Contrast Security
Log4j and the Current State of Code Security

Log4j began making global headlines on December 9th, 2021. Users of the popular video game Minecraft found they could gain access to unauthorized information by pasting special code into a chat message. The root cause, it soon became clear, was a popular open-source programming library called Log4j, which is used by millions of computer programs around the world.

Hackers quickly seized on the vulnerability to gain unauthorized access to major commercial services provided by Apple, Amazon, Cloudflare, Steam, Tesla, Twitter, and Baidu. To illustrate how widespread the attack surface is, former hacker-turned-cybersecurity blogger, Marcus Hutchins, tweeted, "In the case of Minecraft, attackers were able to get remote code execution on Minecraft Servers by simply pasting a short message into the chat box." A warning was issued through the MITRE Common Vulnerabilities and Exposures (CVE) database, and the Apache foundation assigned the problem — now formally known as CVE-2021-44228 — a CVSS score of 10 on a 1-to-10 scale, indicating the highest level of severity.

On Tuesday, January 4th, the FTC announced that it will pursue legal action and fines against companies that fail to protect consumers from Log4j attacks, and other attacks like it in the future. The alert referenced Equifax, the credit reporting agency that failed to address a known Apache Struts flaw in 2017, and was subsequently required to pay $700 million in settlements with the agency and with individual states.

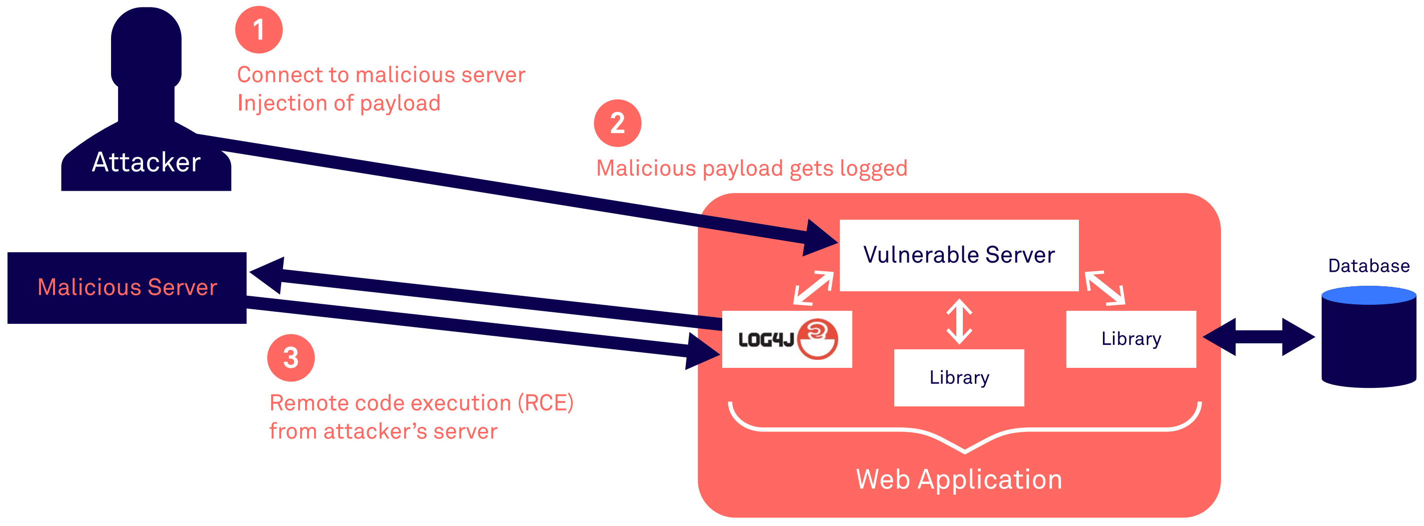
What is Log4j?

Log4j (short for "logging for Java") is a piece of free, pre-written, open-source code that developers use to log user inputs on Java-based applications. That means that any Java-based system that logs even a user name is susceptible to attack.

