Group Members: Seongwon Burm, Yuchan Kim, Cristobal Newman, Jeong Yong Yang

Selected CVE: CVE-2022-42889

Link to CVE Database Entry:

https://nvd.nist.gov/vuln/detail/CVE-2022-42889#vulnCurrentDescriptionTitle

CVE-2022-42889 Report

1. Short description/introduction.

https://commons.apache.org/proper/commons-text/

https://nvd.nist.gov/vuln/detail/CVE-2022-42889#vulnCurrentDescriptionTitle

CVE-2022-42889, also referred to as ‘text4shell’, is a vulnerability within the Apache Commons Text. According to the website of Apache, it is an open-source Java library that focuses on algorithms regarding strings. More specifically, it manipulates strings that go beyond the capacity that Java offers by including functions that expand the text with lookups. Prior to the disclosure of this vulnerability, Apache Commons Text allowed users to insert specific fields (provided by the library) as the value for the “prefix” variable, which stores the location of org.apche.commons.text.lookup.StringLookup, when using the format of the interpolation “${prefix:name}”. Adversaries can execute arbitrary code in remote servers by using one of the “dns”, “script”, or “url” fields (set the “prefix” variable as one of them to launch attacks). Through this attack, the adversary can connect with other servers/websites that use Apache Commons Text and execute arbitrary code within them, which makes this attack very vulnerable.

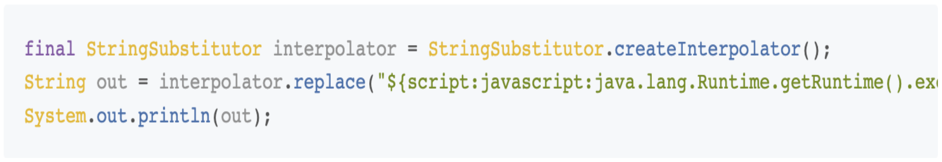
2. Who can launch the attack? Detail the requirements an attacker needs to perform the attack.

https://www.docker.com/blog/security-advisory-cve-2022-42889-text4shell/

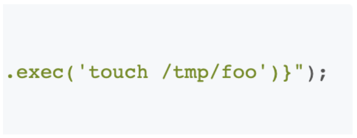
https://www.bugcrowd.com/glossary/remote-code-execution-rce/

https://securitylab.github.com/advisories/GHSL-2022-018_Apache_Commons_Text/

Any adversaries who realize that a program or server uses versions ranging from 1.5 to 1.9 of Apache Commons Text can launch the attack. However, it is important to note that adversaries require the knowledge of Apache Commons Text and its interpolator class of it since this vulnerability occurs from the three fields from it. Furthermore, this vulnerability does not require the connection to WAN or LAN nor physical contact with the targeted device/machine, allowing the attacker to simply execute arbitrary code in remote servers by using terminal commands, Bash scripts (bugcrowd), or other tools that allow a connection to the targeted server. In other words, the adversary can use his/her computer to write the vulnerable code and execute it on the target server.



```
final StringSubstitutor interpolator = StringSubstitutor.createInterpolator();
String out = interpolator.replace("${script:javascript:java.lang.Runtime.getRuntime().ex
System.out.println(out);
```



```
.exec('touch /tmp/foo')}");
```

3. Why does the attack matter? Which industries/systems are impacted?

https://thehackernews.com/2022/10/hackers-started-exploiting-critical.html

https://www.docker.com/blog/security-advisory-cve-2022-42889-text4shell/

https://securityboulevard.com/2022/11/text4shell-cve-2022-42889-vulnerability/

Apache Commons Text is a widely known java library that is used by various companies, which implies that any industries/systems/companies that are using versions ranging from 1.5 to 1.9 of Apache Commons Text can all be impacted by this vulnerability. The three fields listed in the introduction section allow anybody to successfully execute arbitrary code within the targeted server. In fact, the severity of this vulnerability comes from the fact that any adversary can launch the attack no matter the location and the time since it can be done by connecting to the targeted server via online. Furthermore, the numerous types of attacks that adversaries can launch also contribute to the high vulnerability. Attackers can inject malicious code that can trigger a DNS request, call a remote URL, or execute an inline script. These attacks can not only damage the companies but also the users of the industries due to the fact that JavaScript code can be used to steal information from them. According to thehackernew.com, "a successful attempt of this attack would result in the victim site making a DNS query to the attacker-controlled listener domain", which can allow adversaries to perform a myriad of actions on objectives. One example is to open a "reverse shell with the vulnerable application via a specially crafted payload" – providing attackers a foothold from which to launch future attacks.

