Prince of Persia – Game Over

researchcenter.paloaltonetworks.com/2016/06/unit42-prince-of-persia-game-over/

Summary

Unit 42 published a blog at the beginning of May titled “Prince of Persia,” in which we described the discovery of a decade-long campaign using a formerly unknown malware family, Infy, that targeted government and industry interests worldwide.

Subsequent to the publishing of this article, through cooperation with the parties responsible for the C2 domains, Unit 42 researchers successfully gained control of multiple C2 domains. This disabled the attacker’s access to their victims in this campaign, provided further insight into the targets currently victimized in this operation, and enabled the notification of affected parties.

Post Publication

In the week following the publication of the original blog, we observed no unusual changes to the C2 infrastructure. Existing domains did move to new IP addresses, as we had previously seen periodically. Some new install domains were added, adhering to naming conventions of current domains (see appendix for new IOCs).

The attackers developed a new version (31), and we observed this deployed against a single Canadian target.

The file descriptions remained essentially the same (“CLMediaLibrary Dynamic Link Library V3”). Most importantly, there was **no change to the encoding key** (now using offset 20, and offset 11 for second pass against URL encoding) that we had observed being used for the entire decade-long campaign, and documented in our previous blog. From this we conclude that the attackers were unaware of our initial report.

Sinkhole

Through cooperation with the parties responsible for the C2 domains, we took control of all but one of them, transferring the A records to a server we controlled. This prevented the attackers from being able to subsequently make any further changes to the domain configurations, issue commands to victims, or capture any further data for the majority of victims. An analysis of connections after transfer suggests that the attackers may have used a third-party service to try to understand why they had suddenly lost almost all of their traffic. Figure 1 shows that tool, a geographic representation of victim-C2 traffic, with all but one at that time now communicating with our sinkhole server.

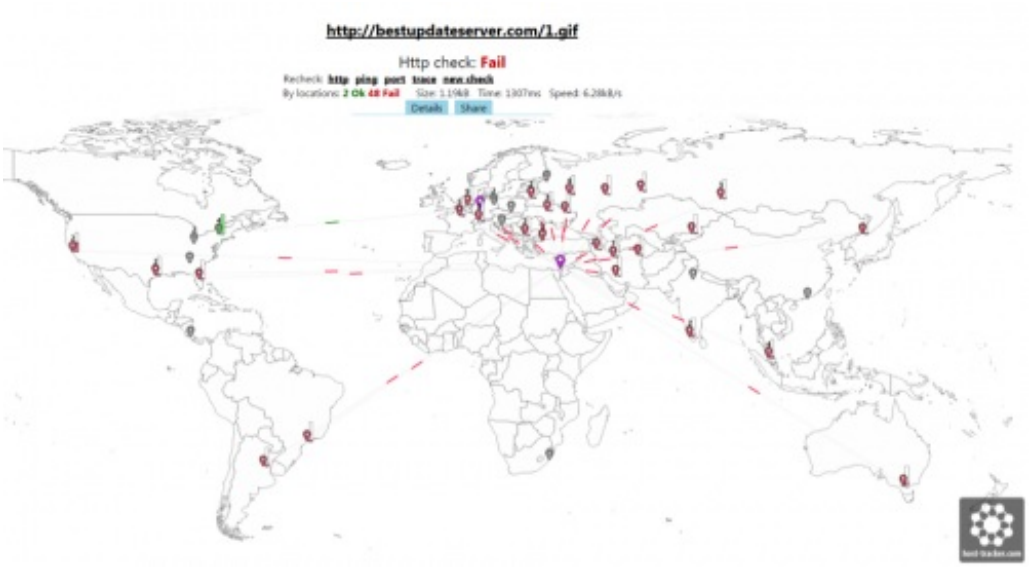

Figure 1 Graphical representation of victim traffic to C2

We have since transferred sinkhole control to Shadowserver, whom we thank for subsequent victim notification & remediation (https://www.shadowserver.org/wiki/pmwiki.php/Involve/GetReportsOnYourNetwork).

Victims

We were able to analyze victim C2 traffic to understand who were victims of the Infy campaign. We identified 456 malware agents installed on 326 victim systems, in 35 countries. Figure 2 shows a geographical breakdown of victim locations. We noted in our original blog the large amount of targeting of Iranian citizens in this campaign, we observed almost one-third of all victims to be Iranian. Also of note was the low overall volume of victims, compared to, for example, crimeware campaigns.

