Note

These slides were presented on 12th September 2014 at 44CON.

Much of the content was provided in the talk and is not on the slides, please see the official 44CON recording (when available) or contact us for further information.

nccgroup
freedom from doubt

Automating extraction from malware & recent campaign analysis

44CON 2014 – Breakfast Briefing

nccgroup
freedom from doubt

Who am I?

- **David Cannings @edeca**
  - Cyber Defence Operations at NCC Group
  - Background in network analysis & reverse engineering.
- **Day job involves managing incidents and ensuring we've got the tools & technology required to help our clients.**
- **Special thanks to Luke @NullMode_ for his collaboration.**

CDO provides timely analysis, advice and supporting services to clients wishing to protect, detect and remediate cyber-attacks.

nccgroup
freedom from doubt

Today's talk

- An overview of what we need to extract from malware.
- Some information about different approaches.
- A look at some recent targeted attack campaigns.
- Aimed at entry level reverse engineers - please enjoy your breakfast if you are already a malware ninja (it's too early for kernel mode debugging!).

Background

nccgroup
freedom from doubt

- **We have a lot of malware samples.**
- **We want to extract useful information from them.**
- **We don't want to do this manually every time!**
  - Automating basic tasks gives us time for fun stuff ☺
- **How can we do this flexibly & efficiently at large scale?**

What is useful information?

nccgroup
freedom from doubt

- **Any "indicators of compromise":**

"an artifact observed on a network or in operating system that with high confidence indicates a computer intrusion"
(Wikipedia)

- **Things we might find on an infected host**
  - Filenames, registry keys, mutex values, named pipes, etc.
- **Things that might be present in the network capture**
  - Domain names, user agents, request parameters, etc.

Summary: Technical data that we can go hunt for elsewhere.

How about strings?

nccgroup
freedom from doubt

```
sub_4017E0 proc near

var_2C= dword ptr -2Ch
var_28= dword ptr -28h
Tm= tm ptr -24h

sub     esp, 2Ch
push    edi
push    offset aWww_google_com ; "www.google.com"
lea     ecx, [esi+2A4h] ; this
mov     dword ptr [esi+74h], 0FFFFFFh
mov     dword ptr [esi+78h], 0FF00FFh
call    ?SetWindowTextA@CWnd@@QAEXPBD@Z ; CWnd::SetWindowTextA(char const *)
push    offset a80      ; "80"
lea     ecx, [esi+2F8h] ; this
```

www.google.com – oh no! It's malicious!

How about strings?

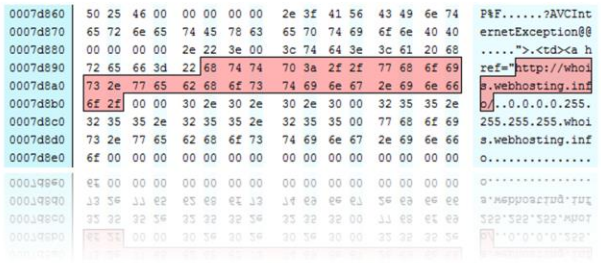
nccgroup
freedom from doubt

```
pAutoProxyOptions= WINHTTP_AUTOPROXY_OPTIONS ptr -34h
pProxyConfig= WINHTTP_CURRENT_USER_IE_PROXY_CONFIG ptr -1Ch
pProxyInfo= WINHTTP_PROXY_INFO ptr -0Ch

push    ebp
mov     ebp, esp
sub     esp, 330h
push    esi
push    edi
mov     ecx, 9
mov     esi, offset aWww_google_com ; "www.google.com.tw"
lea     edi, [ebp+cwszUrl]
rep movsd
```

www.google.com.tw – super malicious!

(this is actually a sample of dyncalc)

Let's do regex!



```
0007d860  50 25 46 00  00 00 00 00  2e 3f 41 56  43 49 6e 74  P%F......?AVCInt
0007d870  65 72 6e 65  74 45 78 63  65 70 74 69  6f 6e 40 40  ernetException@@
0007d880  00 00 00 00  2e 22 3e 00  3c 74 64 3e  3c 61 20 68  .....">.<td><a h
0007d890  72 65 66 3d  22 68 74 74  70 3a 2f 2f  77 68 6f 69  ref="http://whoi
0007d8a0  73 2e 77 65  62 68 6f 73  74 69 6e 67  2e 69 6e 66  s.webhosting.inf
0007d8b0  6f 2f 00 00  30 2e 30 2e  30 2e 30 00  32 35 35 2e  o/..0.0.0.0.255.
0007d8c0  32 35 35 2e  32 35 35 2e  32 35 35 00  77 68 6f 69  255.255.255.whoi
0007d8d0  73 2e 77 65  62 68 6f 73  74 69 6e 67  2e 69 6e 66  s.webhosting.inf
0007d8e0  6f 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  o...............
```

whois.webhosting.info? Oh dear…

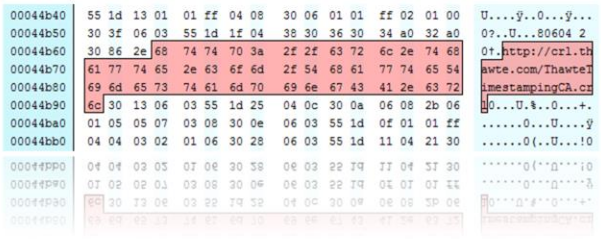
Also note the highly suspicious 0.0.0.0 and 255.255.255.255, these should be filtered everywhere! ☺

Don’t forget percent encoding..

```
xor     edx, edx
mov     dx, [esi+4]
mov     [esp+20h+var_4], 1
push    edx
push    eax             ; in
call    ds:inet_ntoa
push    eax
lea     eax, [esp+28h+arg_4]
push    offset aHttpSD  ; "Http://%s:%d/"
push    eax
```

http://%s:%d/ - How did those characters get there?

