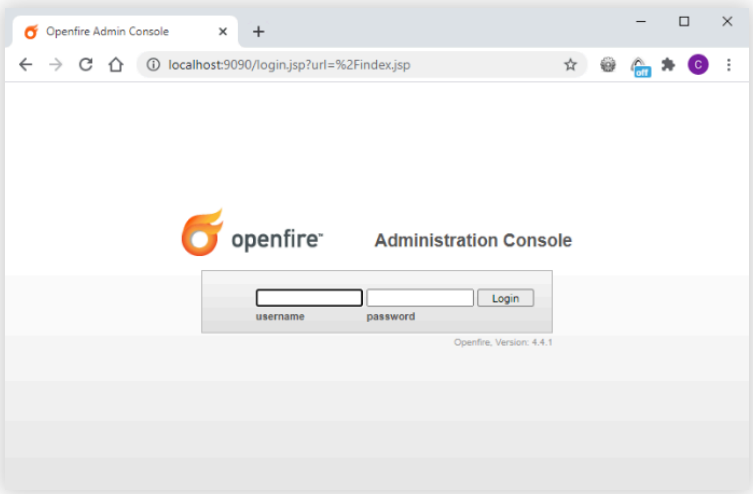
Vulnerabilities in the Openfire Admin Console

Written by Alexandr Shvetsov on August 4, 2020

Openfire is a Jabber server supported by Ignite Realtime. It's a cross-platform Java application, which positions itself as a platform for medium-sized enterprises to control internal communications and make instant messaging easier.

I regularly see Openfire on penetration testing engagements, and most of the time all interfaces of this system are exposed to an external attacker, including the administrative interface on 9090/http and 9091/https ports:



Openfire Administration Console

Since the Openfire system is available on GitHub, I decided to examine the code of this web interface. This is a short writeup about two vulnerabilities I was able to find.

Full Read SSRF Vulnerability

Assigned CVE: CVE-2019-18394
Vulnerable file: FaviconServlet.java (the fix commit)

This vulnerability allows an unauthenticated attacker to send arbitrary HTTP GET requests to the internal network, and obtain full-sized outputs from the targeted web services.

Let's look at the vulnerable code contained in the FaviconServlet.java file:

```
...
public void doGet(HttpServletRequest request,
HttpServletResponse response) {
      String host = request.getParameter("host");
      // Check special cases where we need to change host to
get a favicon
```

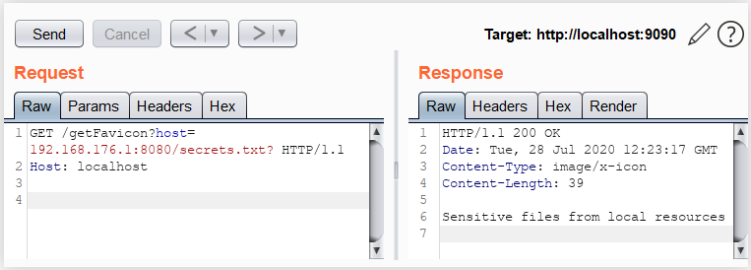
```
        host = "gmail.com".equals(host) ? "google.com" : host;

        byte[] bytes = getImage(host, defaultBytes);
        if (bytes != null) {
            writeBytesToStream(bytes, response);
        }
}

private byte[] getImage(String host, byte[] defaultImage) {
    // If we've already attempted to get the favicon twice
and failed,
    // return the default image.
    if (missesCache.get(host) != null &&
missesCache.get(host) > 1) {
        // Domain does not have a favicon so return default
icon
        return defaultImage;
    }
    // See if we've cached the favicon.
    if (hitsCache.containsKey(host)) {
        return hitsCache.get(host);
    }
    byte[] bytes = getImage("http://" + host +
"/favicon.ico");
    ....
}
...
```

In the *doGet* and *getImage* methods the code gets the host variable from the get parameters, and constructs an URL from it without any constraints to the component parts. Thus, an attacker can place any sequence of characters inside of it, and make the server connect to any URL they want.

An HTTP request to test the vulnerability:

```
GET /getFavicon?host=192.168.176.1:8080/secrets.txt?
HTTP/1.1
Host: assesmenthost.com:9090
```

An example of a vulnerable server's behavior:



An example of CVE-2019-18394 exploitation in Burp Suite

Arbitrary File Read Vulnerability

Assigned CVE: CVE-2019-18393
Vulnerable file: PluginServlet.java (the fix commit)

This vulnerability affects only Windows installations of the OpenFire server, and an attacker has to have an administrative account on the

server to exploit it.