Figure 2 Geographic location of victims. Please note that New Zealand has been omitted from this map only because we observed no victim activity there.

Versions

In our original blog, we noted two distinct primary variants of the Infy malware. In addition to the original “Infy” variant, we also see the newer, more sophisticated, interactive, and fuller-featured “Infy M” variant deployed against

apparently-higher-value targets. Overall, 93% of all victims were infected with Infy, and 60% with Infy “M” (Figure 3). Combined with the low total number of victims, this suggests a great deal of care given to each individual campaign target. The large number of victims with both variants may relate to their complimentary feature set, or represent an “upgrade” path on victims from the original variant infection, later adding the “M” variant as targets appeared more compelling to the attackers.

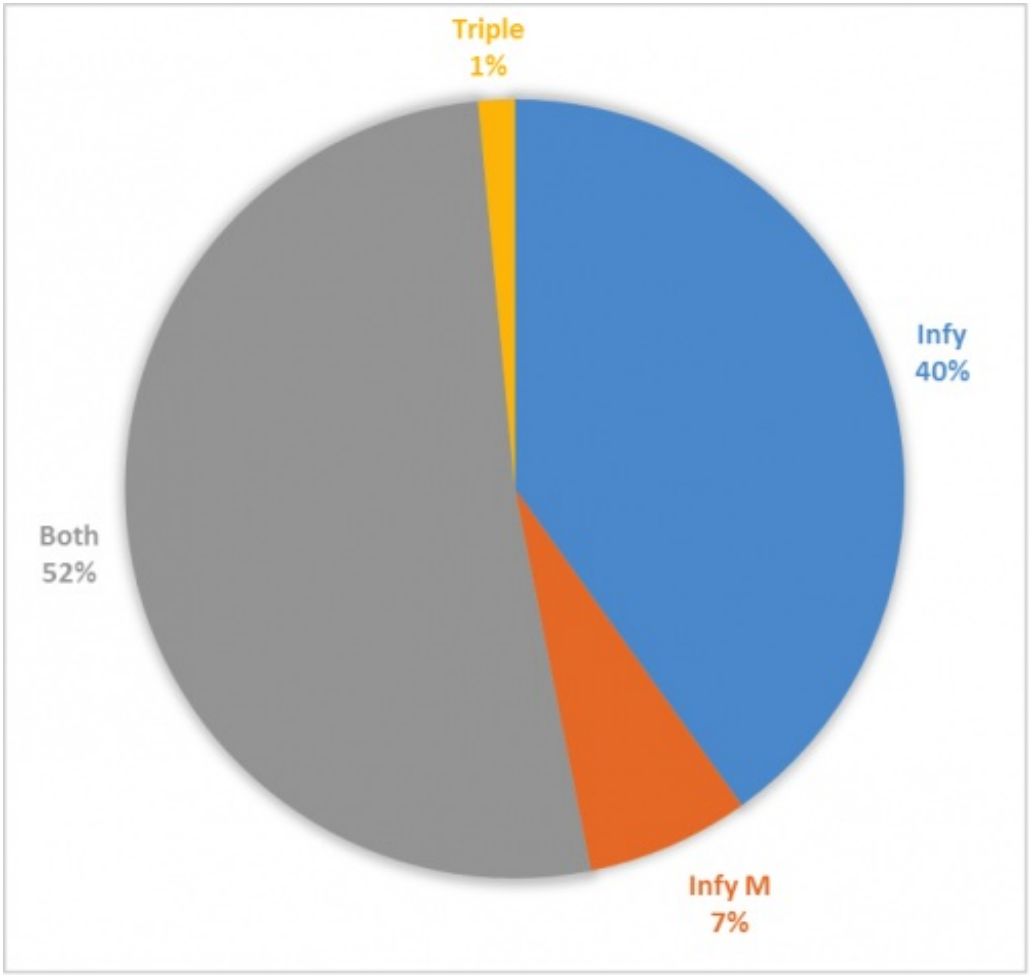

Figure 3 Breakdown of Infy vs. Infy “M” infections

For the Infy “M” variant, we note that the majority of targets are using the latest version (7.8), and that none are using the older 6.x versions at all (Figure 4). This suggests that these higher-value targets are paid much more attention, being kept up-to-date with the latest version.

