Mandiant

HOW DNS EXFILTRATION VULNERABILITIES ARE DETECTED

DNS Exfiltration (DNS Exfil) is an attack technique which allows attackers to extract data from a compromised target host by embedding the data in a DNS lookup made to a DNS server controlled by the attacker. This activity is particularly challenging to detect because the extracted data is often segmented and encrypted within what appears to be benign DNS traffic that is very common in day-to-day operations.

Mandiant Advantage Attack Surface Management (MA-ASM) uses this DNS Exfiltration technique when testing what are known as "blind" vulnerabilities. These are vulnerabilities for which no immediate output is returned by the compromised target after the vulnerability is exploited. In other words, there's no immediate indication that an active exploitation successfully occurred. In order to trigger a detection, MA-ASM uses a specifically crafted payload which, when successfully evaluated by the vulnerable target, will result in a DNS lookup containing a unique ID that ties back to the vulnerable target and the respective vulnerability. This approach allows MA-ASM to confirm active exploitation of blind vulnerabilities using the DNS Exfiltration technique.

You can identify these queries in your security logs by the hostname that initiates these requests: `r.dns-exfil.prod.asm.mandiant.com`

The response from the vulnerable system can come at any time because it's often associated with an offline process. In cases like the Log4j bug, for example, responses may come back days or even weeks later.

If the vulnerable system is deployed behind a firewall or load balancer, the IP address associated with the unique ID in the query response may be assigned with the the firewall or load balancer rather than the vulnerable system that triggered the callback. In these cases, further investigation of backend systems would be required to isolate the vulnerability.

Identifying DNS Exfiltration Issues

Issues associated with DNS Exfiltration can be identified by the following string appearing in the Proof of an Issue:

`The following resolver IP address [IP address] invoked a DNS Lookup with the following data...`

For more information about interpreting Proof data, see **Reviewing Issues (https://docs.mandiant.com/home/asm-issues)**.

The current list of Issues in MA-ASM that are associated with DNS Exfiltration at time of publication can be found below.

The list of DNS Exfiltration Issues in MA-ASM will continue to grow as new vulnerabilities are discovered. To confirm if an Issue is related to DNS Exfiltration, look for the string above in the Issue's Proof.

Apache APISIX - Remote Code Execution (CVE-2022-24112)
Apache HTTP Server Side Request Forgery (CVE-2021-40438)
Apache OFBiz - Remote Code Execution (CVE-2021-26295)
Apache OFBiz - Log4Shell Remote Code Execution
Apache Solr - Log4Shell Remote Code Execution
Atlassian Confluence Server - Arbitrary Code Execution (CVE-2022-26134)
CentOS Web Panel - Remote Code Execution (CVE-2022-44877)
Cisco HyperFlex - Unauthenticated Remote Code Execution (CVE-2021-1497)
Cisco HyperFlex - Unauthenticated Remote Code Execution(CVE-2021-1498)
Cisco vManage - Log4Shell Remote Code Execution
Gitlab - Unauthenticated Remote Code Execution (CVE-2021-22205)
GoAnywhere - Log4Shell Remote Code Execution
Graylog - Log4Shell Remote Code Execution
Jamf Pro - Log4Shell Remote Code Execution
Log4j aka Log4Shell Unauthenticated Remote Code Execution (CVE-2021-44228)
Manage Engine OpManager - Unauthenticated Remote Code Execution (CVE-2020-28653)
Metabase - Log4Shell Remote Code Execution
Microsoft Exchange Server - Server-Side Request Forgery (CVE-2022-41040)
MobileIron - Log4Shell Remote Code Execution
OpenNMS - Log4Shell Remote Code Execution
Oracle Weblogic - Unauthenticated Remote Code Execution (CVE-2019-2725)
Oracle WebLogic - Remote Code Execution (CVE-2020-14883)
Rundeck - Log4Shell Remote Code Execution
Sitecore Experience Platform Remote Code Execution (CVE-2021-42237)
Spring Boot - Log4Shell Remote Code Execution
Spring Cloud Function - Remote Code Execution (CVE-2022-22963)
VMWare HCX - Log4Shell Remote Code Execution
VMware Horizon - Log4Shell Remote Code Execution
VMWare NSX - Log4Shell Remote Code Execution
VMware Workspace ONE - Authentication Bypass (CVE-2022-22972)
Zyxel Firewall - Remote Code Execution (CVE-2022-30525)
Oracle Weblogic - Remote Code Execution (CVE-2020-14882)

Log4Shell Proof Interpretation

Every Log4Shell issue proof contains a data value. This data value is information that was sent to MA-ASM from the vulnerable server when the payload was evaluated. The data value follows the following format:

`__l0c-_____`

The value before `l0c-` in this string is a key that corresponds to the source of the HTTP request which contained the payload that resulted in the vulnerable server making a lookup. The value after `l0c-` is the Java version which was exfiltrated from the vulnerable server.

To avoid false positives, if no Java version is exfiltrated, an issue will not be created.

The key at the beginning of the data value string corresponds to a header in this table:

Value	Header
`ua`	'User-Agent'
`re`	Referer

Value	Header
co	Cookie
vi	Via
or	Origin
xf	'X-Forwarded-For'
au	Authorization
ct	'Content-Type'
cc	'Cache-Control'
pr	Pragma
xr	'X-Requested-With'

Take the following data value for example: `ual0c-1.8.0_181`

Here `ua`, at the beginning of the string and preceding `l0c-`, represents 'User-Agent', according to the table above. The Java version is `1.8.0_181`.