Who needs CRLs?



```
00044b40  55 1d 13 01  01 ff 04 08   30 06 01 01  ff 02 01 00   U....ÿ..0...ÿ...
00044b50  30 3f 06 03  55 1d 1f 04   38 30 36 30  34 a0 32 a0   0?..U...80604 2
00044b60  30 86 2e 68  74 74 70 3a   2f 2f 63 72  6c 2e 74 68   0†.http://crl.th
00044b70  61 77 74 65  2e 63 6f 6d   2f 54 68 61  77 74 65 54   awte.com/ThawteT
00044b80  69 6d 65 73  74 61 6d 70   69 6e 67 43  41 2e 63 72   imestampingCA.cr
00044b90  6c 30 13 06  03 55 1d 25   04 0c 30 0a  06 08 2b 06   l0...U.%..0...+.
00044ba0  01 05 05 07  03 08 30 0e   06 03 55 1d  0f 01 01 ff   ......0...U....ÿ
00044bb0  04 04 03 02  01 06 30 28   06 03 55 1d  11 04 21 30   ......0(..U...!0
```

It's fine, Thawte won't mind!

It's getting silly now…

nccgroup
freedom from doubt

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(
?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[
\t]))*"(?:(?:\r\n)?[ \t])*)*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\0
31]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t]))(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:
(?:\r\n)?[ \t]))*)*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)
?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\
r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
 \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t]
)*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
 \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*
)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t]))*)*
*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+
|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r
\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:
\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t
]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031
]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](
?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?
:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?
:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?
```

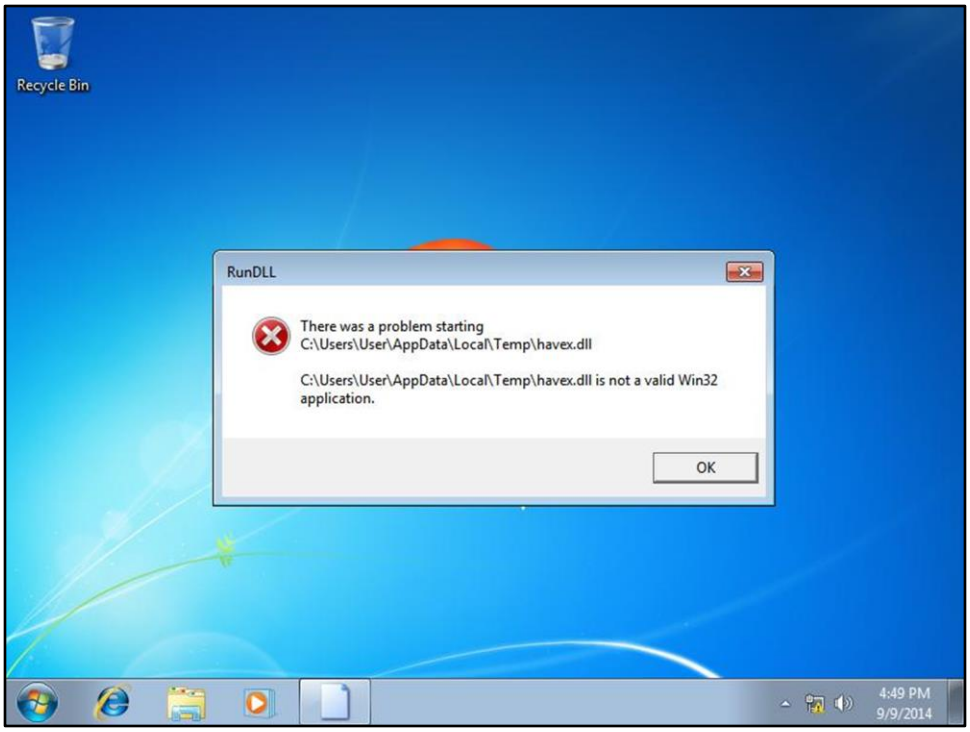
http://ex-parrot.com/~pdw/Mail-RFC822-Address.html

This is a small portion (!) of a regular expression from a Perl module.

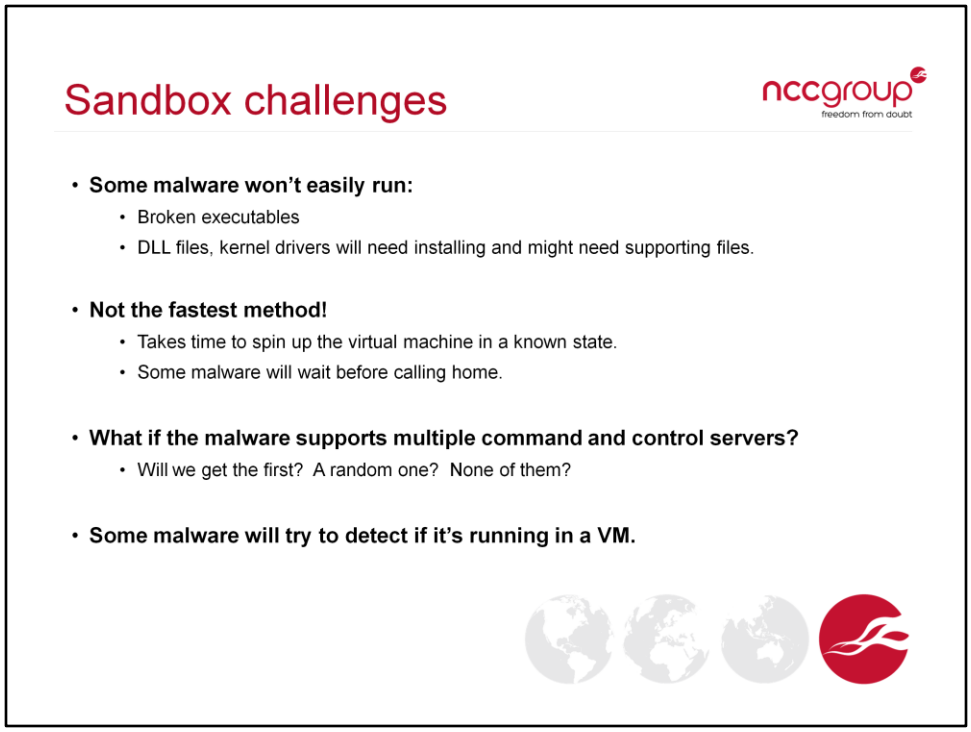
Summary

- **Blind extraction of IP addresses or domains is not good enough**

How about a sandbox?



This is a random Havex DLL, thrown into Cuckoo with no proper configuration or entry point set.