In contrast, for the more basic original Infy variant, we note a full spectrum of versions installed (Figure 5), with many victims on older versions – including the original, decade-old V1 – suggesting much less concern is paid to these individual targets (note that we did observe a small number of the older 6.x versions but these do not announce their version when connecting).

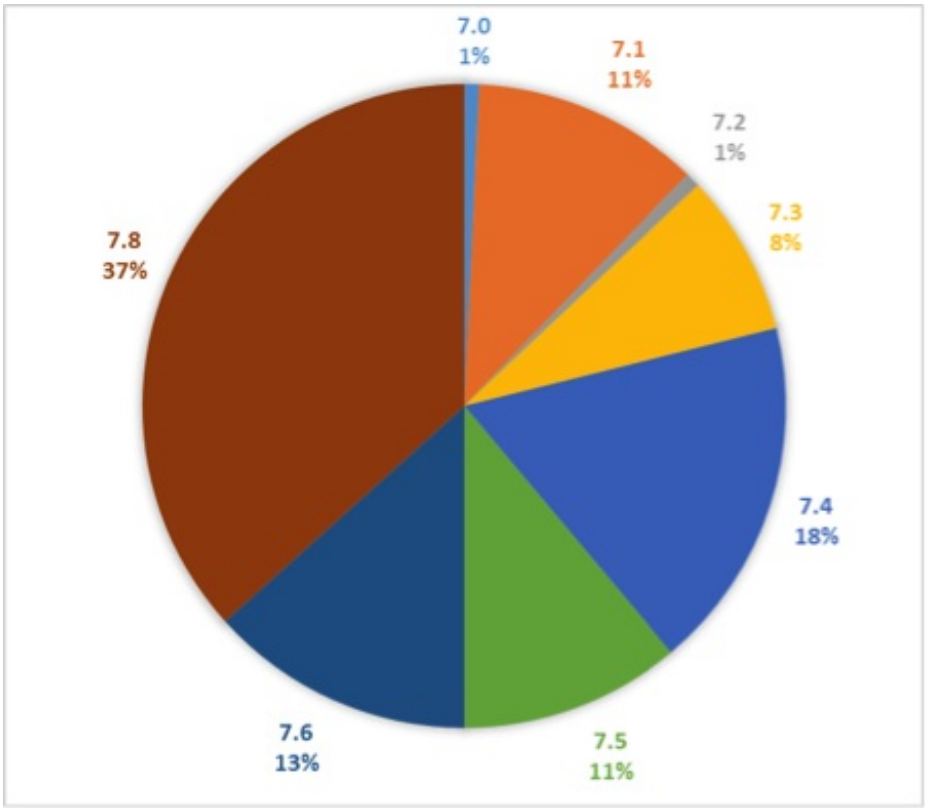

Figure 4 Infy "M" Victim versions

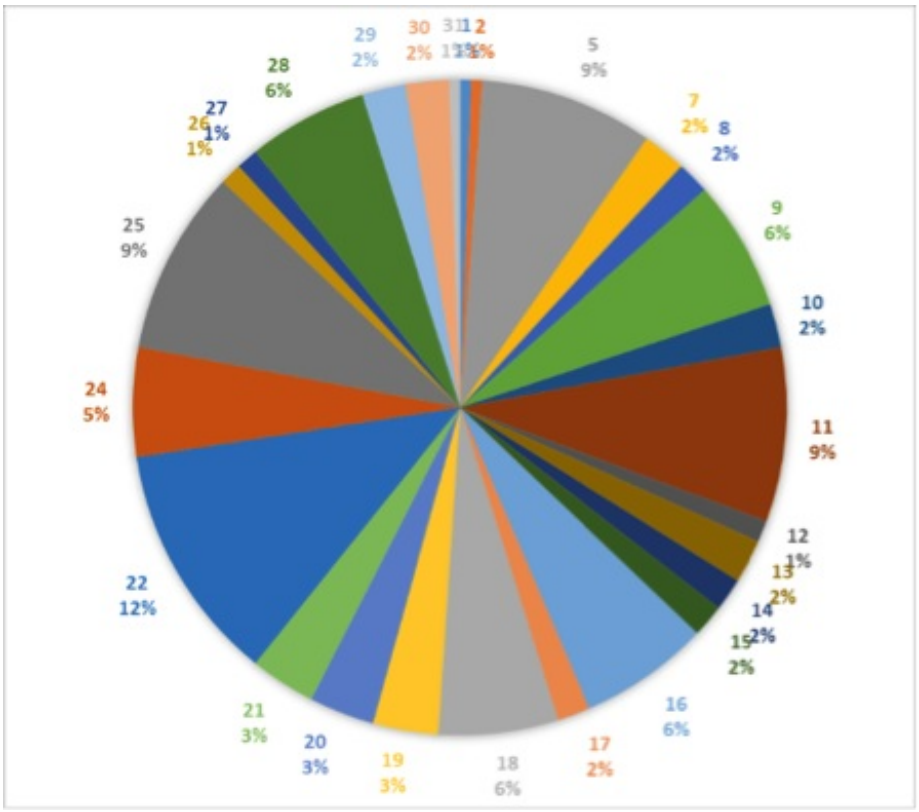

Figure 5 Infy"Original" Victim versions

Game Over

Shortly after the takedown, as well as a new Infy version (31), we also observed the registration of multiple domains

using a previously-seen pattern, against known campaign IP addresses. Almost every domain in the pattern-range box4035[.]net – box4090[.]net (138.201.0.134). These were not observed in any sample C2 lists however. Bestwebstat[.]com was sinkholed by another operator.

Some victims infected with Infy versions 15-24 still used the C2 server us1s2[.]strangled[.]net, which remained in the hands of the attacker. In early June the attackers used this C2 to issue instructions to download new Infy "M" version 8.0 from us1s2[.]strangled[.]net/bdc.tmp. This was the first time we had observed an Infy variant being directly updated to Infy "M". This used camouflage name "Macromedia v4", changed from "v3" seen in Infy v31. They also removed the voice recording capability in this version.

uvps1[.]cotbm[.]com was used for data exfiltration, previously at 138.201.47.150, after publishing of our original blog moving to 144.76.250.205. It was also hosting malware updates at /themes/u.php.

They also added a curious C2 entry "hxxp://box" (note: defanged for publishing). It's unclear how this should function; possibly a compromised victim intranet device, or the attackers have modified the HOSTS file on the victim computer.

