Corelight Bright Ideas Blog

Detecting CVE-2021-42292

November 10, 2021 by **Corelight Labs Team**

On its surface, **CVE-2021-42292** doesn't look like the kind of vulnerability that a network-based tool can find reliably. Marked by Microsoft as a local file format vulnerability, security veterans would expect that between encryption and encoding, there would be a million different ways to evade **network detection** with a weaponized exploit.

Digging a bit deeper, however, this specific vulnerability presents an opportunity to be detected through a behavior you might not even realize is present in Excel: its ability to download other files directly, without user interaction. Specifically, this exploit relies on Excel fetching a second Excel file, which then contains the malicious code.

So how do you detect that across the network? It boils down to monitoring User-Agents and MIME types.

Detection Logic Walkthrough

The first thing the **Corelight Labs** team looked at while investigating this issue was whether Excel even has a sufficiently identifiable User-Agent string - the HTTP field used to identify the software making the request. While there are **many variants** depending on specific versions, the theme of either the combination of "Microsoft Office" plus "Excel," or "Mozilla" plus "ms-office," emerged quickly - and was validated with plenty of real-world examples when we began digging in through our **Polaris research** systems.

MIME types - declarations by the web server about the type of content coming down - are thankfully a much more standardized affair.

RECENT POSTS

Streamlining security investigations with real-time enrichment of Corelight Open NDR and SentinelOne Singularity

Feed me!

Want better network visibility? Don't just go with the (net)flow

Stronger Security with Corelight and Mandiant Managed Defense

Corelight Open NDR Achieves VMware Ready for Telco Cloud Infrastructure Certification

The important question, once we had established that it was possible to detect Excel downloading another Excel file, is whether that happens frequently enough in normal operations that exploits would be lost in the noise. Going back through the past 30 days' worth of data across the Polaris systems, we looked at all of the times we found Excel downloading files, which happened thousands of times per day at some of our large sites, typically with plain text, HTML, or images. The good news was that Excel downloading Excel was exceptionally rare: it happened at most in 0.03% of such downloads, with some sites showing no sign of it at all.

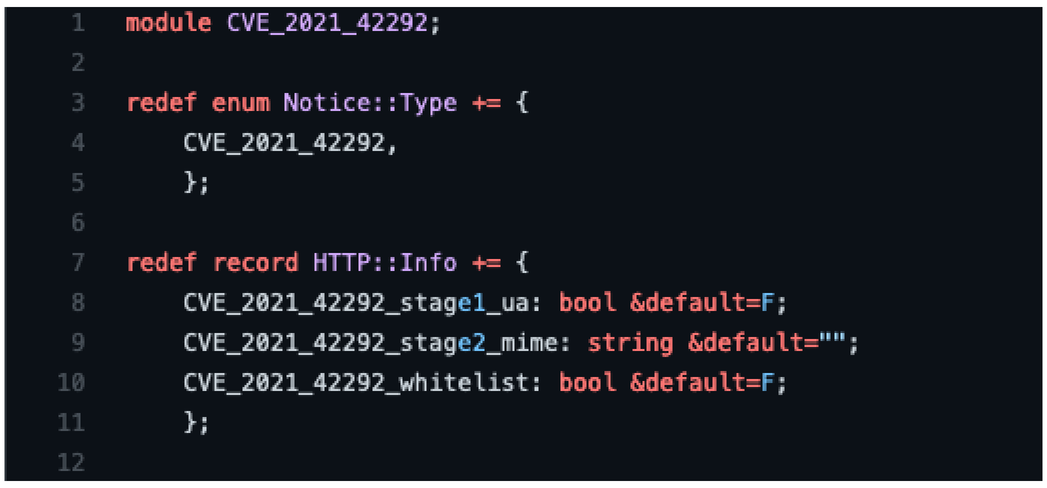
One frequent false positive that we did see during our investigation was **SharePoint**. Since that application helpfully declares itself with a header of "MICROSOFTSHAREPOINTTEAMSERVICES," however, it's simple to exclude those cases. Advanced SOCs concerned about this vector could easily remove this check, however, and rely on post-processing to filter out legitimate usage of that tool.

