WHITEPAPER

SPINNAKER™
SUPPORT

The Zero-Day, No-Patch Scramble

Reducing Your Risk to Zero-Day Vulnerabilities

Table of Contents

Abstract

Apache Log4j is a free, open source of software, that developers have readily included in application code for years. As a result, when it was announced that there was a critical vulnerability in Log4j and no available patch the tech world was sent scrambling. Vendors rushed to determine if their developers had included this the log4j code in their applications. While vendors scrambled to analyze and prepare patches for affected software, companies across the world scrambled as well. Staff members worked long hours to understand if company systems could be or perhaps were already compromised. They brainstormed and kept searching the internet to figure out how to prevent exploitation of the vulnerability. They continuously monitored vendor and technical sites for remediations and news on forth coming patches.

The question is how can companies avoid the no patch scramble? A company needs to be confident in their ability to prevent exploitation of zero-day vulnerabilities while they wait days, weeks or even months for vendors to provide patches. The intended audience for this paper is the administrators involved with the application, user, and data. This paper will discuss the benefits and disadvantages that applying patches to address these high-profile vulnerabilities. Next it will identify practical steps if followed could reduce a system's risks to zero-day vulnerabilities like Log4j. Finally, the paper will cover foundational concepts that can be incorporated into a company's zero-day action plan.

Introduction

Trend Micro (Trend Micro, 2021) defines Zero-day as a vulnerability in a system or device that has been disclosed but is not yet patched. According to Mandiant (Sadowski, 2022), there were 80 zero-days exploited in the wild which is more than double the previous high record set in 2018. Project Zero team at Google (Stone, 2022) reported 58 detected and disclosed zero-days which was the double the previous record recorded in 2015. The intelligence firm Mandiant and Google's Project Zero's numbers are different because the types of zero-day bugs in scope are different. However, both show a dramatic increase in the number detected and disclosed. Those statistics bring good news and bad news. The good news is that researchers believe that the substantial increase is partly due to organizations enhancing their defenses and tightening security protocols leading to substantial increase in detection and disclosure of vulnerabilities. The bad news is that you don't know what you don't know. Attackers who are actively using zero-day exploits do not share which ones they are using that have not been publicly disclosed yet. So, there is no way to know exactly which proportion of zero-day vulnerabilities have been found and which have not.

Attackers are using the same bug patterns, attack surfaces, and types of exploits that have been shared in public research.

The Project Zero team highlight in their paper (Stone, 2022), that most of the vulnerabilities were of the same flavor of previous vulnerabilities. Attackers are using the same bug patterns, attack surfaces, and types of exploits that have been shared in public research. You would think that once a type of attack had been shown to work that vendors and organizations would harden their systems to force attackers to find new bug classes of vulnerabilities or attack surfaces but that is not the case. Project Zero found that in 2021, over 56 of the 58 zero-day vulnerabilities reviewed were using the same standard attacks and bugs that had been seen previously.

When a vulnerability is identified within a product or system it is reported to the CVE Program and provided a Common Vulnerability and Exposures (CVE). This process is laid out on the cve.org website https://www.cve.org/About/Process. When a vendor is supplying a security patch the CVEs are listed within the release notes for the patch.

PATCHING BENEFITS AND DISADVANTAGES

There are two key pieces that make up an exploit. The vulnerability being exploited and how the vulnerability is used. Vendors address the first key piece by providing software patches. In a paper sponsored by the Oracle Corporation (Olofson, 2020), Carl Olofson defined a patch as “a piece of code that is inserted into existing software to alter its behavior.”

Patches are generally released by vendors for one of three reasons.

- Fix a known issue (bug)
- Enhance or provide a requested feature
- Remove a security vulnerability

These are great reasons to patch. You also must balance this out with the disadvantages that are inherent in patches.