Sandboxes are great as a first quick analysis, BUT….

We might need to know the correct export to load a DLL, kernel drivers may need a usermode component. A lot of malware is dropped with a configuration file, if we don't have this then it won't necessarily run.

nccgroup
freedom from doubt

Let's improve / add to these techniques…

Aims

nccgroup
freedom from doubt

- **Needs to be fast**
  - Near to real-time, to support large amounts of data.
- **Should understand the trojans we've analysed recently**
  - Havex (ENERGETIC BEAR).
  - Sayad, MiniSayad (FLYING KITTEN).
  - Plus some that aren't currently public ☺
- **Go further than just basic indicators of compromise**
  - Find encryption keys to decode network traffic.
  - Extract plugins, second stage implants etc.
- **Add value to what Cuckoo and VirusTotal already do well.**

Similar things

nccgroup
freedom from doubt

- **Volatility plugins**
  - Requires running the sample, which is slow (and some are broken).
  - However - sometimes memory analysis is the easiest / only way.
- **Malware Config Extractor -** http://malwareconfig.com/
  - Very good parsing of some remote access trojans.
  - No API? Can't really parse the output ☹
  - Sometimes it takes a long time or doesn't work.
- **Lots of code (Perl, Python, C) scattered around for various malware.**

Challenges

nccgroup
freedom from doubt

- **Identifying the malware**
  - How can we quickly decide if something is a piece of malware we know?
- **Locating the parts that are of interest**
  - Where is the actual data we care about?
- **Extracting useful things from it**
  - How do we implement decoding, decryption etc. in a flexible way?
  - How can we output this in a format which is usable?

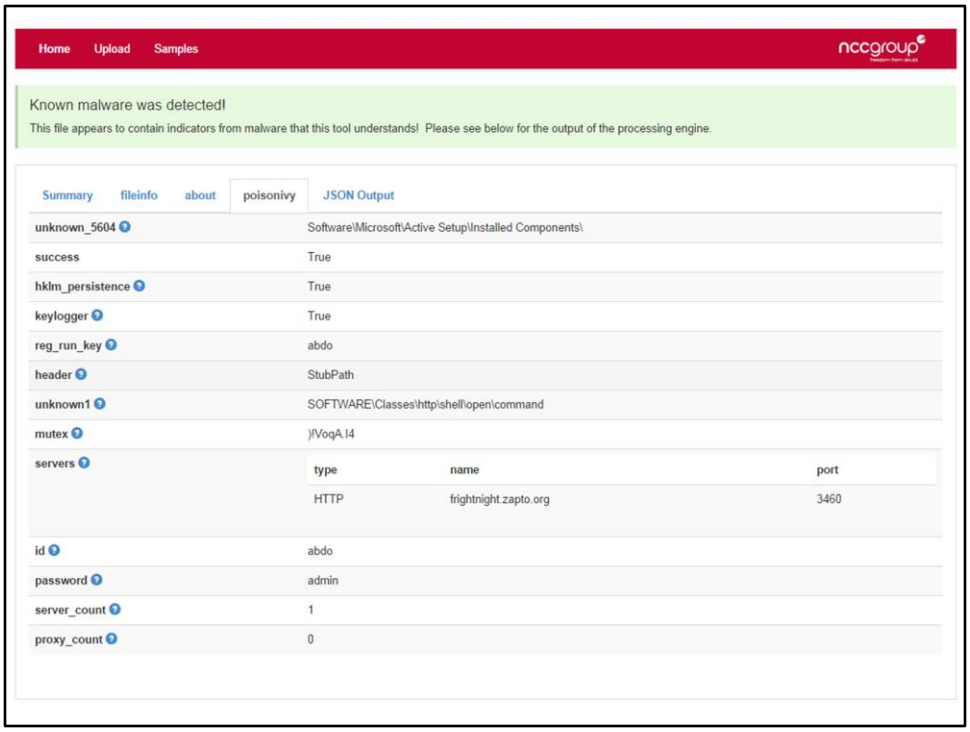
What tools can help us?

nccgroup
freedom from doubt

- If you take one thing away from this presentation, choose this…

http://pythonarsenal.erpscan.com/

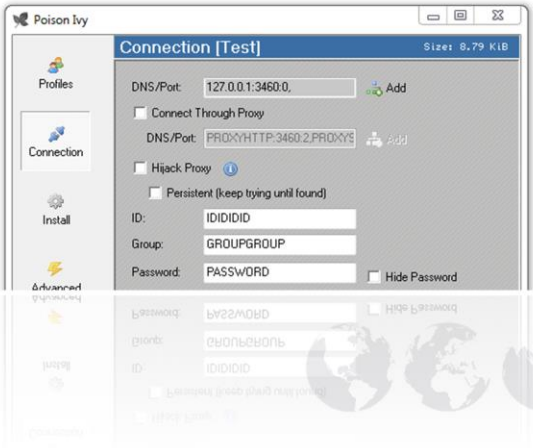
- We could automate IDA, but it's slow because of the depth the auto analyser goes to (and expensive!).
- Instead we have used Yara, pefile and Capstone (all from within Python).



Thanks to Luke!

Poison Ivy

- **Off-the-shelf RAT, available for years.**
  - Certainly not exclusively Cyber Cyber APT, but it has been used in targeted campaigns.