Zeek® Script Walkthrough

We have published **Zeek code to detect** the potential exploitation of this vulnerability on Github.

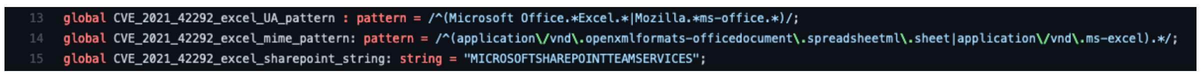
The **main source code file** can be understood as follows:

- Line 1 declares a new module for detecting CVE-2021-42292.
- Lines 3-5 declare the new Notice type that we'll be using when possible exploits are detected.
- Lines 7-11 add new fields to the HTTP records to track the state of the detection logic for each HTTP session.



```
 1  module CVE_2021_42292;
 2
 3  redef enum Notice::Type += {
 4      CVE_2021_42292,
 5      };
 6
 7  redef record HTTP::Info += {
 8      CVE_2021_42292_stage1_ua: bool &default=F;
 9      CVE_2021_42292_stage2_mime: string &default="";
10      CVE_2021_42292_whitelist: bool &default=F;
11      };
12
```

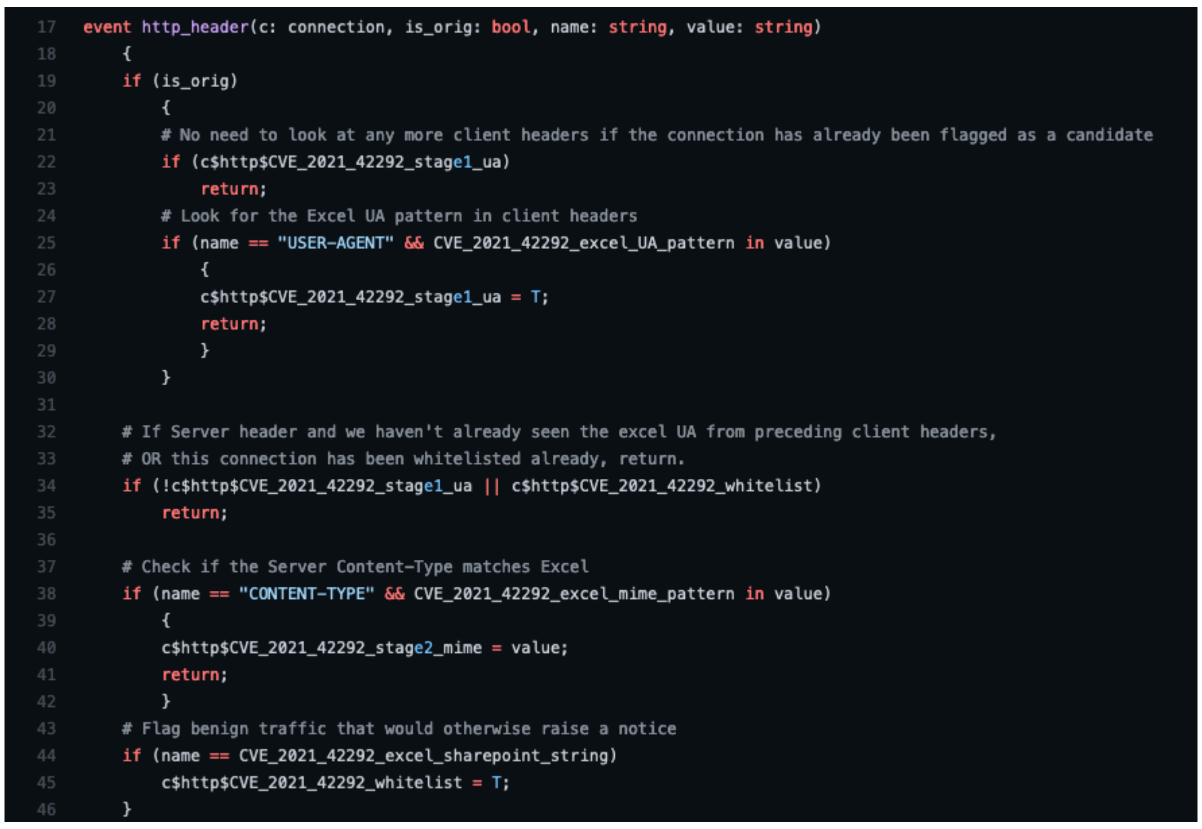
- Lines 13-15 declare the patterns and strings that are used by the detection logic. Declaring them in this way makes it easy to redeclare them without changing this module's code if exploits of CVE-2021-42292 evolve in the future.



```
13  global CVE_2021_42292_excel_UA_pattern : pattern = /^(Microsoft Office.*Excel.*|Mozilla.*ms-office.*)/;
14  global CVE_2021_42292_excel_mime_pattern: pattern = /^(application\/vnd\.openxmlformats-officedocument\.spreadsheetml\.sheet|application\/vnd\.ms-excel).*/;
15  global CVE_2021_42292_excel_sharepoint_string: string = "MICROSOFTSHAREPOINTTEAMSERVICES";
```