Patches do not always work

Software is very complex which can lead to a piece of code having multiple vulnerabilities. Consider the recent log4j vulnerability. On December 9, 2021, security researchers discovered a flaw in the Log4j software code. The software vendor Apache released a fix on December 10th (version 2.15), but it was incomplete. A new patch was released on December 13th (version 2.16). Two new issues were confirmed, and another patch was released on December 17th (version 2.1.7). An yet another issue required a patch on December 28th (version 2.17.2). Each new flaw received its own CVE and is considered a new vulnerability in the log4j software code.

Patches not always available for older product versions

Vendors will often stop providing patches for older versions of software. Patching for vulnerabilities often does continue for a time beyond bug fixes but eventually most companies stop supplying even security fixes. Companies must then decide if they go through the process and expense of upgrading their software so they can continue to receive vulnerability patches. This can be a huge decision for some companies, particularly when the current software meets all other needs and requirements of the company. It can a real issue for companies that do not have the necessary resources like down time, software expertise, funds, or backing from upper management for such an undertaking.

Patches are one-size-fits all

The Oracle April 2022 Critical Patch Update included a patch for the CVE-2022-21449 vulnerability nick named “Psychic Signatures”. The vulnerability was in the implementation of the Elliptic Curve Digital Signature Algorithm (ECDSA). In a nutshell, when Java 15 was written the code to validate the ECDSA signature did not prevent 0 from being submitted as a key. Hackers could submit a 0 as a key response and the application would validate the signature. This allowed hackers to intercept and read communications in clear text. However, just because you were running Java 15 did not mean that you were vulnerable. If you were not using ECDSA signatures there would be no need to apply the patch. Code is complex and a fix in one place could cause unforeseen issues in other places. Every system is different and even if the patch works in one environment it could still cause issues in another. It is a best practice to read and understand what a patch does and evaluate if it is applicable to a system before applying it.

Patches require valuable time to install and test

Due to operational constraints organizations often do not regularly patch. It is difficult for some companies to set aside time on a regularly to perform system maintenance. Depending on the software system the number of changes a patch performs, it could mean hours of downtime to apply a patch. This issue gets multiplied for each vendor product a company uses within the business.

Patches are not always timely

Apache released several patches for Log4j vulnerabilities during the first 19 days after the vulnerability was announced. Vendors like Oracle must provide patches for their own software to address the integrated code vulnerability. They would need to retest any patched products again with the new Log4j version. This can really hinder a vendors release schedule for their own patches. Oracle started releasing log4j patches to their massive software library in December. Oracle's January 2022 Critical Patch Update (CPU) Advisory included 35 references to log4j patches, the April 2022 CPU had over 100 references, and even the January 2024 CPU included a patch for log4j. This lag in patching is not just for zero-day vulnerabilities. It can take years for a vendor to patch a known vulnerability. The table Oracle July 2024 CPU CVEs by Year shows the approximate number of unique vulnerabilities mentioned in Oracle's July 2024 Critical Patch Update Advisory, as referenced at https://www.oracle.com/security-alerts/public-vuln-to-advisory-mapping.html

CPU Publish Date	log4j Patches
Jan 2022	36
Apr 2022	110
Jul 2022	17
Oct 2022	16
Jan 2023	8
Apr 2023	3
Jul 2023	2
Oct 2023	0
Jan 2024	1

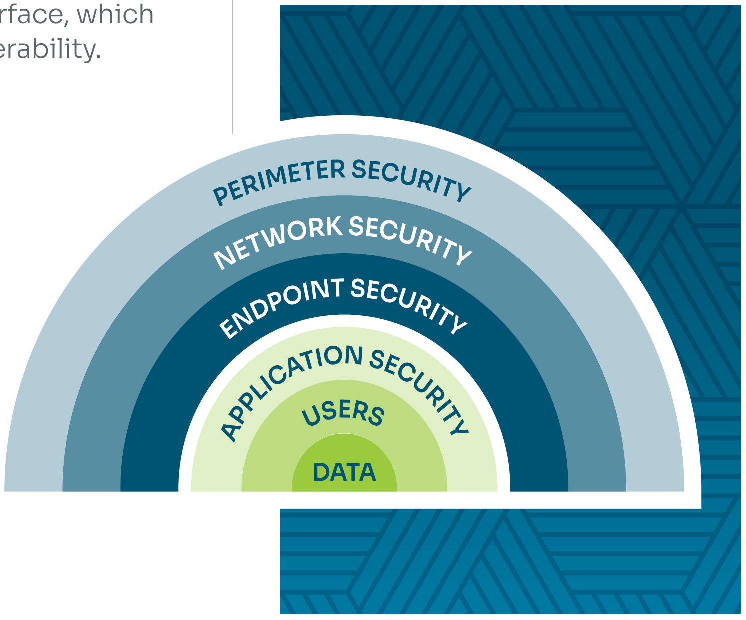
Year of Vulnerability	July 2024 CPU - Count
2019	3
2020	3
2021	13
2022	32
2023	66
2024	119