```
rule claim {
  meta:
    description = "Generic rule to match Poison Ivy samples"

  strings:
    // Start of config marker
    $config = { 0F 04 08 00 53 74 75 62 50 61 74 68 }

    // This is common in many non-obfuscated, non-packed Poison Ivy samples.
    // It will at least catch low hanging fruit.
    /*
     0  b8 00 08 40 00  mov eax, 0x400800
     5  ff d0           call eax
     7  6a 00           push 0
     9  e8 00 00 00 00  call 0xe
    */
    $ep = { B8 00 ?? 40 00 FF D0 6A 00 E8 00 00 00 00 }

  condition:
    any of them
}
```

Introduce why Yara is so good at this. Multi-threaded, decent engine written in C with bindings for Python.

Poison Ivy

nccgroup
freedom from doubt

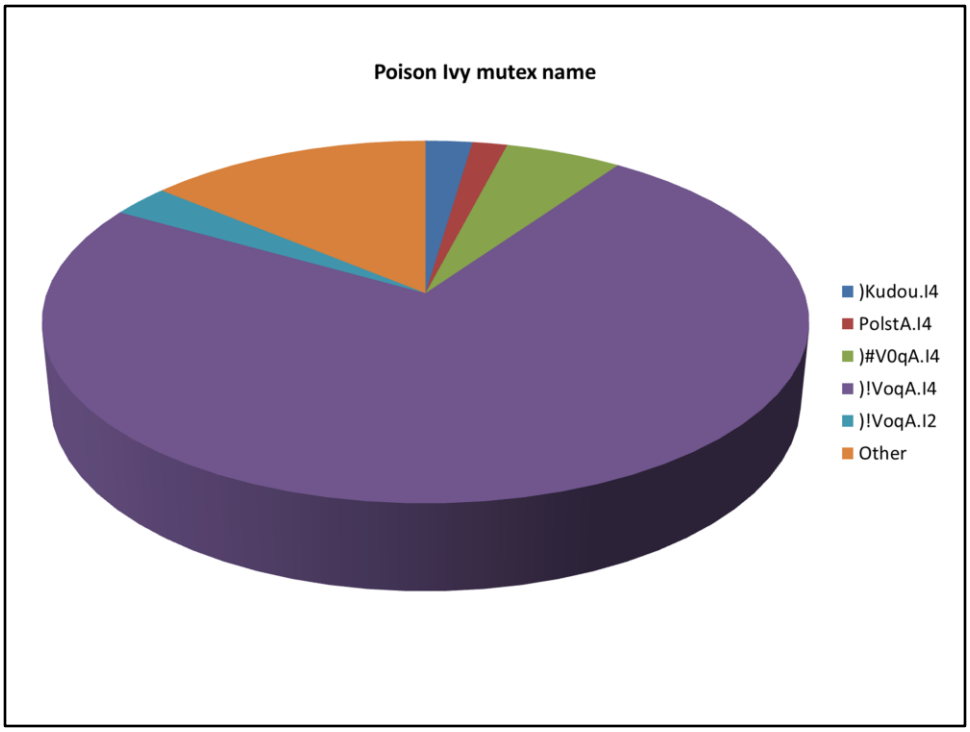
- **Configuration is very nicely structured.**
- **Best way to work it out is to play with it!**
  - Generate a number of samples, use known values to speed up the process (e.g. set the mutex to MUTEXMUTEX123, then look for this).

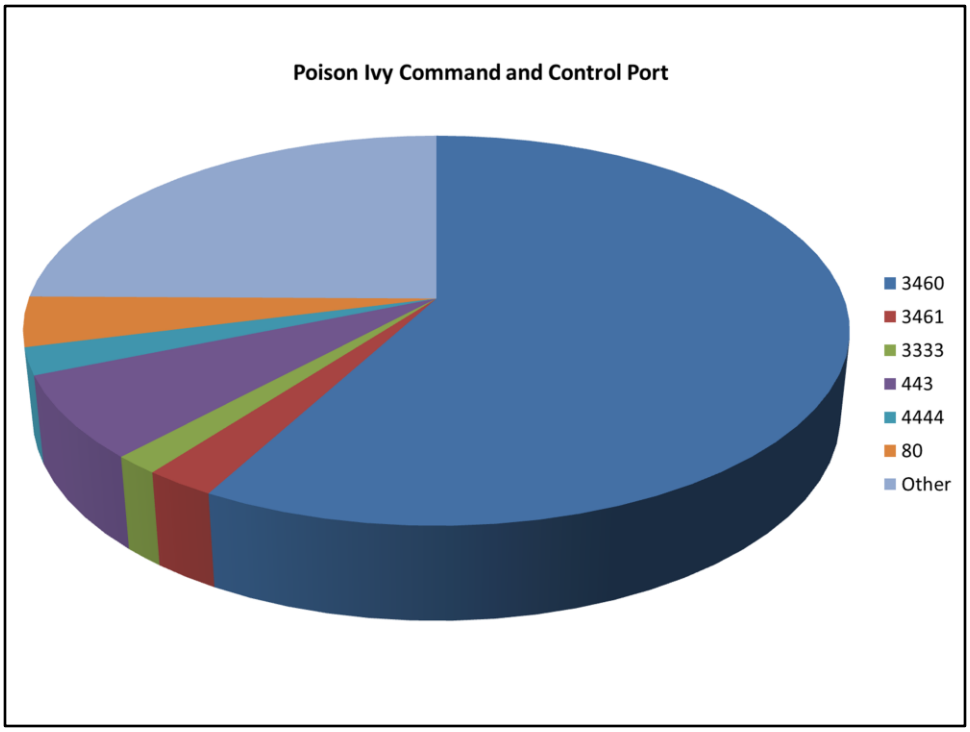
```
00001820  04 08 00 53  74 75 62 50  61 74 68 18  04 28 00 53  ...StubPath..(.S
00001830  4f 46 54 57  41 52 45 5c  43 6c 61 73  73 65 73 5c  OFTWARE\Classes\
00001840  68 74 74 70  5c 73 68 65  6c 6c 5c 6f  70 65 6e 5c  http\shell\open\
00001850  63 6f 6d 6d  61 6e 64 56  04 35 00 53  6f 66 74 77  commandV.5.Softw
00001860  61 72 65 5c  4d 69 63 72  6f 73 6f 66  74 5c 41 63  are\Microsoft\Ac
00001870  74 69 76 65  20 53 65 74  75 70 5c 49  6e 73 74 61  tive Setup\Insta
00001880  6c 6c 65 64  20 43 6f 6d  70 6f 6e 65  6e 74 73 5c  lled Components\
00001890  fa 0a 0b 00  50 6f 6b 65  6d 6f 6e 59  6f 6c 6f 90  ú...PokemonYolo
000018a0  01 11 00 0d  32 35 2e 31  33 34 2e 31  30 35 2e 39  ....25.134.105.9
000018b0  35 00 84 0d  8c 01 04 00  00 00 00 00  c1 02 04 00  5.„.Œ.......Á...
000018c0  ff ff ff ff  45 01 05 00  61 64 6d 69  6e fb 03 09  ÿÿÿÿE...adminû..
000018d0  00 29 21 56  6f 71 41 2e  49 34 fa 03  01 00 01 00  .)!VoqA.I4ú.....
000018e0  00 d5 00 c5  00 55 8b ec  8b 75 08 80  be f7 03 00  .Õ.Å.U‹ì‹u.€¾÷..
```

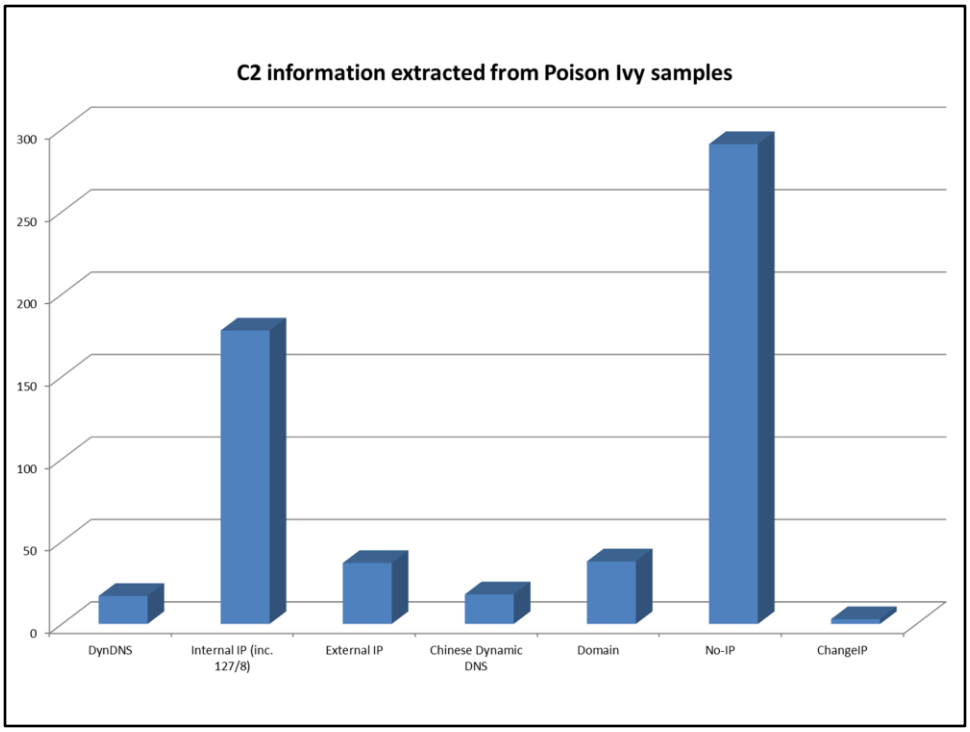
(or cheat & see: http://blog.conixsecurity.fr/wp-content/uploads/2013/10/Poison-Ivy-RAT-conf-comms.pdf for an excellent paper!)

Compare and contrast with targeted samples that FireEye wrote about (our samples are a random set, **not** targeted). Are they very different?

Remember that the purpose of this is to automate the boring parts, not replace a reverse engineer. Let's do 80% by magic, then let skilled analysts loose on the 20%.


Poison Ivy mutex name
)Kudou.I4
PolstA.I4
)#V0qA.I4
)!VoqA.I4
)!VoqA.I2
Other


Poison Ivy Command and Control Port
3460
3461
3333
443
4444
80
Other


C2 information extracted from Poison Ivy samples
300
250
200
150
100
50
0
DynDNS
Internal IP (inc. 127/8)
External IP
Chinese Dynamic DNS
Domain
No-IP
ChangeIP

nccgroup
freedom from doubt

How about cyber cyber APT?

Havex: Our first sample

nccgroup
freedom from doubt

Extracting the payload from a CVE-2014-1761 RTF document

Monday June 9, 2014

Background

In March Microsoft published **security advisory 2953095**, detailing a remote code execution vulnerability in multiple versions of Microsoft Office (CVE-2014-1761). A **Technet blog** was released at the same time which contained excellent information on how a typical malicious document would be constructed.

NCC Group's Cyber Defence Operations team used the information in the Technet blog to identify a malicious document within our malware zoo that exploited this vulnerability which appears to have been used in a targeted attack. In this blog we show one method of analysing the shellcode manually to extract the payload.

Matching the malicious document

The Technet blog gives a number of pointers toward a malicious document. First there is a bad header at the beginning of the document, which should be {\rtf in a real document but is {\rt{. Our sample matches this:

```
00000000  7b 5c 72 74 7b 7b 5c 69  6e 66 6f 7b 5c  {\rt{{\info{\
```

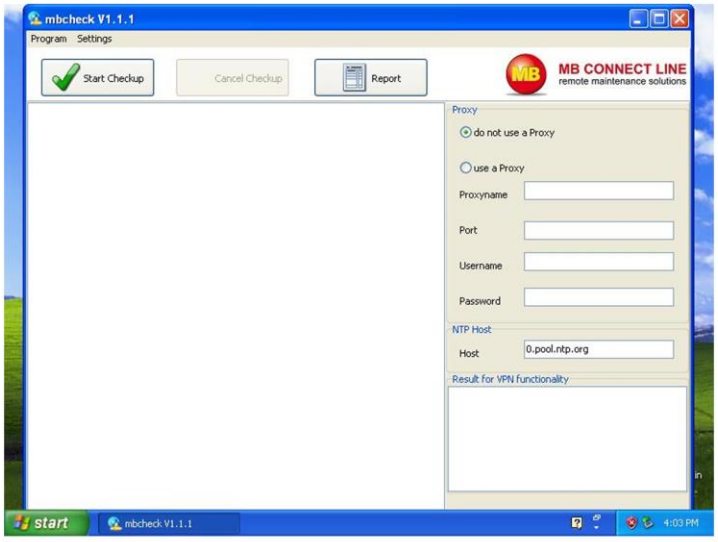
See: https://www.nccgroup.com/en/blog/2014/06/extracting-the-payload-from-a-cve-2014-1761-rtf-document/

Havex: Identifying more samples

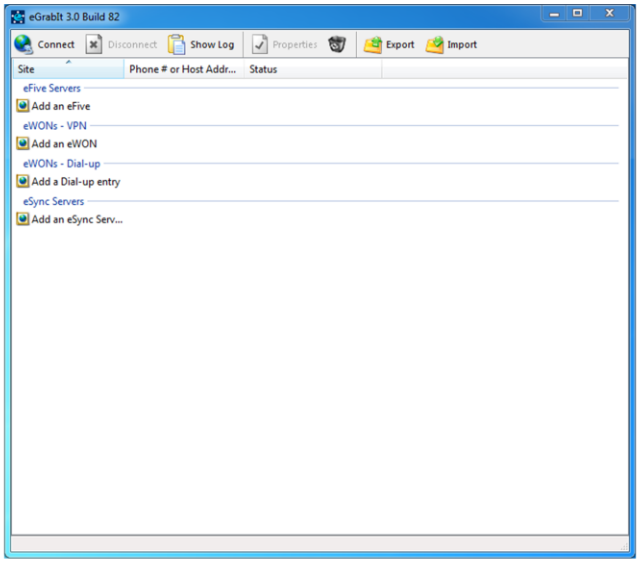

