Prince of Persia – Game Over

Tomer Bar, Lior Efraim, Simon Conant June 28, 2016

By Tomer Bar, Lior Efraim and Simon Conant

June 28, 2016 at 3:00 PM

Category: Malware, Threat Prevention, Unit 42

Tags: C2, Infy

This post is also available in: 日本語 (Japanese)

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.

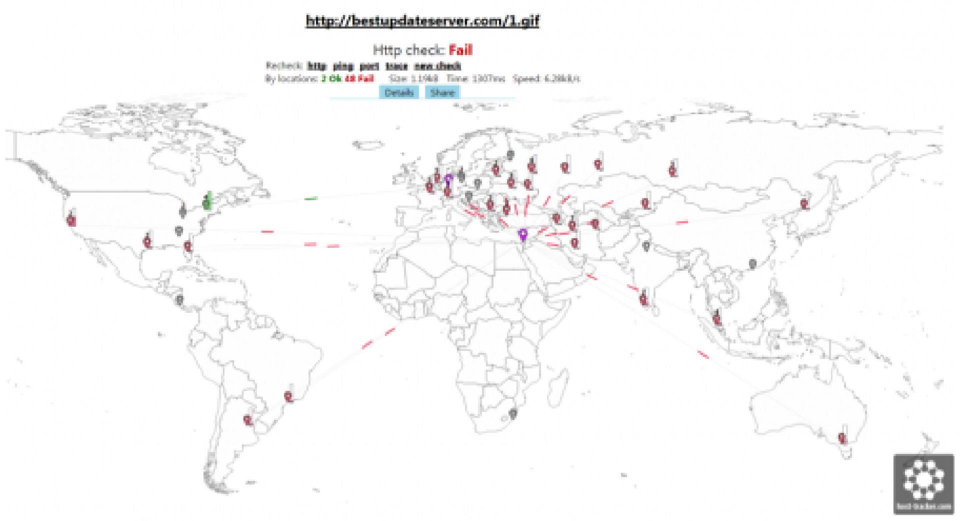
http://bestupdateserver.com/1.gif
Http check: Fail
Recheck: http ping port trace new check
By locations: 2 Ok 48 Fail Size: 1.19kB Time: 1307ms Speed: 6.28kB/s
Details
Share

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.

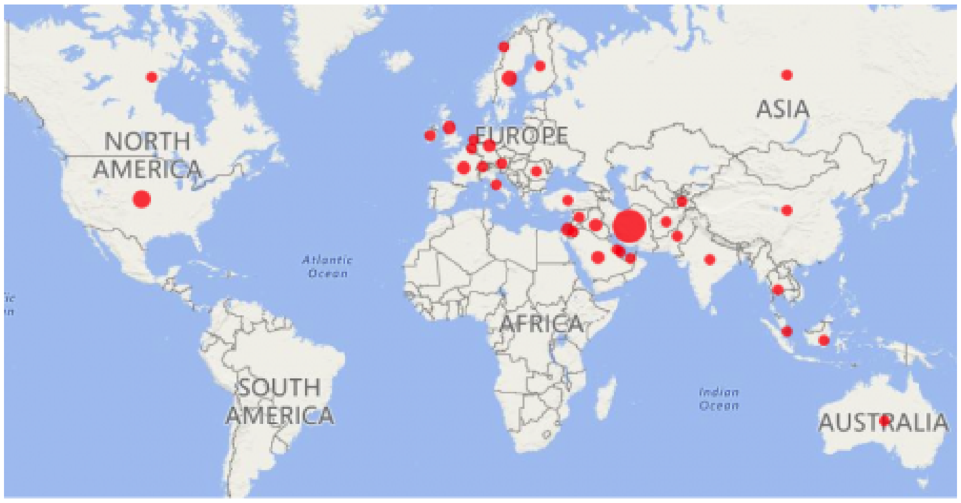

Figure 2 Geographic location of victims. Please note that New Zealand has been omitted from this map only because we observed no victim activity there.

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.