Potential hackers can use the vulnerability to execute their own arbitrary computer code on a company's servers, allowing them to gain access to confidential or privileged information, or establish a foothold in an otherwise protected environment. A single web request can be enough to initiate a Log4j hack. Often the request can occur even before a user is authenticated. The implications are massive.

- Some 8.2 million programmers around the world currently use Java (ZDNet)
- 58% of Java apps contain vulnerable versions of Log4j
- While multiple patches have been so far released (with more likely coming in the future) Log4j is so common that most organizations struggle to identify all the instances in their environments

Add to this the fact that implementing patches across multiple systems is both time and labor-intensive, and the result is a vulnerability even worse than the 2017 Equifax hack, which cost that company over $425 million in fines alone.


1
Connect to malicious server
Injection of payload
Attacker
2
Malicious payload gets logged
Vulnerable Server
Malicious Server
LOG4J
Library
Library
Database
3
Remote code execution (RCE) from attacker's server
Web Application

Large numbers of hacks have so far been documented, however little has been reported about damage caused by breaches, which we will be hearing about in the coming months. Perhaps more alarming is the fact that, despite extensive media coverage, many vulnerable companies still don't know about the problem, which means that attacks are almost certainly still ongoing.

What Log4j has revealed

The present day code security world is dominated by static code analysis tools, SCA tooling tied to code repositories, and web application firewalls (WAF). How did these solutions fare against Log4j? The answer is: not well. Overall, customers running separate security tooling (like SAST, SCA, and WAF) found that they had no single source of truth for where Log4j was deployed, nor any assurance that they were protected against it. Many were forced to use custom scripts to identify which applications were running Log4j.

Scanning tools like SAST deployed across multiple teams forced businesses to try to verify a deluge of results with no single source of truth. Even more crucially, scanning tools were only able to analyze custom code in a silo — without regard to open source components, nor the manner in which those components were used. Static tools gave no hint of the problem, and even if they had, SAST tools are so noisy, no one would have noticed.

"Before Contrast, we were using different static application security testing (SAST), software composition analysis (SCA), and WAF solutions. We found ourselves overwhelmed with tool soup. Contrast consolidated everything into a single platform with accurate and fast results that we were not aware of before. The platform features were more mature than their competition and made it easier to manage, integrate and consume results. We were also impressed with the seamless onboarding process. Contrast is protecting us against the recently disclosed Log4j vulnerability without having to patch or update our servers."