After the take-down, the attackers began to add server IP addresses as well as domain names to their malware C2 list. They also slightly modified their ZIP password from "Z8(2000_2001ul" to "Z8(2000_2001uIEr3". Their new malware version added antivirus checks for Kaspersky Labs, Avast, and Trend Micro. The malware data capture now searches for file extensions:

.doc, .docx, .xls, .xlsx, .xlr, .pps, .ppt, .pptx, .mdb, .accdb, .db, .dbf, .sql, .jpg, .jpeg, .psd, .tif, .mp4, .3gp, .txt, .rtf, .odt, .htm, .html, .pdf, .wps, .contact, .csv, .nbu, .vcf, .pst, .zip, .rar, .7z, .zipx, .pgp, .tc, .vhd, .p12, .crt.pem,.key.pfx, .asc, .cer, .p7b, .sst, .doc, .docx, .xls, .xlsx, .xlr, .pps, .ppt, .pptx.

and folder locations:

:\$recycle.bin, :\documents and settings, :\msocache, :\program files, :\program files (x86), :\programdata, :\recovery, :\system volume information:\users, :\windows, :\boot, :\inetpub, :\i386.

The malware continued to use the **identical decryption key** seen over the entire history of this campaign.

Mid-June, through cooperation with the parties responsible for the C2 domains and law enforcement, we were able to get the remaining C2 domains null-routed and the directly-IP-addressed server disabled. This is the end of a decade-long campaign, though we naturally expect to see this actor back in some other guise before long.

Thanks to the Malware research team – Yaron Samuel, Artiom Radune, Mashav Sapir, Netanel Rimer – for assistance in the takedown.

Appendix 1 – Exfiltration Algorithm

The malware uses a different algorithm than that used for encrypting the malware strings to encrypt the exfiltration data, including:

1. Keylogger data + language.
2. Malware logs – installation time, DLL path and name, log path, number of downloads, number of successful/failed connections.
3. Information about the victim computer: Time zone, list of drives and types, running processes, disk info.