Security patches by vendors focus on addressing known vulnerabilities being exploited. There are benefits and drawbacks to patching. However, even if there were no drawbacks to patching and only benefits it does not help for zero-day exploits. By definition, a zero-day is a vulnerability in a system or device that has been disclosed but is not yet patched. The second part of an exploit is how the vulnerability is being used. Research has noted that most zero-day vulnerabilities are using the same standard attacks and bugs previously disclosed. Utilizing methods to address attack vectors and limit the attacker's ability to utilize software bugs will reducethe number of zero-day vulnerabilities to which a system is susceptible.

An overall security profile involves more than just patching. Software patching focuses on addressing exploits by fixing a vulnerability identified at the product level. That is why you will see the information on patches listing the CVEs they address. The second key piece if preventing an exploit is addressing how the vulnerability is being used.

Defense in Depth (Layered Protection)

How a vulnerability is being used can be classified in weakness categories. The same core supporters of the CVEs also manage the Common Weakness Enumeration (CWE) list. In their own words “CWE™ is a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts (CWE, May 2022).” Most companies focus on fixing issues by patching at a product or system which is at the CVE level. A more encompassing approach is to harden a system against the category of vulnerabilities or at the CWE level. Hardening against a category of weaknesses reduces the attack surface, which limits the ability for attackers to exploit a vulnerability. Reducing the attack surface available to hackers is done by hardening each surface that an attacker needs to cross to breach a system. At Spinnaker Support we refer to this as Defense in Depth.



There are six layers of security that should be considered: perimeter security, network security, endpoint security, application security, user security and data security.

Through the hardening of each layer, the possibility of exploiting an application or system vulnerability be reduced or eliminated altogether.

Core Actions to Reduce Risk

There is no way to cover all the actions that can be used to reduce the risk of a system being breached in a simple white paper. All systems are different and have their own unique needs. The actions each person can take also varies by their position and responsibilities on the systems they support. The hope is that you can use the following ideas as a springboard to generate your own list of actions you can take and a list of actions you can champion to have done on the systems you support.

HAVE AN OVERVIEW OF YOUR OUTER DEFENSE

It should not be expected that a single person knows all the details of the system or systems they support. However, you should have a general knowledge of the defenses in place and who to contact for support on each layer. Here are some questions you can ask yourself to evaluate your knowledge and areas you might want to research a little more on your system's outer defense.

Perimeter Security

Which of our applications can be accessed from the internet?

Do we have non-production / development environments accessible from the internet?

Do we have an intrusion detection system, or an intrusion prevent system or both?