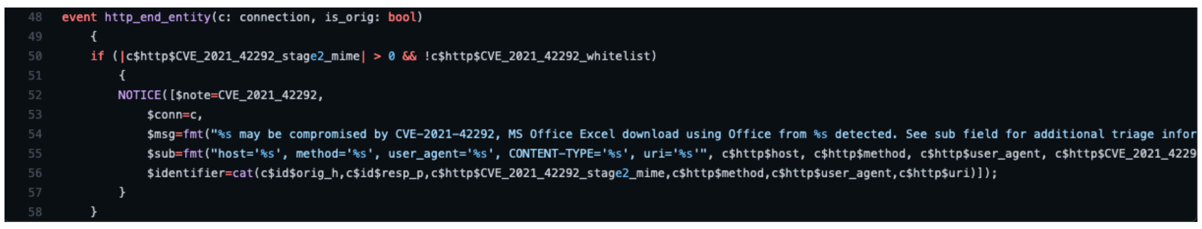
- Lines 17-46 are the heart of the detection logic. For each HTTP

- Lines 32-35 limit this search to only those HTTP sessions that were already flagged as having a client User-Agent associated with Excel.
- Lines 37-42 search for HTTP content that appears to be Excel files.
- Lines 43-45 will cause the detection logic to ignore HTTP sessions being served by Microsoft SharePoint to avoid false positive notices.



```
17  event http_header(c: connection, is_orig: bool, name: string, value: string)
18      {
19      if (is_orig)
20          {
21          # No need to look at any more client headers if the connection has already been flagged as a candidate
22          if (c$http$CVE_2021_42292_stage1_ua)
23              return;
24          # Look for the Excel UA pattern in client headers
25          if (name == "USER-AGENT" && CVE_2021_42292_excel_UA_pattern in value)
26              {
27              c$http$CVE_2021_42292_stage1_ua = T;
28              return;
29              }
30          }
31  
32      # If Server header and we haven't already seen the excel UA from preceding client headers,
33      # OR this connection has been whitelisted already, return.
34      if (!c$http$CVE_2021_42292_stage1_ua || c$http$CVE_2021_42292_whitelist)
35          return;
36  
37      # Check if the Server Content-Type matches Excel
38      if (name == "CONTENT-TYPE" && CVE_2021_42292_excel_mime_pattern in value)
39          {
40          c$http$CVE_2021_42292_stage2_mime = value;
41          return;
42          }
43      # Flag benign traffic that would otherwise raise a notice
44      if (name == CVE_2021_42292_excel_sharepoint_string)
45          c$http$CVE_2021_42292_whitelist = T;
46      }
```