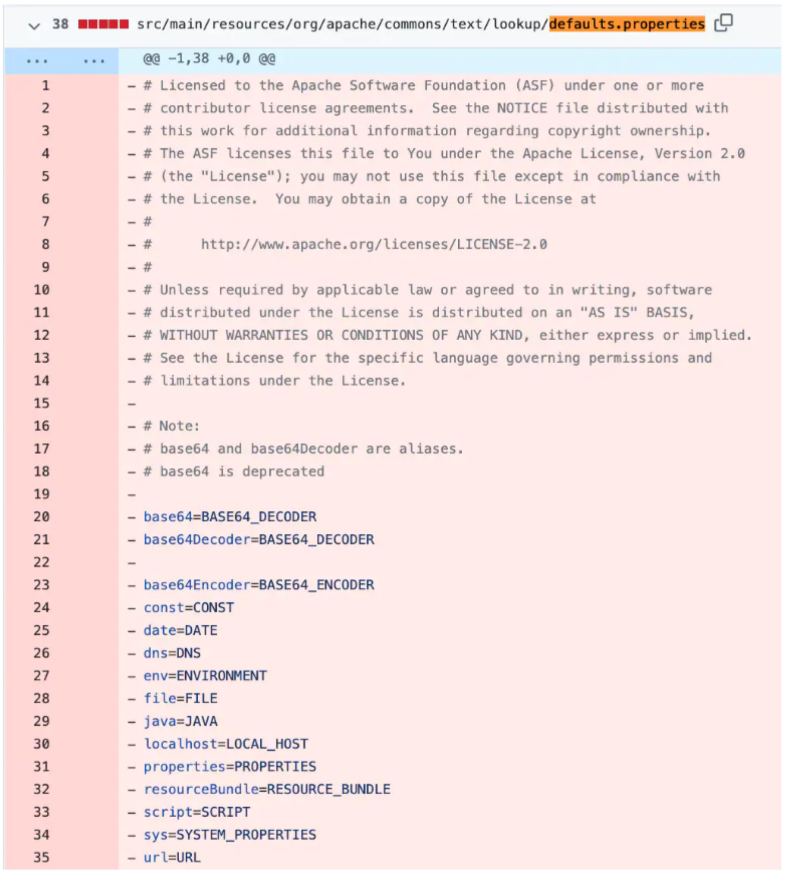
4. What is the attack? How does it work? What damage does this CVE cause? Explain in rigorous details.

Source (including images):
https://www.paloaltonetworks.com/blog/prisma-cloud/analysis_of_cve-2022-42889_text4shell_vulnerability/?utm_source=thenewstack&utm_medium=website

Once the adversary discovers that the target company uses versions ranging from 1.5 to 1.9 of Apache Common Text, he/she can make use of *StringSubstitutor* class in combination with the default Java interpolators, which could pose security risks. The paloalto website describes the vulnerability in detail.

Final StringSubstitutor interpolator = StringSubstitutor.createInterpolator()

