URL Restriction Bypass in plantuml/plantuml

Valid Reported on Apr 10th 2022

0

Description

The validation of URLs contains flaws that allow bypassing security restrictions that are applied in the security profiles of PlantUML. There are two different flaws through which validation mechanisms can be circumvented.
In the examples images are loaded to showcase the bypass. However, it applies to all methods in PlantUML that can be used to retrieve remote content like `!include` or `%loadJSON`.

Accessing Local and Intranet Addresses in SecurityProfile.INTERNET

When running PlantUML with `SecurityProfile.INTERNET`, access to URLs with IP addresses or local addresses like `localhost` is denied. The `forbiddenURL` function is responsible for filtering out those addresses.
Relevant code:
https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/SURL.java#L235-L242

```
    private boolean isUrlOk() {
        [...]
        if (SecurityUtils.getSecurityProfile() == SecurityProfile.INTERNET)
            if (forbiddenURL(cleanPath(internal.toString())))
                return false;

            final int port = internal.getPort();
            // Using INTERNET profile, port 80 and 443 are ok
            return port == 80 || port == 443 || port == -1;
        }
        [...]
```


https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/

SURL.java#L246-L252

```
    private boolean forbiddenURL(String full) {
        if (full.matches("^https?://\\d+\\.\\d+\\.\\d+\\.\\d+.*"))
            return true;
        if (full.matches("^https?://[^.]+/.*"))
            return true;
        return false;
    }
```

The validation can be bypassed with IP addresses that contain internal 0 components. Those components can be omitted. The first validation regex above only takes IPs consisting of 4 dotted parts into account. Access to `localhost` (IP `127.0.0.1`) can thus be achieved by using `127.1` as host IP in the URL for example.

Proof of Concept:

Use the security profile "INTERNET" as described at https://plantuml.com/de/security or by setting `System.setProperty("PLANTUML_SECURITY_PROFILE", "ALLOWLIST");`.
Start a HTTP server to see the incoming request.
Create the following diagram:

```
@startuml
Bob -> Alice : hello <img:"http://127.1/test1.png">
@enduml
```

Bypassing Allowlist Validation

When running PlantUML with the restrictive `SecurityProfile.ALLOWLIST` according to the documentation at https://plantuml.com/en/security the following restrictions apply:
In ALLOWLIST mode, PlantUML cannot have any access to local files or URL. You have to use allowlists to explicitely authorize access to local or remote ressources.
A flaw in the validation check that determines if a provided URL is allowed, lets attackers access arbitrary URLs. When a URL gets checked, the `cleanPath` function removes the "userinfo" part of the URL that contains username and password (if it exist). This is done by calling `removeUserInfoFromUrlPath`, which rewrites the URL string by basicall
anything between the scheme and a `@` character.
The problem is that the regex `PATTERN_USERINFO` does not differentiate if there is actually a

userinfo part or not. The `@` could also be located in other parts of the URL, like the path, query or hash. Furthermore when the HTTP request is sent, the initial unmodified URL is used.

Through this behavior attackers can create URLs that pass the validation check and match an entry of the allow list, but actually point to a totally different URL, than what is validated. This can be achieved by using a URL of the following format: `attacker-target#@allow-list-entry`. Where `attacker-target` is the actual target URL, and `allow-list-entry` an entry in the list of allowed URLs. So if `https://fileserver.tld` is in the allow list, the URL `https://target.tld/path/file?a=b#@fileserver.tld` would be recognized as allowed, but would send a request to `https://target.tld/path/file?a=b`.
Relevant code:
https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/SURL.java#L254-L261

```
    private boolean isInAllowList() {
        final String full = cleanPath(internal.toString());
        for (String allow : getAllowList())
            if (full.startsWith(cleanPath(allow)))
                return true;

        return false;
    }
```

https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/SURL.java#L263-L272

```
    private String cleanPath(String path) {
        // Remove user information, because we don't like to store user/pas
        // userTokens in allow-list
        path = removeUserInfoFromUrlPath(path);
        path = path.trim().toLowerCase(Locale.US);
        // We simplify/normalize the url, removing default ports
        path = path.replace(":80/", "");
        path = path.replace(":443/", "");
        return path;
    }
```


https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/

SURL.java#L650-L657

```
    private static String removeUserInfoFromUrlPath(String url) {
        // Simple solution:
        final Matcher matcher = PATTERN_USERINFO.matcher(url);
        if (matcher.find())
            return matcher.replaceFirst("$1$3");

        return url;
    }
```

https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/SURL.java#L113

```
    private static final Pattern PATTERN_USERINFO = Pattern.compile("(^http
```

Proof of Concept:

Use the security profile "ALLOWLIST" and add "https://plantuml.com/" to the allow list (see https://plantuml.com/de/security).

```
System.setProperty("PLANTUML_SECURITY_PROFILE", "ALLOWLIST");
System.setProperty("plantuml.allowlist.path", "https://plantuml.com/");
```

Start a HTTP server to see the incoming request.
Create the following diagram:

```
@startuml
Bob -> Alice : hello <img:"http://127.1/test1.png#/@plantuml.com/">
@enduml
```

Note: It seems to be a mistake that SURL also uses `SecurityUtils.PATHS_ALLOWED` (`plantuml.allowlist.path`) instead of `plantuml.allowlist.url` as described in the documentation. The separate allowlist for URLs does not exist at all in the code. `plantuml.allowlist.path` needs to be used for paths *and* URLs. However, the functionality is

not affected. Also the allow list entries are split by ";" as described in the documentation. See: https://github.com/plantuml/plantuml/blob/v1.2022.3/src/net/sourceforge/plantuml/security/SURL.java#L275

Impact

An attacker can abuse this to bypass URL restrictions that are imposed by the different security profiles and achieve server side request forgery (SSRF). This allows accessing restricted internal resources/servers or sending requests to third party servers.

Occurrences

SURL.java L650-L657

Allowlist validation bypass

CVE
CVE-2022-1379
(Published)

Vulnerability Type
CWE-918: Server-Side Request Forgery (SSRF)

Severity
High (7.2)

Registry
Other

Affected Version
v1.2022.3 and below

Visibility
Public

Status
Fixed

Found by
Tobias S. Fink
@7085
legend

This report was seen 851 times.

We are processing your report and will contact the **plantuml** team within 24 hours. 8 months ago

We have contacted a member of the **plantuml** team and are waiting to hear back 8 months ago

We have sent a follow up to the **plantuml** team. We will try again in 7 days. 7 months ago

A **plantuml/plantuml** maintainer has acknowledged this report 7 months ago

PlantUML validated this vulnerability 7 months ago

Tobias S. Fink has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

We have sent a fix follow up to the **plantuml** team. We will try again in 7 days. 7 months ago

We have sent a second fix follow up to the **plantuml** team. We will try again in 10 days. 7 months ago

We have sent a third and final fix follow up to the **plantuml** team. This report is now considered stale. 7 months ago

PlantUML marked this as fixed in **V1.2022.5** with commit **93e596** 6 months ago

The fix bounty has been dropped ✖

This vulnerability will not receive a CVE ✖

SURL.java#L650-L657 has been validated ✔

Sign in to join this conversation

Chat with us