Finally, when every HTTP session has ended, lines 48-58 check the flags that we added to the associated HTTP::Info record to see if our detection flags have been set, and a Notice is generated if a possible exploit of CVE-2021-42292 has been detected.



```
48  event http_end_entity(c: connection, is_orig: bool)
49      {
50      if (|c$http$CVE_2021_42292_stage2_mime| > 0 && !c$http$CVE_2021_42292_whitelist)
51          {
52          NOTICE([$note=CVE_2021_42292,
53              $conn=c,
54              $msg=fmt("%s may be compromised by CVE-2021-42292, MS Office Excel download using Office from %s detected. See sub field for additional triage infor
55              $sub=fmt("host='%s', method='%s', user_agent='%s', CONTENT-TYPE='%s', uri='%s'", c$http$host, c$http$method, c$http$user_agent, c$http$CVE_2021_4229
56              $identifier=cat(c$id$orig_h,c$id$resp_p,c$http$CVE_2021_42292_stage2_mime,c$http$method,c$http$user_agent,c$http$uri)]);
57          }
58      }
```

Suricata Signature Walkthrough

In addition to the above **Zeek** script, we're releasing equivalent **Suricata rules** in the same Github repository. This allows users to be flexible about how they deploy this detection, both for Corelight customers and open source users.

Since Suricata is not session-oriented like Zeek, but is instead packet stream oriented, we need signatures for both the request and response packets in order to create detection. Since Suricata is also fastest when it has unique strings to feed its fast pattern matching engine, we break up the two possible User-Agent strings into a pair of rules:

```
alert http $HOME_NET any -> any any (msg:"CORELIGHT MS
Excel User-Agent flowbit set 1";
flow:established,to_server; http.user_agent;
```

```
alert http $HOME_NET any -> any any (msg:"CORELIGHT MS
Excel User-Agent flowbit set 2";
flow:established,to_server; http.user_agent;
content:"Mozilla"; nocase; content:"ms-office"; nocase;
distance:0; pcre:"/^Mozilla.*?ms-office/Vi";
flowbits:set,CORELIGHT.excel.ua; flowbits:noalert;
classtype:misc-activity; sid:3000012; rev:1;
metadata:created_at_2021_11_10, udpated_at 2021_11_10;)
```

The key piece to note here is the flowbit clauses. By setting a bit here - and then telling Suricata to not generate an alert for these rules - they become a prerequisite for the alert that we can check in our following rules:

```
alert http any any -> $HOME_NET any (msg:"CORELIGHT MS
Excel downloading Excel - potential exploit (CVE-2021-
42292)"; flow:established,to_client;
flowbits:isset,CORELIGHT.excel.ua; http.content_type;
content:"application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"; nocase; http.header;
content:!"MICROSOFTSHAREPOINTTEAMSERVICES"; nocase;
reference:cve,2021-42292; classtype:attempted-user;
sid:3000013; rev:1; metadata:created_at_2021_11_10,
udpated_at 2021_11_10;)
```

```
alert http any any -> $HOME_NET any (msg:"CORELIGHT MS
Excel downloading Excel - potential exploit (CVE-2021-
42292)"; flow:established,to_client;
flowbits:isset,CORELIGHT.excel.ua; http.content_type;
content:"application/vnd.ms-excel"; nocase; http.header;
content:!"MICROSOFTSHAREPOINTTEAMSERVICES"; nocase;
flowbits:unset,CORELIGHT.excel.ua; reference:cve,2021-
42292; classtype:attempted-user; sid:3000014; rev:1;
metadata:created_at_2021_11_10, udpated_at 2021_11_10;)
```

These rules check for the previously set flowbits, and then validate that the appropriate MIME type has come down; the negated content match, denoted by the "!" in front of the string, rules out SharePoint services per the false positive detailed above.

Conclusion

Detecting the exploit this way is not going to be completely free of false positives, but with all of the context of the transaction provided by Zeek, weeding out the noise should be straightforward. An Excel

By Corelight Labs Team

Tagged With: **Zeek**, **network detection response**, **network security**, **NDR**, **Suricata**, **GitHub**, **microsoft**, **featured**, **CVE-2021-42292**, **SharePoint**, **Excel**, **mozilla**, **polaris**, **Microsoft Patch Tuesday**, **MIME types**