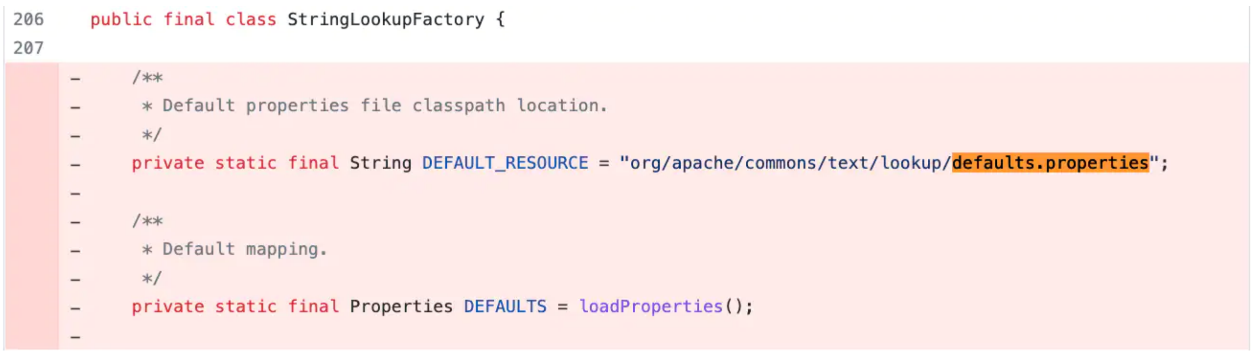
When the adversary declares the following code above, a new instance of the default interpolator containing default lookups will be created, which allows the interpolator to perform string lookups when "${prefix}" variable is specified with one of the lookup keys. The problem arises from the 'script', 'dns', and 'url' lookup keys within the default.properties.



```
38 src/main/resources/org/apache/commons/text/lookup/defaults.properties
@@ -1,38 +0,0 @@
- # Licensed to the Apache Software Foundation (ASF) under one or more
- # contributor license agreements.  See the NOTICE file distributed with
- # this work for additional information regarding copyright ownership.
- # The ASF licenses this file to You under the Apache License, Version 2.0
- # (the "License"); you may not use this file except in compliance with
- # the License.  You may obtain a copy of the License at
- #
- #      http://www.apache.org/licenses/LICENSE-2.0
- #
- # Unless required by applicable law or agreed to in writing, software
- # distributed under the License is distributed on an "AS IS" BASIS,
- # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
- # See the License for the specific language governing permissions and
- # limitations under the License.
-
- # Note:
- # base64 and base64Decoder are aliases.
- # base64 is deprecated
-
- base64=BASE64_DECODER
- base64Decoder=BASE64_DECODER
-
- base64Encoder=BASE64_ENCODER
- const=CONST
- date=DATE
- dns=DNS
- env=ENVIRONMENT
- file=FILE
- java=JAVA
- localhost=LOCAL_HOST
- properties=PROPERTIES
- resourceBundle=RESOURCE_BUNDLE
- script=SCRIPT
- sys=SYSTEM_PROPERTIES
- url=URL
```

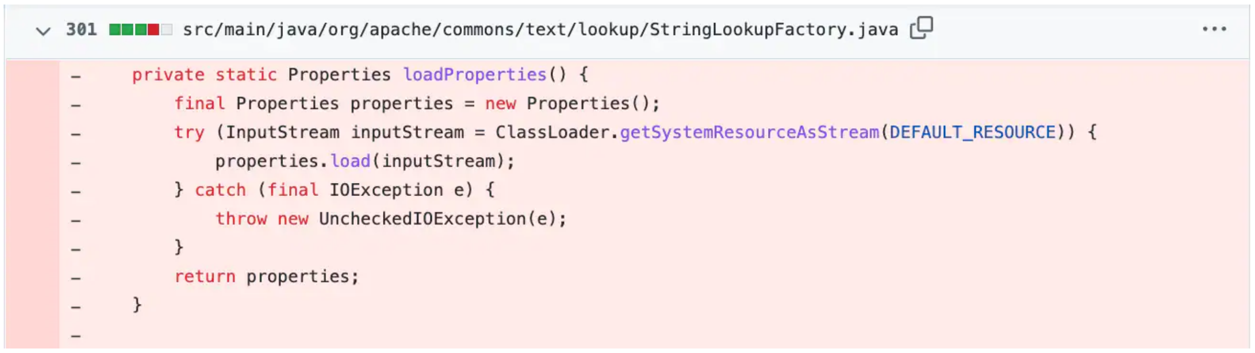
Figure 2. Default properties

The figure above displays the fact that the 'script', 'url', and 'dns' fields are located within default.properties.



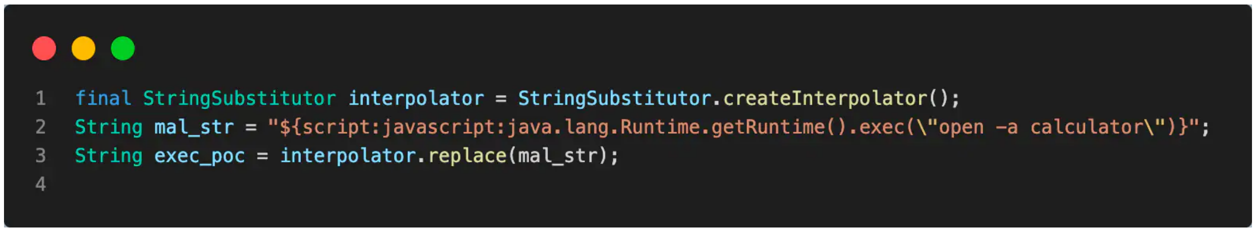
```
public final class StringLookupFactory {

-     /**
-      * Default properties file classpath location.
-      */
-     private static final String DEFAULT_RESOURCE = "org/apache/commons/text/lookup/defaults.properties";
-
-     /**
-      * Default mapping.
-      */
-     private static final Properties DEFAULTS = loadProperties();
-
```