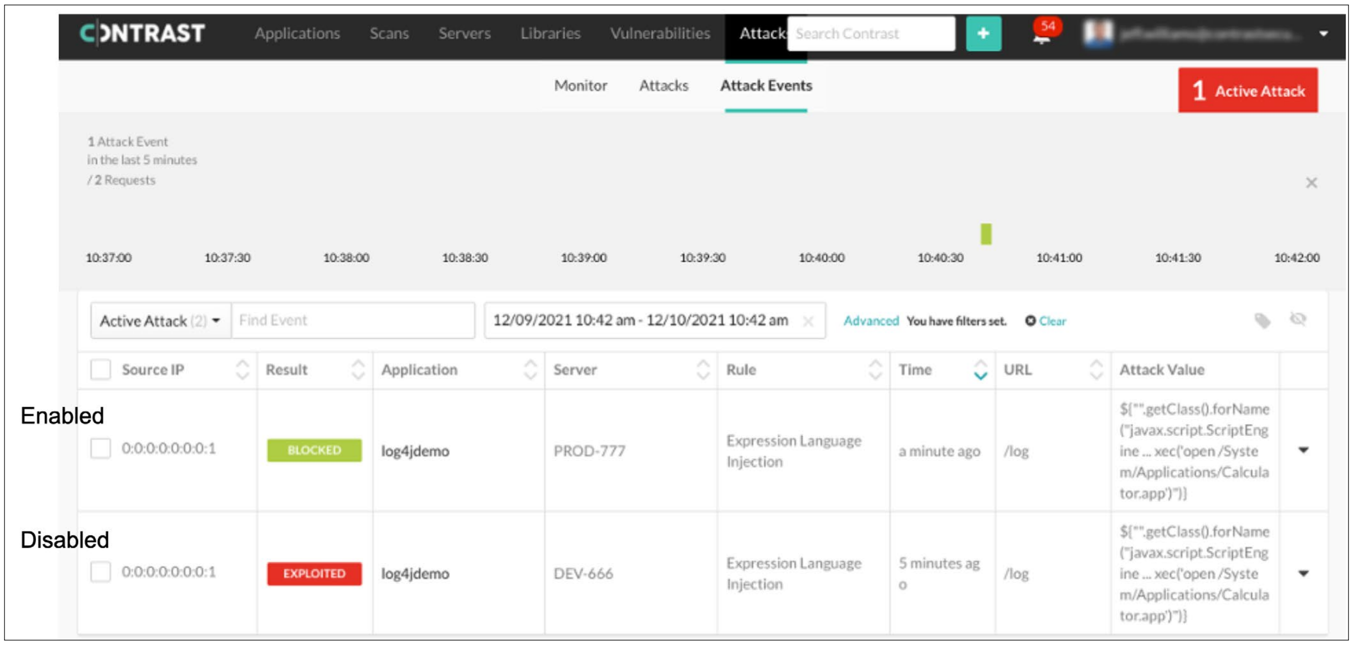
–Brian Vlootman. BackBase CISO

SCA plugged into code repositories did no better, as it severely over-reported the issue. SCA tooling only looks at what's bundled, not what is being used or the manner in which it is being used. Many of the Log4j dependencies turned up by SCA are only present for testing or transitive purposes, and aren't packaged in the app itself. More than that, even some of the dependencies that are packaged into the app aren't actually used (Contrast data shows that while 58% of Java applications package a vulnerable version of Log4j, only 37% actually use it). SCA, however, has no way of discriminating between used and unused dependencies, which left businesses in the position of having to chase them all.

WAFs simply failed to protect running applications that had vulnerable versions of Log4j running. The reason, because the data in those applications doesn't just come from web input, and so isn't visible with signatures. In addition, recipes to evade most major commercial WAFs were disclosed within days of the exploit being uncovered. Companies found that their attempts to issue updated rule sets were quickly defeated.

Contrast: Built to handle vulnerabilities like Log4j

The Contrast Secure Code Platform has been specifically engineered for security vulnerabilities like Log4j. Contrast's embedded approach allowed our customers, immediately and at scale, to identify which apps had versions of Log4j in them; detect which apps were vulnerable, and; immediately stop attacks against those apps (without patching).



12/09/2021 10:42 am - 12/10/2021 10:42 am

You have filters set.