```
rule claim {
  meta:
    description = "Matches generic strings from the havex RAT family"

  strings:
    $xor_key1 = "MTMxMjMxMg==" wide ascii

    $generic_bzip = "This is a bug in bzip2/libbzip2, %s."
    $generic_bzip_ver = "1.0.6, 6-Sept-2010"
    $generic_copyright = "Copyright (c) 1992-2004 by P.J. Plauger, licensed by
Dinkumware, Ltd"

  condition:
    $xor_key1 and 1 of ($generic*)
}
```

This is a **very basic rule** for havex. It is not a good rule for finding new samples, but it's OK at detecting the main implant component. It's not the only rule used within the MICE framework.

Havex: Distribution



Havex: Distribution



nccgroup
freedom from doubt

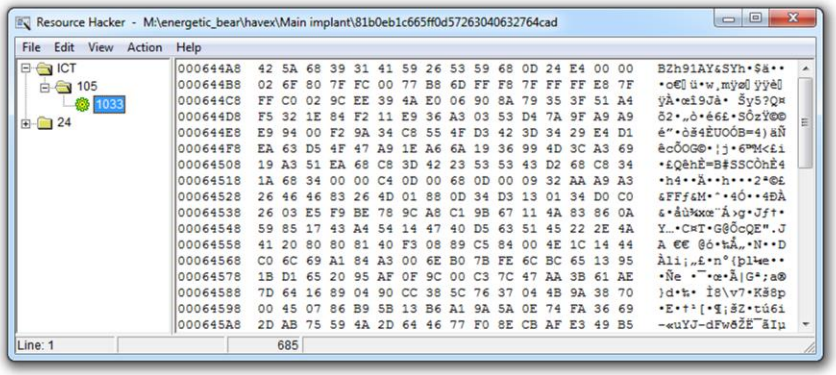
So where's the config?

Havex: early versions

nccgroup
freedom from doubt

```
.rdata:1004E447                 align 4
.rdata:1004E448 aPekanin_freeva:                        ; DATA XREF: sub_10001012+23↑o
.rdata:1004E448                 unicode 0, <pekanin.freevar.com/include/template/isx.php?id=>,0
.rdata:1004E4AA                 align 10h
.rdata:1004E4B0 aRandallweil_co:                        ; DATA XREF: sub_10001012+4F↑o
.rdata:1004E4B0                 unicode 0, <randallweil.com/cms/tinymce/examples/access.php?id=>,0
.rdata:1004E518 aShwandukani_ue:                        ; DATA XREF: sub_10001012+7C↑o
.rdata:1004E518                 unicode 0, <shwandukani.ueuo.com/modules/mod_search/mod_research.php?>
.rdata:1004E518                 unicode 0, <id=>,0
```

Havex: later versions



02:00

Havex: later versions

nccgroup
freedom from doubt

00000000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000000	00	33	03	32	39	31	7f	31	67	31	7f	33	49	32	7c	33	.3.291▯1g1▯3I2|3
00000010	5b	32	7e	31	4a	31	7e	31	55	33	0c	32	0c	33	3b	32	[2~1J1~1U3.2.3;2
00000020	06	31	38	31	5b	31	53	33	47	32	54	33	49	32	39	31	.181[1S3G2T3I291
00000030	03	31	07	31	06	33	01	32	01	33	01	32	03	31	02	31	.1.1.3.2.3.2.1.1
00000040	39	31	03	33	03	32	3b	33	74	32	4b	31	42	31	5f	31	91.3.2;3t2K1B1_1
00000050	5d	33	43	32	54	33	43	32	1d	31	77	31	6b	31	77	33	]3C2T3C2.1w1k1w3
00000060	3b	32	01	33	3b	32	01	31	38	31	05	31	04	33	3b	32	;2.3;2.181.1.3;2
00000070	42	33	58	32	5d	31	54	31	46	31	5e	33	52	32	54	33	B3X2]1T1F1^3R2T3
00000080	5d	32	56	31	50	31	40	31	1c	33	57	32	43	33	54	32	]2V1P1@1.3W2C3T2
00000090	56	31	41	31	56	31	4a	33	48	32	52	33	5e	32	5e	31	V1A1V1J3H2R3^2^1
000000a0	5b	31	50	31	41	33	1f	32	52	33	5e	32	5e	31	1d	31	[1P1A3.2R3^2^1.1
000000b0	44	31	42	33	01	32	04	33	1e	32	44	31	42	31	1e	31	D1B3.2.3.2D1B1.1
000000c0	53	33	55	32	5c	33	58	32	5d	31	1d	31	5a	31	5c	33	S3U2\3X2]1.1Z1\3

04:00

Havex: later versions

nccgroup
freedom from doubt

00000000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000000	31	00	32	00	0a	00	4d	00	54	00	4d	00	78	00	4d	00	1.2...M.T.M.x.M.
00000010	6a	00	4d	00	78	00	4d	00	67	00	3d	00	3d	00	0a	00	j.M.x.M.g.=.=...
00000020	35	00	0a	00	68	00	61	00	76	00	65	00	78	00	0a	00	5...h.a.v.e.x...
00000030	31	00	34	00	34	00	30	00	30	00	30	00	30	00	30	00	1.4.4.0.0.0.0.0.
00000040	0a	00	31	00	32	00	0a	00	45	00	78	00	70	00	6c	00	..1.2...E.x.p.l.
00000050	6f	00	72	00	65	00	72	00	2e	00	45	00	58	00	45	00	o.r.e.r...E.X.E.
00000060	0a	00	30	00	0a	00	32	00	0a	00	36	00	36	00	0a	00	..0...2...6.6...
00000070	73	00	69	00	6e	00	66	00	75	00	6c	00	63	00	65	00	s.i.n.f.u.l.c.e.
00000080	6c	00	65	00	62	00	73	00	2e	00	66	00	72	00	65	00	l.e.b.s...f.r.e.
00000090	65	00	73	00	65	00	78	00	79	00	63	00	6f	00	6d	00	e.s.e.x.y.c.o.m.
000000a0	69	00	63	00	73	00	2e	00	63	00	6f	00	6d	00	2f	00	i.c.s...c.o.m./.
000000b0	77	00	70	00	30	00	35	00	2f	00	77	00	70	00	2d	00	w.p.0.5./.w.p.-.
000000c0	61	00	64	00	6d	00	69	00	6e	00	2f	00	69	00	6e	00	a.d.m.i.n./.i.n.

04:30

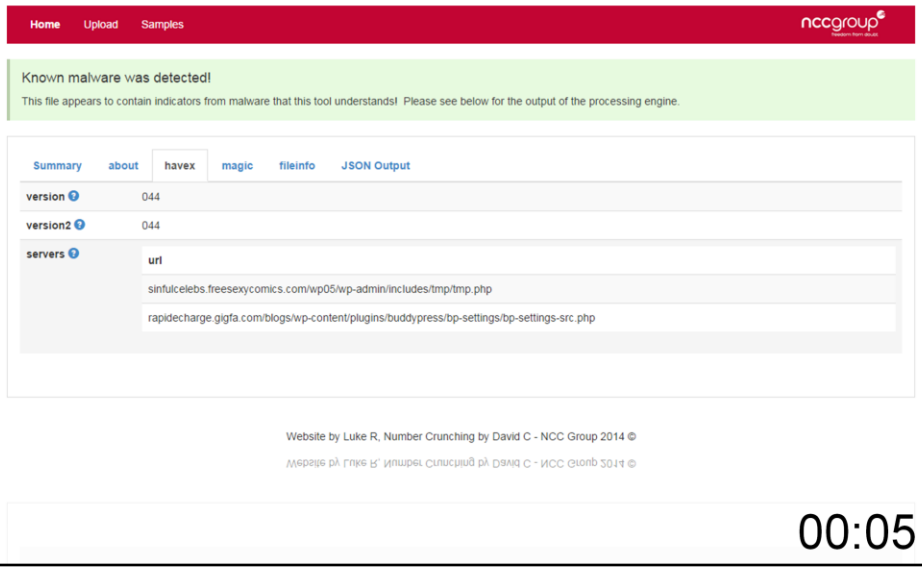
Havex: later versions

nccgroup
freedom from doubt

