, computers are addressed on a network by their Internet Protocol (IP) address. A computer's IP address is just a number; network infrastructure can identify a computer and direct traffic to that computer by its IP address. However, IP addresses are not very ergonomic; it would not be ideal to only use the IP address 172.253.115.113 to direct traffic to Google. On top of carrying minimal semantic information, IP addresses for a computer or a network can change. To work around these constraints, the Domain Name System (DNS) is used. This system effectively maps domains that many people are today familiar with (ex. google.com) to useful information for networked communication (IP addresses, among other things). For a computer to find the IP address (useful to the computer) mapped to a given domain, it must perform a DNS lookup. To over-simplify things a bit, this can be thought of as asking the DNS (a collection of servers) the following: "what is the IP address of this domain?". A computer somewhere is responding to this DNS lookup, with a process similar to other common request-response communication.

The noteworthy insight is that since DNS lookups are served by a program running on a computer somewhere, having a program that both serves DNS lookups and logs the lookup event is useful. A DNS lookup is needed to initiate a wide swath of communication archetypes. Whether it is loading a website, sending an email, or transferring a file, the networked communication for that action probably starts with a DNS lookup.

Differentiating between communication archetypes is difficult to do with just DNS lookup tracking. Tools like BOAST can recognize one popular protocol, HTTP(S), but fall short when other types of communication are initiated. Interact.sh is unique in this respect, because it can detect DNS, HTTP(S), SMTP(S), and LDAP interaction (*projectdiscovery/interactsh*, n.d.). It also integrates easily with ZAP and provides additional useful functionality. Detection tools built on top of Interact.sh are versatile, informative, and easy to work with.

3.3 Additional requirements

To build useful detection tools, additional criteria must be met. Interact.sh is a server program that other tools can point to, so it must be hosted somewhere. Freely available instances exist, but using a public instance means vulnerability data will be sent to an unknown third party. Thus, it is necessary to host a private instance on controlled and trusted infrastructure.

Detection tools can be useful in isolation, but leveraging new tools as components in a greater system can be more efficient. This company has pre-existing security automation, so a detection tool that also integrates with this system is required.

Finally, an easily-forgotten requirement of detection tools is proof that the tool works. There is extraneous technical setup and programming work required to prove that the detection tool performs as expected. Successfully detecting a known vulnerability in a realistic but controlled environment provides a reasonable level of confidence in the tool's ability to detect a real vulnerability.

4. RESULTS

The first half of the internship was spent developing a strong foundation for a production detection tool. After gaining a strong understanding of detection mechanics and the unique affordances of Interact.sh, a local prototype was developed. Once the local prototype successfully detected a vulnerability in a controlled environment, Interact.sh was deployed to cloud infrastructure and the test was repeated.

The second half of the internship expanded on this work, prototyping various tools that leveraged Interact.sh. Manual ZAP scans of production Internet-facing services were performed with new Interact.sh access. An additional stage of automated security testing was developed, using Interact.sh and integrating with existing execution and data systems. Finally, a new security testing program was prototyped, enabling programmatic detection of thousands of variations of a particular vulnerability instead of manual testing of individual variations.

5. CONCLUSION

Out-of-band vulnerabilities remain a noteworthy problem in the cybersecurity world. They are not trivial to detect, and they may grow more relevant as new software communication patterns emerge. This project demonstrated an effective approach for creating an out-of-band application security testing basis. Understanding the shape of the problem enabled evaluation of solutions. After performing incremental testing on the Interact.sh setup to ensure efficacy, various practical prototypes were developed that strengthened detection capabilities.

6. FUTURE WORK

There is always more that can be done to increase detection capabilities with this project. Prototypes developed during the internship primarily relied on manual activation; further work could integrate Interact.sh into regular automated scans. The deployment of an Interact.sh instance was also relatively brittle, so greater automation and process surrounding deployment could be developed.

REFERENCES

Goodin, D. (2021, December 12). The Log4Shell 0-day, four days on: What is it, and how bad is it really? *Ars Technica*. https://arstechnica.com/information-technology/2021/12/the-log4shell-zeroday-4-days-on-what-is-it-and-how-bad-is-it-really/

Out of Band Exploitation (OOB) CheatSheet. (2018, August 30). NotSoSecure. Retrieved February 23, 2024, from https://notsosecure.com/out-band-exploitation-oob-cheatsheet

projectdiscovery/interactsh: An OOB interaction gathering server and client library. (n.d.). GitHub. Retrieved March 22, 2024, from https://github.com/projectdiscovery/interactsh

ZAP – BOAST. (n.d.). ZAP. Retrieved February 23, 2024, from https://www.zaproxy.org/docs/desktop/addons/oast-support/services/boast/