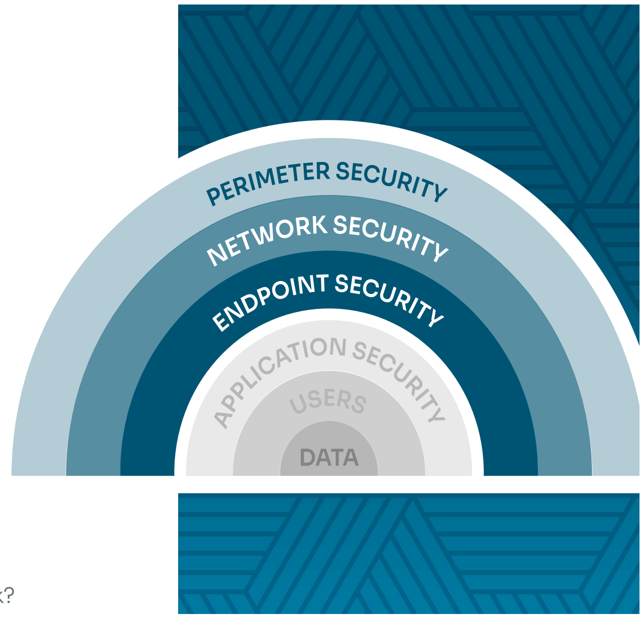
Who do I contact if I see unknown IP addresses accessing our system?

Are we using API firewalls?

Does the company do penetration test or vulnerability scans?

Do we have an asset management tool that we can get a current list of assets associated with each application / system?

When systems are added/removed what changes occur on the perimeter security?



Network Security

Do we separate our development network from production network?

Do we separate systems to prevent lateral movement between systems?

What methods are used to limit access between internal systems?

Do we encrypt network traffic between internal systems?

Do we monitor / alert for excessive network traffic?

Do different networks have different speeds?

Endpoint Security

How are our systems accessed (local desktops, smart phones, laptops, tablets)?

Are there devices that access our systems (sales terminals, online stores, printers, faxes, scanners)?

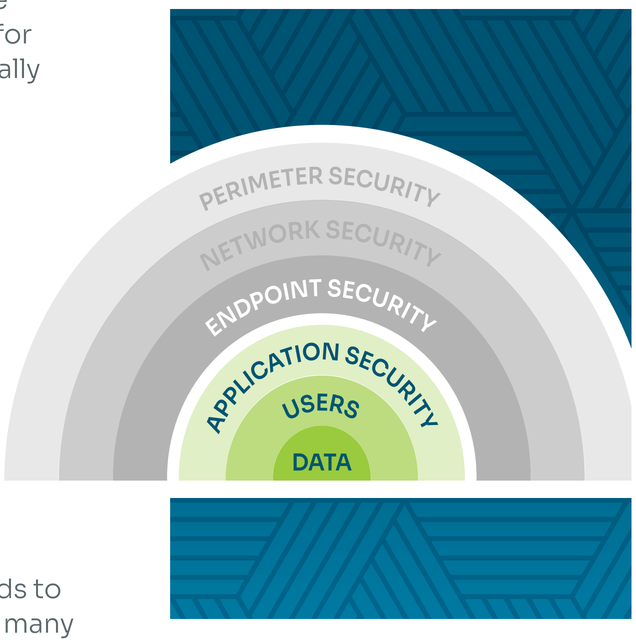
What endpoint security solutions does our company use?

What features is the company using (malware detection, malicious script protection, phishing protection, white and blacklisting)?

EVALUATE THE INNER DEFENSE

This paper was written with the application, user, and database administrators as the primary audience. When considering layers of defense, these administrators are the ones with the highest amount of influence and responsibility for the inner three layers: Application Security, Users and Data.

When evaluating the inner defense, the best approach is from a CWE level. This might seem overwhelming because there are over 900 weakness that have been identified. Knowing where to start and which CWEs are applicable is the key. Probably of the highest concern for most administrators are web applications and specifically ones that are accessible from the internet. The non-profit foundation Open Web Application Security Project® (OWASP) exists to educate companies, administrators, and developers on how to minimize web application security risks. OWASP brought together a group of web application security experts from around the world and created the 2021 Top 10 vulnerability list (https://owasp.org/Top10/). OWASP does a fantastic job of providing details information on the top 10 vulnerabilities. Including a description of the vulnerability, steps on how to prevent it, example attack scenarios, and a list of mapped CWEs. There are many suggestions, and each would need to be evaluated to see if could or even needs to be applied to an environment. The list is still large, and many suggestions are geared towards the developers. However, there are some core actions that are administrators should investigate performing. The table below provides core actions that may represent some effort into performing but will reduce system risk to the described vulnerability.