The vulnerable code is located in the PluginServlet.java file:

```
...
@Override
public void service(HttpServletRequest request,
HttpServletResponse response) {
    String pathInfo = request.getPathInfo();
    if (pathInfo == null) {

response.setStatus(HttpServletResponse.SC_NOT_FOUND);
    }
    else {
        try {
            // Handle JSP requests.
            if (pathInfo.endsWith(".jsp")) {
                ...
            }
            // Handle servlet requests.
            else if (getServlet(pathInfo) != null) {
                handleServlet(pathInfo, request, response);
            }
            // Handle image/other requests.
            else {
                handleOtherRequest(pathInfo, response);
            }
        }
       ...
}

private void handleOtherRequest(String pathInfo,
HttpServletResponse response) throws IOException {
    String[] parts = pathInfo.split("/");
    // Image request must be in correct format.
    if (parts.length < 3) {

response.setStatus(HttpServletResponse.SC_NOT_FOUND);
        return;
    }

    String contextPath = "";
    int index = pathInfo.indexOf(parts[1]);
    if (index != -1) {
        contextPath = pathInfo.substring(index +
parts[1].length());
    }

    File pluginDirectory = new
File(JiveGlobals.getHomeDirectory(), "plugins");
    File file = new File(pluginDirectory, parts[1] +
File.separator + "web" + contextPath);

    // When using dev environment, the images dir may be
under something other that web.
    Plugin plugin = pluginManager.getPlugin(parts[1]);
    ...
}
```

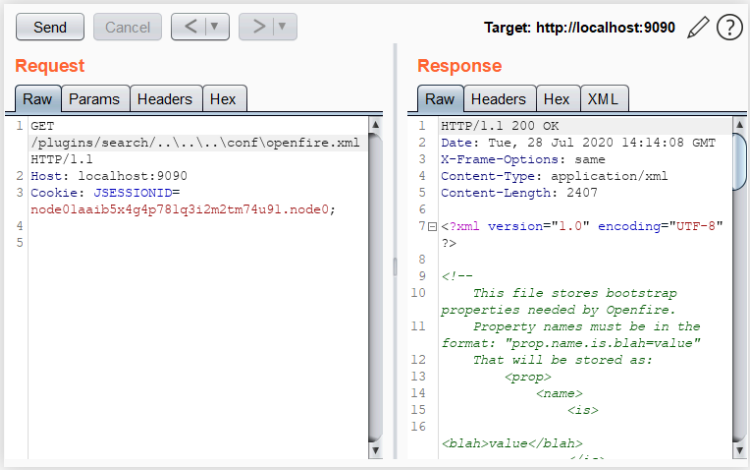
This vulnerability is interesting in that it exists in the URI itself, and the HTTP parameters are not involved.

The *handleOtherRequest* method, which is responsible for handling the */plugin/search/* path, makes an assumption that if it splits the *pathInfo* variable by the "/" character, the obtained sequence will be safe to use. But since there is no allowlist of characters or any checking for the "\" character, we can perform a path-traversal attack for Windows systems.

To test the vulnerability, log in to the server, and send the following request with the administrator's JSESSIONID cookie:

```
GET /plugins/search/..\..\..\conf\openfire.xml HTTP/1.1
Host: assesmenthost.com:9090
Cookie: JSESSIONID=node01aaib5x4g4p781q3i2m2tm74u91.node0;
```

An example of a vulnerable server's behavior:



An example of CVE-2019-18393 exploitation in Burp Suite

Conclusion

Both discovered vulnerabilities were the result of unexisting user input data validation. So, my recommendation for the developers was to validate the parameters before performing sensitive operations with them, such as reading files and accessing URLs.

It's worth noting that system administrators should also protect all of the administrative interfaces against unauthorized access, and not make them available to external or internal attackers.

The timeline:

- 2 October, 2019 – Reported to Ignite Realtime
- 3 October, 2019 – Issues have been addressed in main codeline
- 1 November, 2019 – Ignite Realtime released the 4.4.3 version
- 4 August, 2020 – Public disclosure

Links:

- https://issues.igniterealtime.org/browse/OF-1885
- https://issues.igniterealtime.org/browse/OF-1886

If you have an Openfire server, make sure you've updated it to version 4.4.3 or higher.

Author

Alexandr Shvetsov
Penetration Testing Expert

shvetsovalex007

Arbitrary File Read, SSRF, Web Application Security

Previous
Attacking MS Exchange Web Interfaces

Next
Kerberoasting without SPNs

Twitter