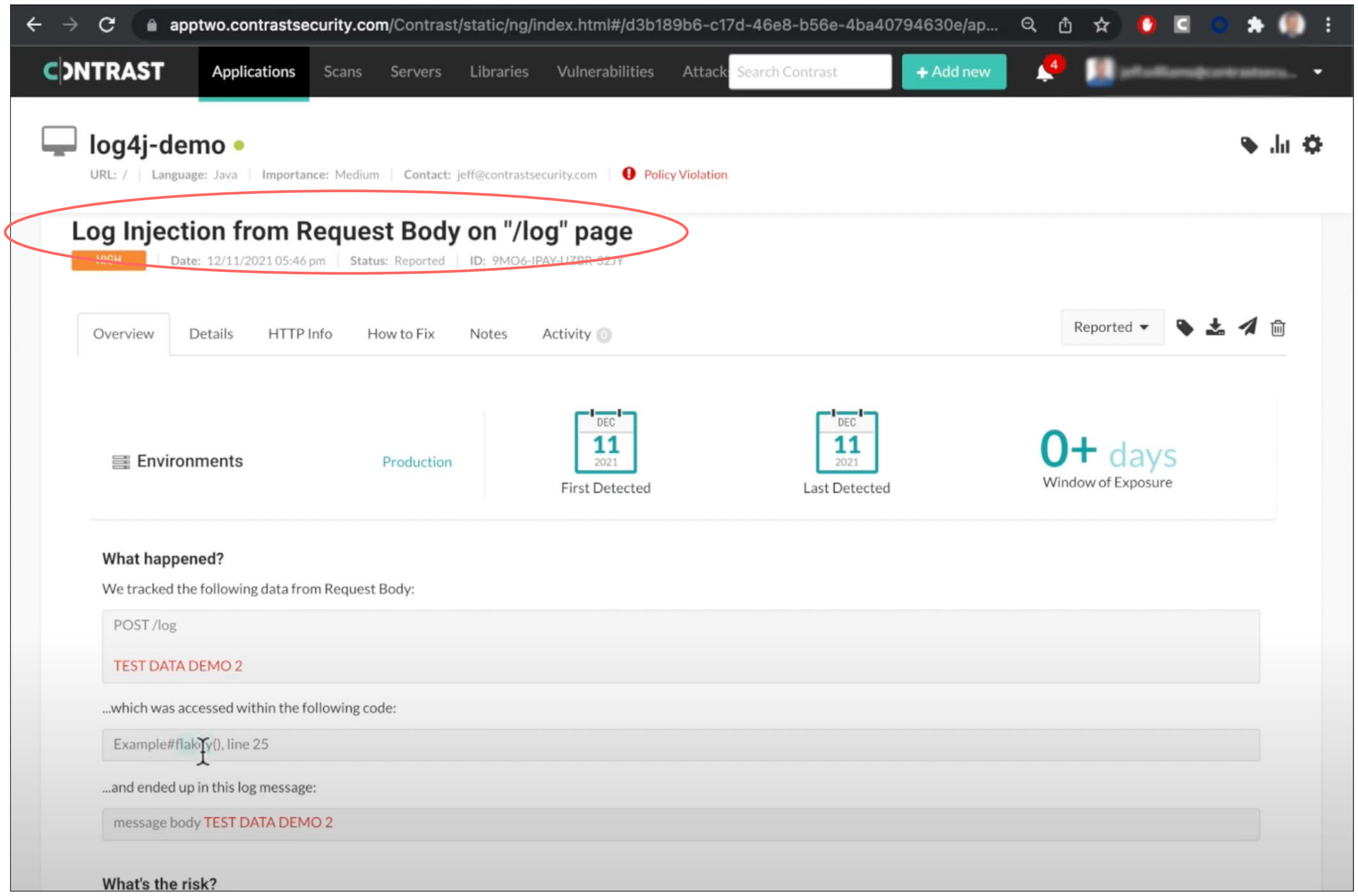
	Source IP	Result	Application	Server	Rule	Time	URL	Attack Value
Enabled	0:0:0:0:0:0:0:1	BLOCKED	log4jdemo	PROD-777	Expression Language Injection	a minute ago	/log	${"".getClass().forName("javax.script.ScriptEngine ... xec('open /System/Applications/Calculator.app')")}
Disabled	0:0:0:0:0:0:0:1	EXPLOITED	log4jdemo	DEV-666	Expression Language Injection	5 minutes ago	/log	${"".getClass().forName("javax.script.ScriptEngine ... xec('open /System/Applications/Calculator.app')")}

Figure 1: The Log4j attack (classified as "Expression Language Injection") with and without Contrast Protect enabled.

"Just received the official notification that Contrast is protecting against this. GREAT WORK! This highlights a really big win and has direct resource impacts for folks internally using it. They actually get to have the weekend off vs. other teams scrambling to fix."

–Distinguished advisor from a major medical device manufacturer

Contrast Assess saw the problem from the start. It has always reported log injection and deserialization vulnerabilities associated with Log4j, whenever untrusted data is logged without proper validation or escaping. Any customer who fixes log injection vulnerabilities reported by Contrast Assess is safe from Log4j.



log4j-demo

URL: / | Language: Java | Importance: Medium | Contact: jeff@contrastsecurity.com | Policy Violation

Log Injection from Request Body on "/log" page

HIGH | Date: 12/11/2021 05:46 pm | Status: Reported | ID: 9MO6-IPAY-LIZPR-82JY

Environments Production

DEC 11 2021 First Detected

DEC 11 2021 Last Detected

0+ days Window of Exposure

What happened?