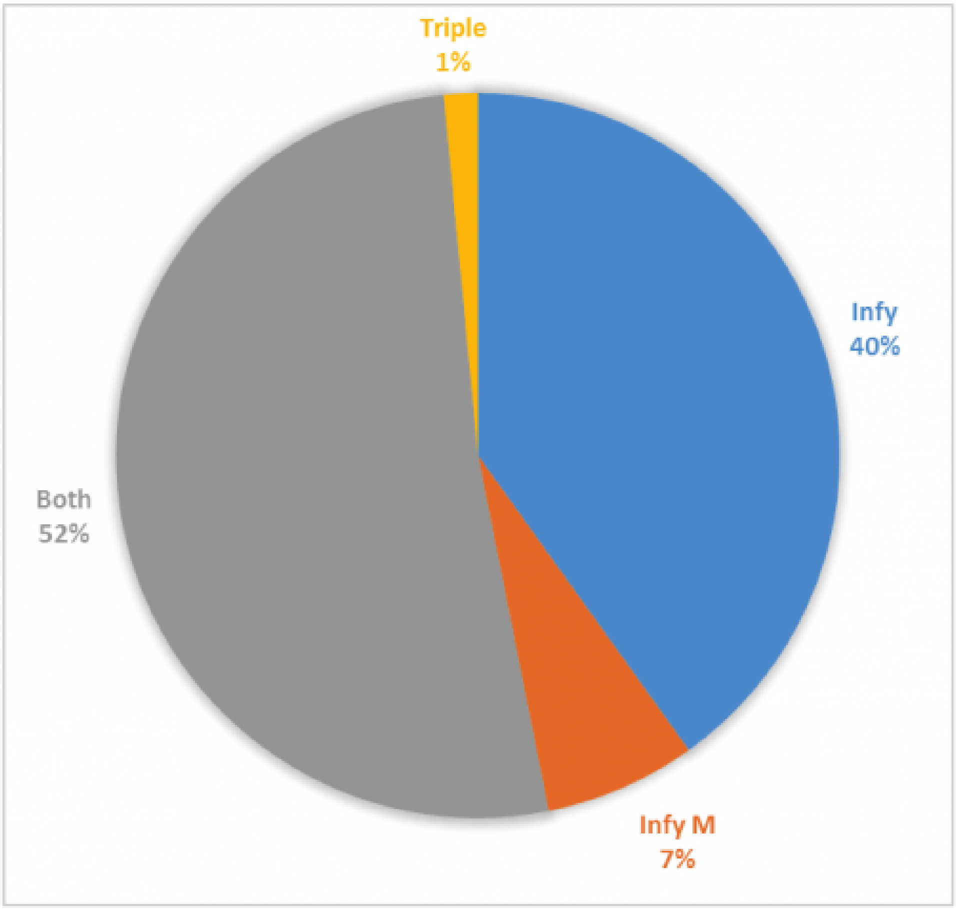

Figure 3 Breakdown of Infy vs. Infy "M" infections

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).