NOTE: The impact of these actions should be negligible on system performance but as always should be applied using normal change control process that includes testing on non-production systems before performing changes on production system.

Title	Description	Core Actions to Reduce Risk
Broken Access Control	When restrictions on the ability of authenticated users to perform actions outside their level of permission are not correctly enforced.	Regularly audit and delete inactive or unnecessary accounts. Regularly audit accounts for changes in privileges needed (enforce principle of least privilege). Shutdown unnecessary access points. Audit and eliminate services that are not needed on the server.
Cryptographic Failures	Failures related to cryptography or the absence of cryptography altogether.	Audit system to verify data protections needs in transit and at rest and apply appropriate Review configuration of applications for utilization of old or weak cryptographic algorithms. Verify HTTP headers (browser) security directives and headers are defined. Audit for deprecated hash functions or utilization of non-cryptographic hash functions. Encrypt all data in transit.
Injection	Untrusted data accepted by an application forces execution of commands. The most common attacks include SQL injections and cross-site scripting (XSS) attacks	Enforce the use of SQL control statements in queries to limit the number of records returned. Incorporate the use of Content Security Policies Set HTTP Security headers as per OWASP recommendations
Insecure Design	Missing or ineffective control design, flaws in design or architecture.	Determine exposure and protection needs (exposed to internet?) Segregate tier/network layers Limit resource consumption by user or services Work with testers to compile use and misuse cases for every tier
Security Misconfiguration	Security controls are not secured or not configured correctly	Benchmark system security (i.e., CIS Benchmarks, Oracle DBSAT) Create repeatable hardening process Audit for unnecessary ports, services, pages, accounts, user privileges Disable / uninstall unnecessary features and frameworks Audit default accounts for being enabled and passwords unchanged Check if latest security features are enabled and configured correctly Verify correct security settings at all tier levels and frameworks Continuously review all security notes, updates, vulnerabilities, and patches for changes Watch for error messages / logging containing sensitive information Perform penetration, vulnerability, and dynamic analysis security scans Automate process to locate configuration flaws Restrict administrative access
Vulnerable and Outdated Components	Components including underlying dependencies have known vulnerabilities, are unsupported or out of date.	Know and continuously inventory versions of all components involved in technology stack including nested dependencies Stay aware of new vulnerabilities in components by following CVE or the National Vulnerability Database (NVD) Remove unused files, components, features Where possible keep software i.e., OS, web/application server, DBMS, APIs, libraries up to date Where updates are not possible remediate vulnerabilities If remediations unavailable consider virtual patching (Waratek, TrendMicro, McAfee) Scan for vulnerabilities regularly Secure components (see Security Misconfiguration)

Title	Description	Core Actions to Reduce Risk
Identification and Authentication Failures	User's identity, authentication, and session is not confirmed and maintained properly.	Champion multi-factor authentication for applications Do not use default credentials for deploying application Do not use default admin-level users for performing work Enforce strict standards for passwords using security guidelines like National Institute of Standards and Technology (NIST) 800-63b Implement weak password checks Monitor and address repeated login failures
Software and Data Integrity Failures	Updates to software, libraries, modules, configurations, or data from unverified sources.	Use mechanisms like digital signatures to verify software or data has not been altered. Verify libraries and dependencies are using vetted trusted repositories
Security Logging and Monitoring Failures	Insufficient logging, detection, monitoring, and active responses.	Review logging setup to assure it includes failed attempts, denied access, input validation failures but avoids sensitive data Assure logged information is sufficient to identify attackers Ensure that logs are formatted to be consumable by other tools Protect logs as highly sensitive – implement audit trail with integrity controls Integrate with SIEMs and other dashboards, monitoring, alerting tools
Server-Side Request Forgery	Web application fetches remote resource without validating the user-supplied URL.	Disable HTTP redirections Enforce URL schema, port, and destination with a positive allow list Utilize VPNs as appropriate

Zero-Day Action Plan Foundations

The title of this paper is “The Zero-Day, No-Patch Scramble”, the key is though you don’t want to have to scramble. The announcement of high-risk vulnerabilities will probably happen at the most inconvenient time like right before a weekend or holiday. They often have a very high media presence that drive company managers to put enormous pressure on their tech teams to determine what response is needed from their company. Creating a vulnerability response plan before hand is critical to acting quickly and making decisions on what mitigations, if any, are needed.