```
12
MTMxMjMxMg==
5
havex
14400000
12
Explorer.EXE
0
2
66
sinfulcelebs.freesexycomics.com/wp05/wp-admin/includes/tmp/tmp.php
90
rapidecharge.gigfa.com/blogs/wp-content/plugins/buddypress/bp-settings/bp-
settings-src.php
354
AATXn+MiwLu+xCoMG7SqY1uQxAk1qLdyoED9LxIVQr2Z/gsrHIsgTvK9AusdFo+9
fzAxf1zXj42880+kUmktmVb5HSYi8T27Q54eQ4ZLUFKPKZstgHcwPVHGdwpmmRmk
09fL3KGd9SqR60Mv7QtJ4VwGDqrzOja+Ml4SI7e60C4qDQAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAQAB
(snip..)
```

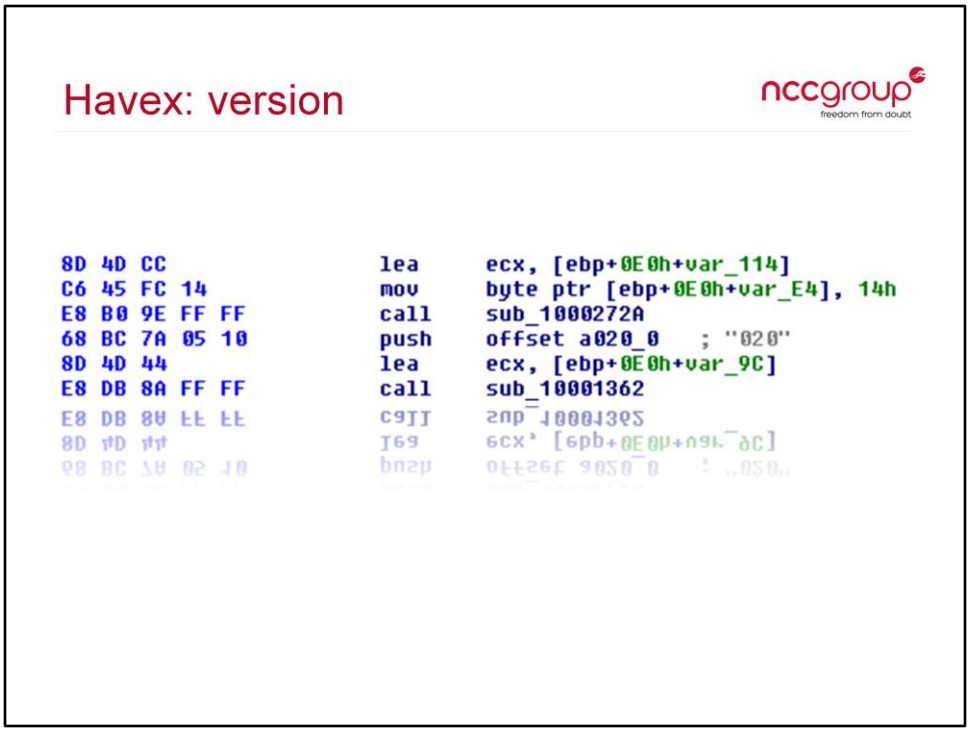
05:00

Let's automate it..



00:05

Havex: version

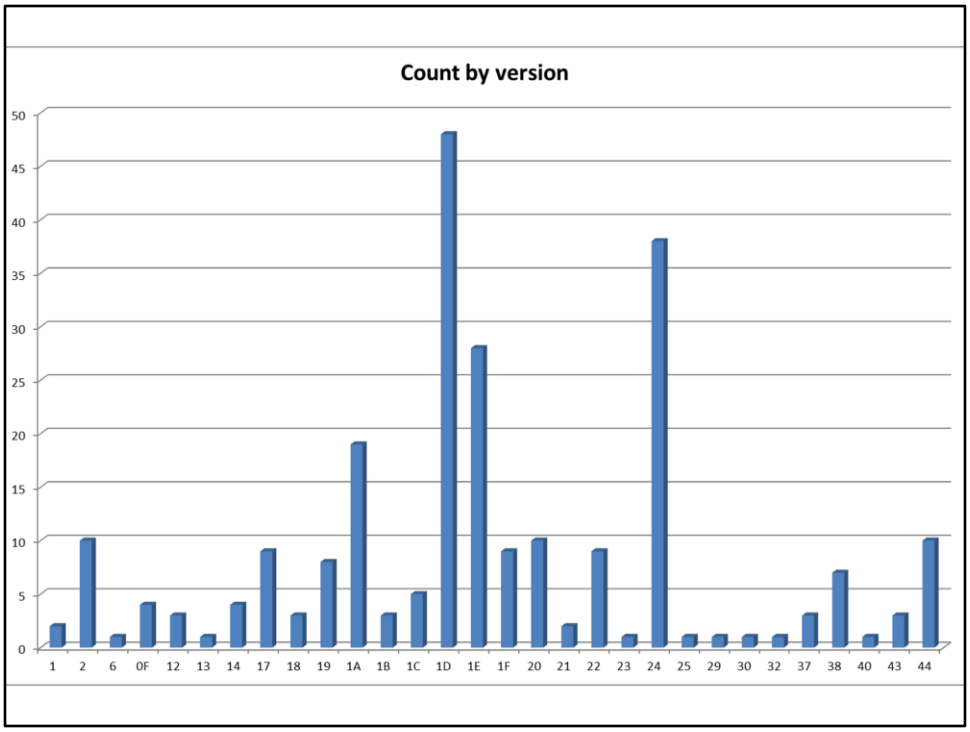


It's also possible to extract this from various strings, but this is a nice example of using Yara, pefile and, optionally, Capstone.

Havex: version

nccgroup
freedom from doubt

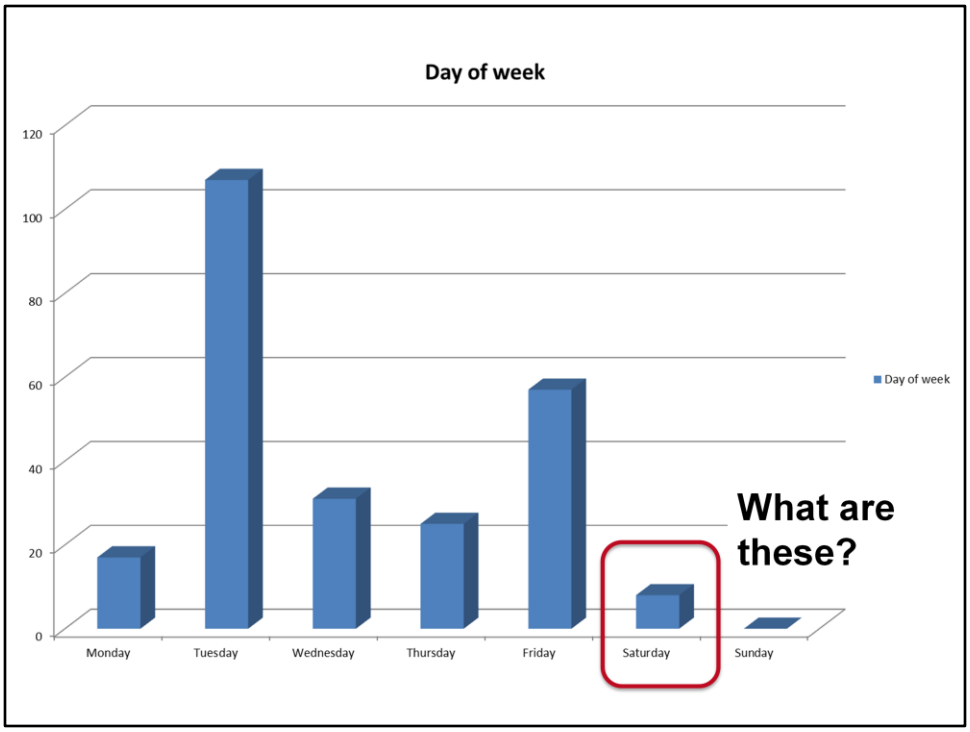
```
rule version {
    meta:
        description = "Finds CALL/PUSH/LEA sequence, iterate results to find
version"