First the malware adds 1 to all bytes, then an encryption key is initialized based on the victim computer name (the offset in the key is calculated by sum of the computer name letters %key length). Then the key is used to encrypt the data (see decrypt function). The encrypted data is then base64 encoded.

Exfiltration data decryption python code:

```
1   import os,sys
2   import string
3   import base64
4   import fileinput
5   FIRST_PHASE = "OQTJEqtsK0AUB9YXMwr8idozF7VWRPpnhNCHI6Dlkaubyxf5423jvcZ1LSGmge"
6   SECOND_PHASE = "PqOwI1eUrYtT2yR3p4E5o6WiQu7ASlDkFj8GhHaJ9sKdLfMgNzBx0ZcXvCmVnb"
7   global FULL_KEY
8   FULL_KEY= ""
9   def sub_1_for_hex(str_input):
10      str_output = ""
11      for letter in str_input:
12          try:
13              str_output += chr(ord(letter)-1)
14          except:
15              print "sub_1_for_hex func problem"
16              continue
17      return str_output
18
19  def sum_comp_name(comp_name):
20      sum = 0
21      for letter in comp_name:
22          sum+= ord(letter)
23      return sum
24
25  def init_key(comp):
26      comp_name_sum = sum_comp_name(comp)
27      carry = divmod(comp_name_sum, 62)
28      index = carry[1] -1
29      end_key = FIRST_PHASE[:index]
30      key = FIRST_PHASE[index:]
```

```
31      key = key + end_key
32      key = key + key
33      return key
34
35  def decrypt(num_list,offset):
36      global FULL_KEY
37      input = ""
38      for num_str in num_list:
39          try:
40              input += num_str.decode('hex')
41          except:
42              input += ')'
43      result = ""
44      for i, c in enumerate(input):
45          i = i % 62 +1
46          try:
47              index = FULL_KEY.index(c)-1
48          except ValueError:
49              result += c
50              continue
51          translated = SECOND_PHASE[(index - i +offset) % len(SECOND_PHASE)]
52          result += translated
53      return result
54
55  def found_infy_enc_data(line):
56      found_infy_str = "show=\"---------- Administration Reporting Service "
57      found_infy_index = line.find(found_infy_str)
58      if not found_infy_index==-1:
59          return True,found_infy_index
60      else:
61          return False,found_infy_index
62  def extract_comp_name(line):
```

```
63          comp = r"\xd\xa-----"
64          comp_index = line.find(comp)
65          comp_name = line[comp_index+len(comp):]
66          comp_name = comp_name[:comp_name.find("-----")]
67          print "(((=)))" + comp_name
68          return comp_name
69
70      def extract_enc_data(line):
71          header = r"\xd\xa_____"
72          start_index = line.find(header)+len(header)
73          line = line[start_index:]
74          endindex = line.index("_____\" value=")
75          line = line[:endindex]
76          return line
77
78      def write_enc_infy_data_to_file(dec_line,comp_name,filename):
79          file1 = open(filename + "\\" + comp_name + ".txt",'ab')
80          file1.writelines(dec_line)
81          file1.close()
82
83      def enc_wrapper(enc,comp_name):
84          global FULL_KEY
85          print FULL_KEY
86          FULL_KEY = init_key(comp_name)
87
88          enc_final = ""
89          for letter in enc:
90              if len(hex(ord(letter))[2:])==1:
91              enc_final += "0" + hex(ord(letter))[2:]
92            elif len(hex(ord(letter))[2:])==2:
93              enc_final += hex(ord(letter))[2:]
94            else:
```

```
95              print "not good hex length"
96              exit()
97
98          enc = enc_final.upper()
99
100         enc = enc.replace("2E","21")
101         enc = enc.replace("C5DC5A","")
102         enc = enc.replace("D03D00","")
103         enc = enc.replace("0B0E","2121")
104
105         enc = enc.replace("01","21")
106
107         enc_len = len(enc)
108
109         enc_rev = ""
110         num_list = []
111         enc_print =""
112         for i in range(0,enc_len/2):
113             enc_rev = enc[-2:]
114             if not enc_rev=="0B" and not enc_rev=="0E" and not enc_rev=="00" and not enc_rev=="D0":
115                 enc_print +=enc_rev
116                 num_list.append(enc_rev)
117             enc= enc[:-2]
118
119         #the first part is always ok
120         dec_str = decrypt(num_list,0)
121         final = sub_1_for_hex(dec_str)
122         index = final.find("OK: Sent")
123         if index==-1:
124             print comp_name + " - did not found OK: Sent !!!!\n\n\n\n"
125             #exit()
126         decrypt_data = comp_name + " ++==++ " +  str(i) + ": " + final + "\n"
```

```
127
128         final_start = final[0:500]
129         if final_start in UNIQUE_DATA:
130             print comp_name + " already have this data"
131             return
132         UNIQUE_DATA.append(final_start)
133         index = final.find("Installed Date:")
134
135         if index==-1:
136             for i in range(1,61):
137                 dec_str = decrypt3(num_list,i)
138                 final = sub_1_for_hex(dec_str)
139
140                 ##print all 62 options
141                 index2 = final.find("PROGRAM START:")
142                 index3 = final.find("Installed Date:")
143                 if not index2 ==-1 or not index3 ==-1:
144                     decrypt_data += str(i) + ": " + final + "\n"
145         write_enc_infy_data_to_file(decrypt_data,comp_name,FILE_OUTPUT_NAME)
146
147     def read_enc_data_files():
148
149         for root,dir,files in os.walk(PDML_PATH):
150             for file in files:
151                 filename = root+ "\\" + file
152                 if os.path.isfile(filename):
153                     print filename
154                     for line in fileinput.input([filename]):
155                         line = line.strip()
156                         is_found,found_infy_index= found_infy_enc_data(line)
157                         if not is_found:
158                             continue
```

```
159                 line = line[found_infy_index:]
160
161                 #get computer name (for use in init_key() later)
162                 comp_name = extract_comp_name(line)
163                 UNIQUE_COMP.append(comp_name)
164                 #get the infy encrypted data
165                 line = extract_enc_data(line)
166                 #base64 decode enc_data
167                 dec_line = line.decode('base64')
168                 #append enc_data to file
169                 write_enc_infy_data_to_file(dec_line,comp_name,FILE_ENC_OUTPUT_NAME)
170                 enc_wrapper(dec_line,comp_name)
171 try:
172     read_enc_data_files()
173 except:
174     print "exception!!!!"
175
```