1. Develop a thorough understanding of your company’s IT infrastructure:
Know the types of systems that are in place, the function of the various components, and get general knowledge of the architecture. This knowledge will help you to understand if your system is even vulnerable. If there is a possibility that it is vulnerable it will help guide you in where remediation might take place to lower the ability for an attacker to use the vulnerability.

2. Create a response team list with contact information:
Time should be spent on analyzing and finding remediations not scrambling to find the right resources to contact. The list should include contacts in each system/business area (hardware, network, applications, DBMS, testing, customer care…). You should also put together a distribution list of individuals that should be kept in the know about the vulnerability and the progress being made to address it.

3. Create quick response guides:
There should be Vulnerability Response Guides that clearly outline responsibilities and tasks to accomplish. Some possible response guides would include monitoring (application, network, perimeter, connections, dbms), communication (to management, users, general company, public) and response team (assembly, communication methods)

4. Have a record of reliable resources that can be used to monitor updates to the vulnerability.
The information that is provided by CVE and NVD can be delayed and often incomplete. Generally, they will not provide example exploits and remediations are often slow in appearing. Maintain a watch on resources such as TechRepublic, ZDNet, CNET, and Network News sites. Sometimes the individual that discovered the vulnerability will provide additional information on the vulnerability and possible remediations. Know how your vendors communicate information on vulnerabilities and have their support sites bookmarked.

Conclusion

The zero-day problem is not going away. Consider just these two factors.

THE EVER-GROWING USE OF OPEN-SOURCE CODE.

If an attacker can find a popular open-source software like log4j. A single software vulnerability provides multiple attack vectors. It cannot be patched by a single software vendor. Each vendor will have to provide a patch for every program that contains the code.

COMPANIES RELYING MORE AND MORE ON CLOUD VENDORS.

This provides incentive to attackers to find vulnerabilities to compromise those systems. If they can breach a single system, it could possibly provide them with multiple high-value targets.

The best defense against zero-day vulnerabilities is a Defense in Depth approach. Patching can be applied once it is released by a vendor, but that can take months if not years to occur. Organizations need to take control of their security. They need to adhere to best practices. The need to consider implementing technologies like Application Firewalls, ... They need to continue to train their staff on new security

For more information, please visit our website at spinnakersupport.com

About Spinnaker Support

Today's leaders are navigating an increasingly uncertain and ever-changing world. They can't be held back by restrictive, ineffective, or complicated software systems as they move their organizations forward. Spinnaker optimizes software ecosystems through services designed for sustainable transformation, maximizing software investments and freeing up the capital and resources leaders need to navigate the future with certainty.

APPENDIX

API Security IO. (retrieved 2022, April 17). CHEAT SHEET OWASP API Security Top 10.
https://apisecurity.io/encyclopedia/content/owasp-api-security-top-10-cheat-sheet-a4.pdf

Mitre. (retrieved 2022, April 18). CVE > CWE Mapping Guidance.
https://cwe.mitre.org/documents/cwe_usage/guidance.html

Sadowwski J. (2022, April 21). Zero Tolerance: More Zero-Days Exploited in 2021 Than Ever Before.
https://www.mandiant.com/resources/zero-days-exploited-2021

Stone M. (2022, April 19). The More You Know, The More You Know You Don't Know.
https://googleprojectzero.blogspot.com/

Trend Micro. (2022, April 27). Zero-Day Vulnerability.
https://www.trendmicro.com/vinfo/us/security/definition/zero-day-vulnerability

Map of CVE to Advisory/Alert (2024, September).
https://www.oracle.com/security-alerts/public-vuln-to-advisory-mapping.html