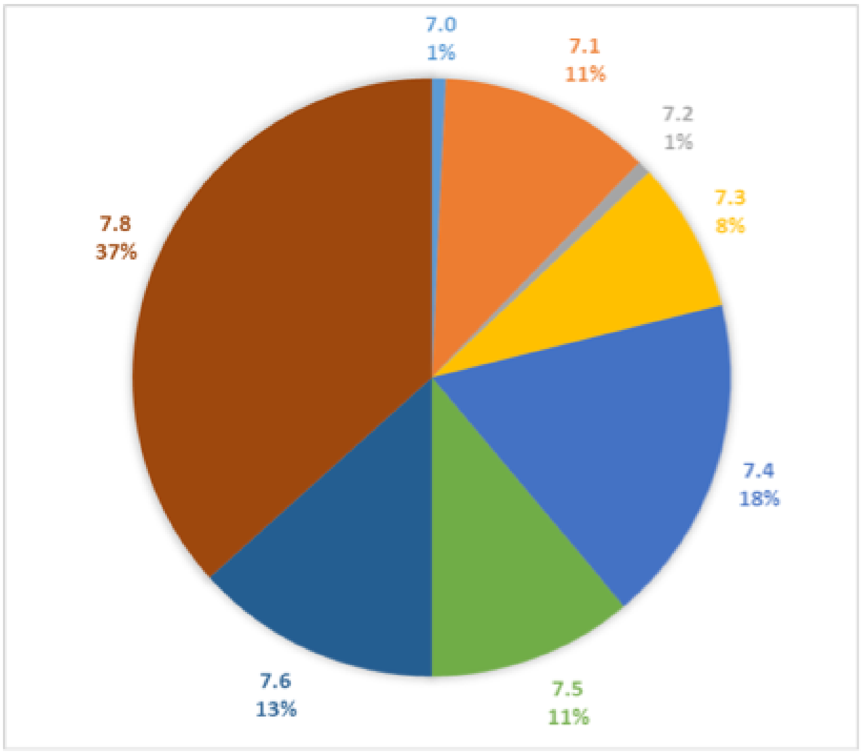

Figure 4 Infy "M" Victim versions

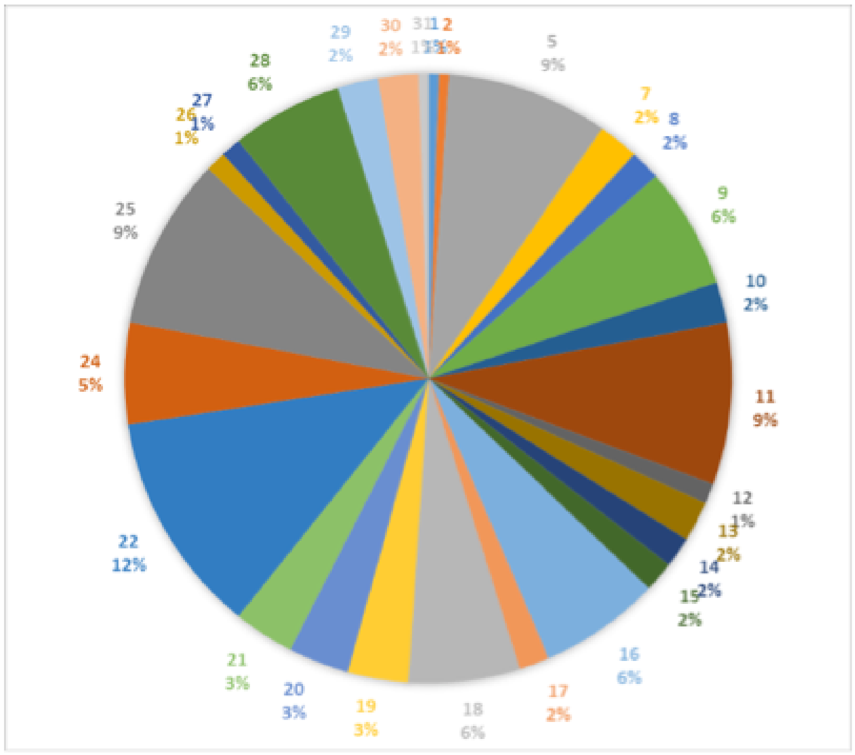

Figure 5 Infy"Original" Victim versions

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:

```
import os,sys
import string
import base64
import fileinput
FIRST_PHASE =
"OQTJEqtsK0AUB9YXMwr8idozF7VWRPpnhNCHI6Dlkaubyxf5423jvcZ1LSGmge"
SECOND_PHASE =
"PqOwI1eUrYtT2yR3p4E5o6WiQu7ASlDkFj8GhHaJ9sKdLfMgNzBx0ZcXvCmVnb"
global FULL_KEY
FULL_KEY= ""
def sub_1_for_hex(str_input):
    str_output = ""
    for letter in str_input:
        try:
            str_output += chr(ord(letter)-1)
        except:
            print "sub_1_for_hex func problem"
            continue
    return str_output

def sum_comp_name(comp_name):
    sum = 0
    for letter in comp_name:
        sum+= ord(letter)
    return sum

def init_key(comp):
    comp_name_sum = sum_comp_name(comp)
    carry = divmod(comp_name_sum, 62)
    index = carry[1] -1
    end_key = FIRST_PHASE[:index]
    key = FIRST_PHASE[index:]
    key = key + end_key
    key = key + key
    return key