The figure above shows that the variable DEFAULT_RESOURCE specifies the path for the default properties that include the three vulnerable lookups.



```
301 src/main/java/org/apache/commons/text/lookup/StringLookupFactory.java
-     private static Properties loadProperties() {
-         final Properties properties = new Properties();
-         try (InputStream inputStream = ClassLoader.getSystemResourceAsStream(DEFAULT_RESOURCE)) {
-             properties.load(inputStream);
-         } catch (final IOException e) {
-             throw new UncheckedIOException(e);
-         }
-         return properties;
-     }
-
```

The figure above loads the variable DEFAULT_RESOURCE.



```
final StringSubstitutor interpolator = StringSubstitutor.createInterpolator();
String mal_str = "${script:javascript:java.lang.Runtime.getRuntime().exec(\"open -a calculator\")}";
String exec_poc = interpolator.replace(mal_str);

```

Once the adversary connects to the server of the targeted company/server, he/she can use the ‘script’ key to write malicious Javascript code and use the replace function to execute the code. The paloalto website describes that the string variable mal_str “is interpolated by the default interpolator replace() function, which will trigger scriptStringLookup" due to the usage of the “script” key.

Even though the example provided by the paloalto website was simply to open a calculator, adversaries can perform countless attacks that are more severe through the ability to write JavaScript code in a remote server. Adversaries can include XSS and CSRF codes to steal sensitive information regarding the users (financial data such as the bank account of the users). When users visit the website of the target company that includes the Javascript code by the adversary, he/she can steal sensitive information regarding the users (financial data such as the bank account of the users) and personal information of the users (cookie value or security permissions on a different website that would have been blocked to perform an attack at a separate location). Adversaries can then force the user to send money to themselves or sell personal information to other people as well. In other words, important information of the users can get stolen when an adversary takes advantage of the 'script' field, which can lead to further vulnerabilities. Moreover, adversaries can gain access to the geolocation, webcam, microphone, and files of the user by using social engineering. In fact, acunetix website describes that social engineering can be used to "pull off advanced attacks including cookie theft, planting trojans, keylogging, phishing, and identity theft." Adversaries can also force the users to download additional malware to the device that people are using.

https://www.acunetix.com/websitesecurity/cross-site-scripting/

Similarly, an adversary can use the 'url' lookup to force the website of the target server to interact with a malicious URL that he/she has made prior to the attack. The adversary can perform the same set of actions as when he/she uses the 'script' field (${url}:javascript:java.lang.Runtime.getRuntime().exec(...)}";. Instead of writing the JavaScript code in the space provided by using the 'script' fielld, he/she can include the code that would be malicious to users in the URL that he/she made and connect it with the targeted website.

https://brightsec.com/blog/dns-attack/#what-is-dns

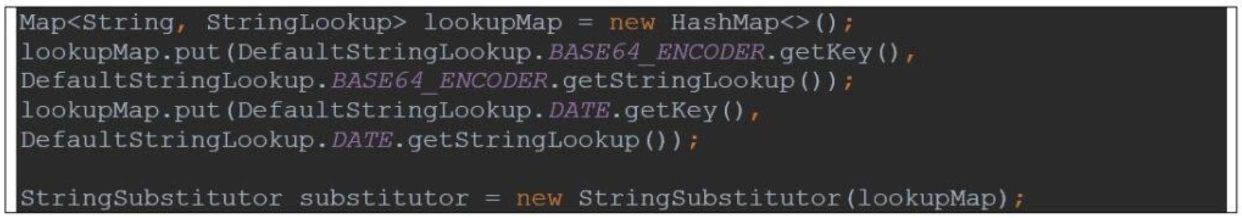
Finally, the adversary can use the 'dns' lookup. DNS, which stands for Domain Name System, converts the domain of the website into IP addresses. Once again, attacks using the 'dns' field are similar to attacks done by 'url' and 'script' lookup keys. Here, the attacker could make DNS request attacks, which could threaten remote server applications by losing functionality through the shutdown of a website.