We tracked the following data from Request Body:

```
POST /log

TEST DATA DEMO 2
```

...which was accessed within the following code:

```
Example#flaky(), line 25
```

...and ended up in this log message:

```
message body TEST DATA DEMO 2
```

What's the risk?

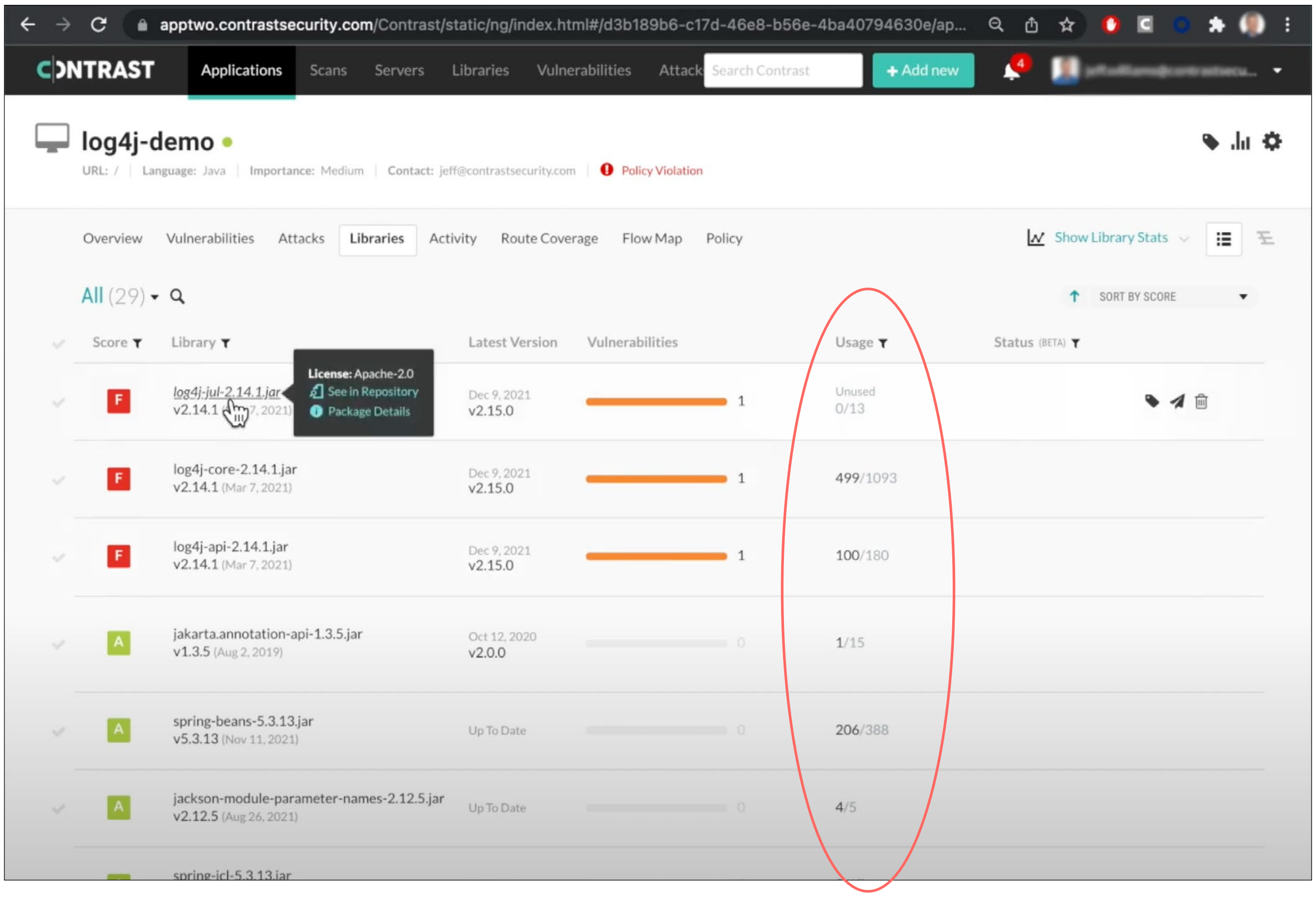
Figure 2: Contrast Assess intrinsically identifies log injection vulnerabilities associated with the Log4j exploit.