```

```
def decrypt(num_list,offset):
    global FULL_KEY
    input = ""
    for num_str in num_list:
        try:
            input += num_str.decode('hex')
        except:
            input += ')'
    result = ""
    for i, c in enumerate(input):
        i = i % 62 +1
        try:
            index = FULL_KEY.index(c)-1
        except ValueError:
            result += c
            continue
        translated = SECOND_PHASE[(index - i +offset) % len(SECOND_PHASE)]
        result += translated
    return result

def found_infy_enc_data(line):
    found_infy_str = "show=\"---------- Administration Reporting Service "
    found_infy_index = line.find(found_infy_str)
    if not found_infy_index==-1:
        return True,found_infy_index
    else:
        return False,found_infy_index
def extract_comp_name(line):
    comp = r"\xd\xa-----"
    comp_index = line.find(comp)
    comp_name = line[comp_index+len(comp):]
    comp_name = comp_name[:comp_name.find("-----")]
    print "(((=)))" + comp_name
    return comp_name

def extract_enc_data(line):
    header = r"\xd\xa_____"
    start_index = line.find(header)+len(header)
    line = line[start_index:]
    endindex = line.index("_____\" value=")
    line = line[:endindex]
    return line

def write_enc_infy_data_to_file(dec_line,comp_name,filename):
    file1 = open(filename + "\\" + comp_name + ".txt",'ab')
    file1.writelines(dec_line)
    file1.close()

def enc_wrapper(enc,comp_name):
    global FULL_KEY
    print FULL_KEY
    FULL_KEY = init_key(comp_name)

    enc_final = ""
    for letter in enc:
            if len(hex(ord(letter))[2:])==1:
                enc_final += "0" + hex(ord(letter))[2:]
            elif len(hex(ord(letter))[2:])==2:
                enc_final += hex(ord(letter))[2:]
            else:
                print "not good hex length"
                exit()

    enc = enc_final.upper()

    enc = enc.replace("2E","21")
    enc = enc.replace("C5DC5A","")
    enc = enc.replace("D03D00","")
    enc = enc.replace("0B0E","2121")

    enc = enc.replace("01","21")

    enc_len = len(enc)

    enc_rev = ""
    num_list = []
    enc_print =""
    for i in range(0,enc_len/2):
        enc_rev = enc[-2:]
        if not enc_rev=="0B" and not enc_rev=="0E" and not enc_rev=="00" and not enc_rev=="D0":
            enc_print +=enc_rev
            num_list.append(enc_rev)
        enc= enc[:-2]

    #the first part is always ok
    dec_str = decrypt(num_list,0)
    final = sub_1_for_hex(dec_str)
    index = final.find("OK: Sent")
    if index==-1:
        print comp_name + " - did not found OK: Sent !!!!\n\n\n\n"
        #exit()
    decrypt_data = comp_name + " ++==++ " +  str(i) + ": " + final + "\n"

    final_start = final[0:500]
    if final_start in UNIQUE_DATA:
          print comp_name + " already have this data"
          return
    UNIQUE_DATA.append(final_start)
    index = final.find("Installed Date:")

    if index==-1:
        for i in range(1,61):
            dec_str = decrypt3(num_list,i)
            final = sub_1_for_hex(dec_str)

            ##print all 62 options
            index2 = final.find("PROGRAM START:")
            index3 = final.find("Installed Date:")
            if not index2 ==-1 or not index3 ==-1:
                decrypt_data += str(i) + ": " + final + "\n"
    write_enc_infy_data_to_file(decrypt_data,comp_name,FILE_OUTPUT_NAME)

def read_enc_data_files():

    for root,dir,files in os.walk(PDML_PATH):
        for file in files:
            filename = root+ "\\" + file
            if os.path.isfile(filename):
                print filename
                for line in fileinput.input([filename]):
                    line = line.strip()
                    is_found,found_infy_index= found_infy_enc_data(line)
                    if not is_found:
                        continue
                    line = line[found_infy_index:]

                    #get computer name (for use in init_key() later)
                    comp_name = extract_comp_name(line)
                    UNIQUE_COMP.append(comp_name)
                    #get the infy encrypted data
                    line = extract_enc_data(line)
                    #base64 decode enc_data
                    dec_line = line.decode('base64')
                    #append enc_data to file
                    write_enc_infy_data_to_file(dec_line,comp_name,FILE_ENC_OUTPUT_NAME)
                    enc_wrapper(dec_line,comp_name)
try:
    read_enc_data_files()
except:
    print "exception!!!!"
```

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