5. How was the bug fixed? Explain whether the patch is a real fix or just a stopgap that could be circumvented later.

https://checkmarx.com/blog/cve-2022-42889-text4shell-vulnerability-breakdown/

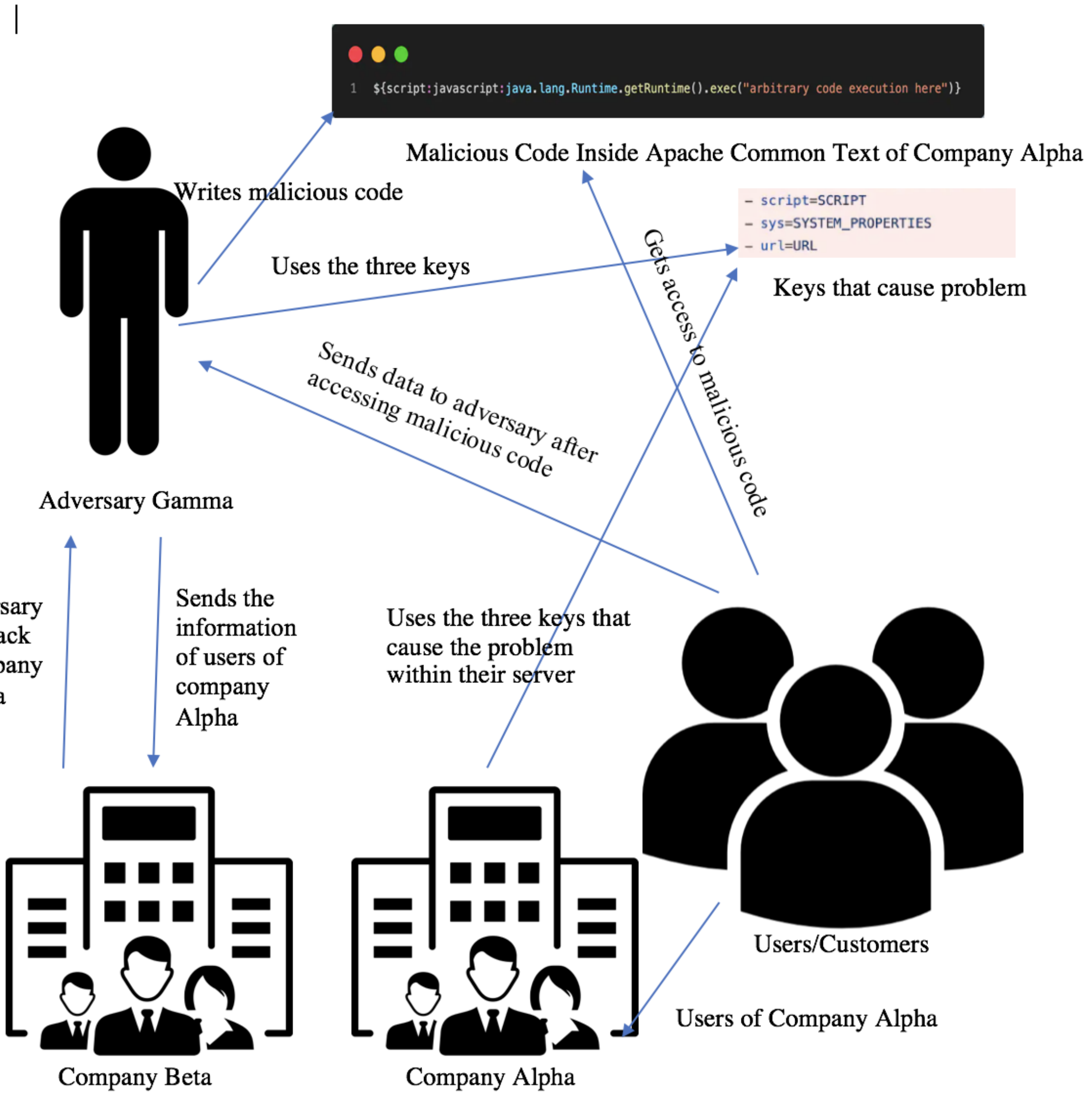
According to Checkmarx, the bug was fixed through the removal of the dns, script, and url lookups (the lookups that caused the vulnerability) from the default interpolator behavior. These three lookups that were deleted are replaced as methods that serve the same purposes. The StringLookupFactory.createDefaultStringLookups() method includes DefaultStringLookup.DNS for dns lookup, DefaultStringLookup.URL for url lookup, and DefaultStringLookup.SCRIPT for script lookup as default. The deletion of the lookups that led to the vulnerability and the addition of alternative methods perfectly fixes the problem instead of providing a stopgap that could be circumvented later since adversaries cannot use the three lookups anymore. However, we believe that Apache Common Texts should analyze the three lookups that caused the vulnerability to prevent similar problems from arising at different lookups or other places within the system.