Appendix 2 –IoCs

Infy version 31: f07e85143e057ee565c25db2a9f36491102d4e526ffb02c83e580712ec00eb27

Infy “M” version 8.0: 583349B7A2385A1E8DE682A43351798CA113CBBB80686193ECF9A61E6942786A

5.9.94.34
138.201.0.134
138.201.47.150
144.76.250.205
138.201.47.158
138.201.47.153
us1s2[.]strangled[.]net
uvps1[.]cotbm[.]com
gstat[.]strangled[.]net
secup[.]soon[.]it
p208[.]ige[.]es
lu[.]ige[.]es
updateserver1[.]com
updateserver3[.]com
updatebox4[.]com
bestupdateserver[.]com
bestupdateserver2[.]com

bestbox3[.]com
safehostline[.]com
youripinfo[.]com
bestupser[.]awardspace[.]info
box4035[.]net
box4036[.]net
box4037[.]net
box4038[.]net
box4039[.]net
box4040[.]net
box4041[.]net
box4042[.]net
box4043[.]net
box4044[.]net
box4045[.]net
box4046[.]net
box4047[.]net
box4048[.]net
box4049[.]net
box4050[.]net
box4051[.]net
box4052[.]net
box4053[.]net
box4054[.]net
box4055[.]net
box4056[.]net
box4057[.]net
box4058[.]net
box4059[.]net
box4060[.]net
box4061[.]net
box4062[.]net
box4063[.]net
box4064[.]net
box4065[.]net
box4066[.]net
box4067[.]net
box4068[.]net
box4069[.]net
box4070[.]net
box4071[.]net
box4072[.]net
box4075[.]net
box4078[.]net
box4079[.]net
box4080[.]net
box4081[.]net
box4082[.]net
box4083[.]net
box4084[.]net

box4085[.]net
box4086[.]net
box4087[.]net
box4088[.]net
box4089[.]net
box4090[.]net