    strings:
        $version = { E8 ?? ?? FF FF 68 ?? ?? ?? ?? [0-1] 8D [2-5] }

    condition:
        any of them
}
```


Count by version
50
45
40
35
30
25
20
15
10
5
0
1
2
6
0F
12
13
14
17
18
19
1A
1B
1C
1D
1E
1F
20
21
22
23
24
25
29
30
32
37
38
40
43
44

Recent campaigns

nccgroup
freedom from doubt

- **What's the oldest sample?**
  - 2nd June 2011
- **What's the newest sample?**
  - 7th May 2014
- **How many unique beacon URLs are there?**
  - 167 - including some that only appear once (why is that?)


Day of week
120
100
80
60
40
20
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of week
What are these?

Unusual dates

nccgroup
freedom from doubt

- **Saturday December 3rd 2011**
  - Russian Parliamentary Elections held on Sunday 4th December 2011.
- **Saturday June 9th 2012**
  - Russian public holiday on Monday June 11th – compensated by June 9th.
  - BP forced to sell 50% stake in Joint Venture to Russian oligarchs.

(Disclaimer: Plenty of other things happened in the world on these days, but these 8 samples potentially relate to interesting stuff)

Next steps

nccgroup
freedom from doubt

- **Create some Maltego transforms**
- **Take the output and automagically create indicators of compromise (e.g. OpenIOC format)**
- **Integrate with Volatility support to assist with difficult malware.**
  - Makes some unpacking tasks much more simple.
- **Interested? See us on the NCC Group stand today!**

nccgroup
freedom from doubt

Questions?

nccgroup
freedom from doubt

UK Offices

Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Milton Keynes

European Offices

Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland

North American Offices

San Francisco
Atlanta
New York
Seattle

Australian Offices

Sydney