Users need to upgrade their package to version 1.10.0 since versions ranging from 1.5 to 1.9 include the three lookups that caused the problem. If an upgrade to version 1.10.0 is impossible, the Checkmarx website addresses a possible solution to prevent the vulnerability: "the StringSubstitutor can be initialized with safe StringLookup configurations." If users are using versions that include the vulnerability and require the usage of the dangerous lookups, Checkmarx suggests sanitizing all user inputs before being interpolated through the usage of a whitelist in the input validation phase.



```
Map<String, StringLookup> lookupMap = new HashMap<>();
lookupMap.put(DefaultStringLookup.BASE64_ENCODER.getKey(),
DefaultStringLookup.BASE64_ENCODER.getStringLookup());
lookupMap.put(DefaultStringLookup.DATE.getKey(),
DefaultStringLookup.DATE.getStringLookup());

StringSubstitutor substitutor = new StringSubstitutor(lookupMap);
```

6. Threat model. Use diagrams and/or narrative to model the entire attack. Explain the different players (attacker, company, users, and other interacting parties).



```
${script:javascript:java.lang.Runtime.getRuntime().exec("arbitrary code execution here")}
```

```
- script=SCRIPT
- sys=SYSTEM_PROPERTIES
- url=URL
```

Malicious Code Inside Apache Common Text of Company Alpha
Writes malicious code
Uses the three keys
Keys that cause problem
Gets access to malicious code
Sends data to adversary after accessing malicious code
Adversary Gamma
Hires adversary to attack Company Alpha
Sends the information of users of company Alpha
Uses the three keys that cause the problem within their server
Users/Customers
Users of Company Alpha
Company Beta
Company Alpha

Consider two companies called Alpha and Beta that both sell phones to their customers. Since both companies sell the same product, they compete against each other to produce better-quality phones. Company Beta hires adversary Gamma to attack company Alpha and requires the adversary to steal information regarding the users of company Alpha.

Company Alpha uses the Apache Common Text within their website “to handle and manipulate strings beyond the capacity offered in the core Java library”. However, lazy workers of company Alpha fail to update their version of Apache Common Text used in their server/website and leave it as 1.7.0, allowing adversaries to perform attacks on their server.

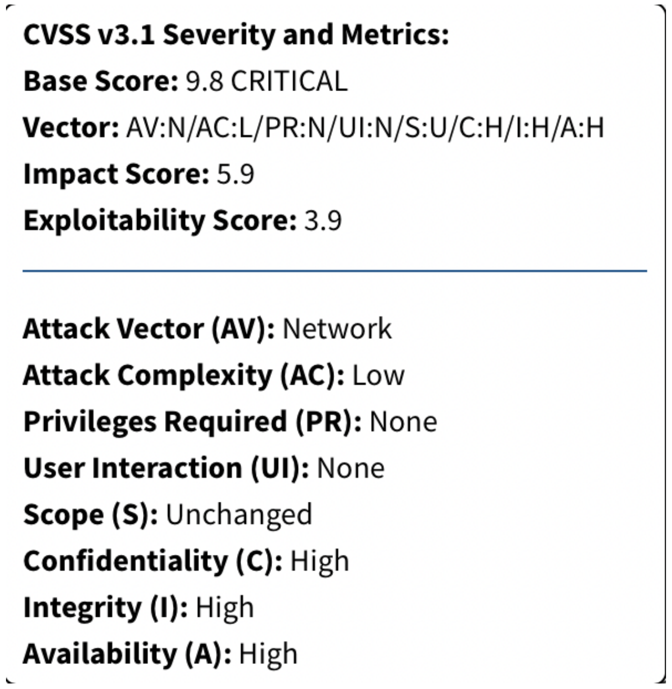
Adversary Gamma, after being hired by company Beta, realizes that the version of the Apache Common Text of company Alpha is 1.7.0. He learns the specific IP address of company Alpha and connects to their server by using terminal and netcat. He decides to use the three keys to attack company Alpha by writing malicious JavaScript code within the space provided as ___ in “${script:javascript:java.lang.Runtime.getRunTime().exec(“__”).