Contrast SCA made identifying which apps were actually at risk as easy as a Google search, even cloud-native apps. Instead of reporting libraries that aren't actually used by the application in production, Contrast SCA provides exact details on whether the library is actually invoked at runtime to help defenders implement the most urgent fixes.

"Contrast provided useful reporting on log4j CVE for apps onboarded to Contrast. We also used information from your blog. After this week I can categorically say there is a LOT more interest in Contrast."

–Elsevier BISO. Paul Lewis

For more information, please visit https://www.contrastsecurity.com/log4j2.


apptwo.contrastsecurity.com/Contrast/static/ng/index.html#/d3b189b6-c17d-46e8-b56e-4ba40794630e/ap...

log4j-demo

URL: / | Language: Java | Importance: Medium | Contact: jeff@contrastsecurity.com | Policy Violation

Score	Library	Latest Version	Vulnerabilities	Usage	Status (BETA)
F	log4j-jul-2.14.1.jar v2.14.1 7, 2021)	Dec 9, 2021 v2.15.0	1	Unused 0/13	
F	log4j-core-2.14.1.jar v2.14.1 (Mar 7, 2021)	Dec 9, 2021 v2.15.0	1	499/1093	
F	log4j-api-2.14.1.jar v2.14.1 (Mar 7, 2021)	Dec 9, 2021 v2.15.0	1	100/180	
A	jakarta.annotation-api-1.3.5.jar v1.3.5 (Aug 2, 2019)	Oct 12, 2020 v2.0.0	0	1/15	
A	spring-beans-5.3.13.jar v5.3.13 (Nov 11, 2021)	Up To Date	0	206/388	
A	jackson-module-parameter-names-2.12.5.jar v2.12.5 (Aug 26, 2021)	Up To Date	0	4/5	
	spring-jcl-5.3.13.jar				

Figure 3: Contrast SCA highlights if Log4j is actually called at runtime by highlighting class usage.

Conclusions

First it's important to recognize that Log4j is only the most recent major security breach. The truth is that equally serious events happen every day (if not always on a global scale). Businesses that are serious about security need to be on constant alert. Real-time protection is, and always will be, the key to defending against zero-day events like Log4j.

"This kind of thing happened before and will happen again," said Steve Wilson, Chief Product Officer at Contrast Security. "In 2017, Equifax announced a data breach that exposed personal, confidential information and was very similar to this situation in many ways. However Log4j is far more common than Apache Struts was at the time of the 2017 incident. This means that the exposure is far, far broader. The best strategy is to use Runtime Protection, like Contrast Protect, to defend immediately without patching."

"We were able to analyze whether our own built software would be vulnerable to the Log4j zero-day, using the Contrast Secure Code Platform, and got the answer within 30 seconds by just looking at the Libraries menu! How fast is that!"

–Sandor Incze. CISO at CM.com

The second thing to remember is that the Log4j crisis is far from over. Organizations are just beginning to understand their exposure to Log4j. It's an attack that's easy-to-execute at scale, that evades firewalls, and is extremely damaging once executed. So expect to see a lot of it in the coming weeks and months.

For more information, please visit https://www.contrastsecurity.com/log4j2.

Third, all commonly used open-source code that lives in your supply chain is a potential point of failure.

Looking at the positives of the Log4j experience, communities and agencies were able to respond with answers, data and tools much faster than ever before. We can only hope to see more such responsiveness in the future.

Overall, Log4j demonstrates the distinct advantage of security platforms like Contrast, which allow you to respond instantly to a zero-day like Log4j without waiting for a patch, and let you future-proof your stack against the next major vulnerabilities.