Adversary Gamma begins the attack by stealing the information of the users of company Alpha by using XSS and CSRF to send it to company Beta. Using Javascript code, adversary Gamma successfully takes personal and sensitive information regarding the users once they visit the website of company Alpha. It is important to note that the users do not realize that their personal information is getting stolen and being sent to adversary Gamma.

In addition, he steals the authority of an administrator of company Alpha. He takes sensitive information such as the model of the new phone released in a few months to send it to company Beta. Once company Beta receives the information, they can steal the ideas of the new model that would have been released by company Alpha and produce them prior to the release date of company Alpha.

Finally, after adversary Gamma sends all essential information, he deletes all of the information and shuts down the server of company Alpha.

7. Vector justification.



CVSS v3.1 Severity and Metrics:
Base Score: 9.8 CRITICAL
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Impact Score: 5.9
Exploitability Score: 3.9

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): None
Scope (S): Unchanged
Confidentiality (C): High
Integrity (I): High
Availability (A): High

References

https://nvd.nist.gov/vuln/detail/CVE-2022-42889#vulnCurrentDescriptionTitle

https://sysdig.com/blog/cve-2022-42889-text4shell/

https://nvd.nist.gov/vuln/detail/CVE-2022-44544

https://fossa.com/blog/cve-2022-42889-text4shell-vulnerability-impact-fixes/

Attack vector (AV): Network

Categorizing the attack vector for this CVE as “Network” is reasonable because the adversary can execute attacks remotely without connecting to any other devices. The vulnerability occurs when running a string lookup with keys involving “script”, “dns”, and “url” within the StringSubstitutor interpolator class. The usage of these keys will allow the adversary to infiltrate the server and execute arbitrary code. Since there are no other requirements (such as the need to connect to the same network as the server) for the adversary to successfully launch the attack, it is correct to assign the attack vector as local for this CVE.

Attack complexity (AC): Low

The connection to the application is completed when "${prefix: name}" is answered successfully after sending a crafted request using the Netcat command with keys such as "script","dns", and "url". Since the only requirement is to complete the code above, the complexity of the attack is considered to be low.

Privileges required (PR): None

The only requirement of this CVE is the usage of versions ranging from 1.5 to 1.9 of Apache Commons Text and the keys ("script", "url", and "dns") that allow the adversary to connect to the server. Since the adversary does not require privileges or authorization for the application when exploiting this vulnerability, it is correct to classify this category as None.

User Interaction (UI): None

Unlike other CVEs such as CVE-2022-39376 (a Free Asset and IT Management Software package that provides ITIL service desk features, licenses tracking, and software auditing through the click of "mailto" links from users to be injected) that require users to perform certain actions to successfully launch the attack, this CVE can be executed successfully without requiring any information from the user. Therefore, it is correct to classify the user interaction of this CVE as "None".

Scope: Unchanged

We argue that the scope of this CVE is “changed”. If the adversary successfully launches the attack, he/she can affect other resources such as the microphone and webcam of the user’s device. In addition, the adversary can spread malware using this CVE (adversaries can include javaScript that forces the user to download malware and spread it to other people). Since attackers can affect external applications, we believe that the scope of this CVE is “changed”.

Confidentiality Impact (C): High

The confidentiality of an application is completely lost when it is attacked by this CVE by an adversary. Once the resources within the components are exposed to the adversary, he/she can modify/add/delete all files and systems of the application without any restriction. In other words, the attacker can simply perform all actions as if he/she is the administrator of the application. Since a single attack forces the application to be defenseless, we strongly agree that the confidentiality impact of this CVE is “high”.

Integrity Impact (I): High

The integrity impact of this CVE is categorized as “high” because the application loses the ability to protect itself from the adversary. One of the problems that may arise once the adversary successfully uses this CVE to attack the application is delegating all of the administrator’s authority to the attacker, allowing the attacker to steal the administrative authority of the application from the original administrator. Since the application can be exposed to elements that will threaten the application through the attack of an adversary, we strongly agree that the integrity impact of this CVE is high.

Availability Impact (A): High

The availability impact of this CVE is high. When an application using the Apache Commons Text library is attacked through this CVE, arbitrary codes can be added. The arbitrary code can allow the adversary to take full authority of the application as if the adversary is the administrator. Therefore, the availability impact on the target application due to this CVE is high.