F-Secure | LABS

ATTACK DETECTION FUNDAMENTALS: INITIAL ACCESS

Riccardo Ancarani

Alfie Champion

F-Secure | LABS

```
C:\> whoami /all
```

- **Riccardo Ancarani** - Security Consultant, Active Directory Security, @dottor_morte
- **Alfie Champion** - Security Consultant, Global Attack Detection Lead, @ajpc500

F-Secure | LABS

Goals of this series:

- Help improve understanding of attacks, so we can ***detect*** and ***prevent*** them
- Demonstrate attack detection fundamentals and understand ***how enterprise products work*** under the hood

F-Secure | LABS

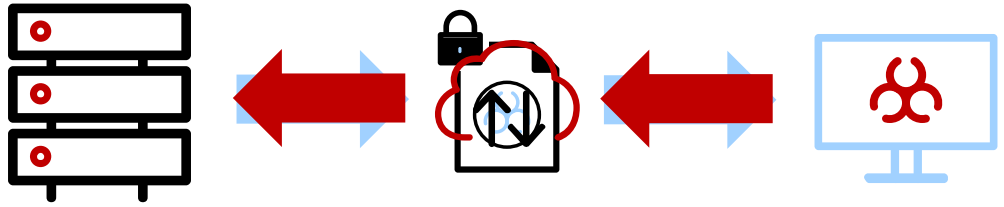
How are we going to do that:

- ***Analyse*** set of known TTPs used by real threat actors
- ***Emulate*** them in a controlled lab environment
- ***Observe*** the traces that they leave

F-Secure | LABS

Before jumping into the action...

- Simple lab setup
- Open-source offensive and defensive tools
- Accompanying lab scripts



F-Secure | LABS

Episode 1 is all about **Initial Access**. But what exactly is Initial Access?

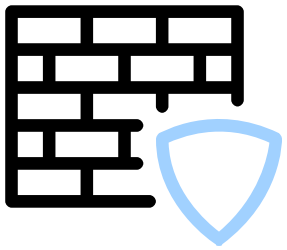
"Initial Access is the set of tactics, techniques and procedures used by malicious actors to obtain a foothold in the target environment"

Examples of Initial Access techniques could be ***spear phishing*** or the ***compromise of an internet-facing web application*** linked to the internal network.

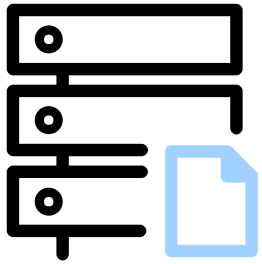
https://attack.mitre.org/tactics/TA0001/

F-Secure | LABS

All the security events will fall into four main categories of log sources:



Networking
- Firewall
- Web Proxy
- Sysmon EID 3



Endpoint Logs
- Windows Event Log
- Sysmon

EDR/AV
- Defender
- AMSI
- <EDR Product Here>

Host Memory
- Raw Memory Access

F-Secure | LABS

For each log type we just described, we're going to analyse a representative technique that can be detected using that specific source.

Lab 1: Maldoc that uses PowerShell to establish a C2

Lab 2: Malicious HTA that spawns a Koadic Implant

Operation Cobalt Kitty

Lab 3: Excel Macro 4.0 that executes shellcode

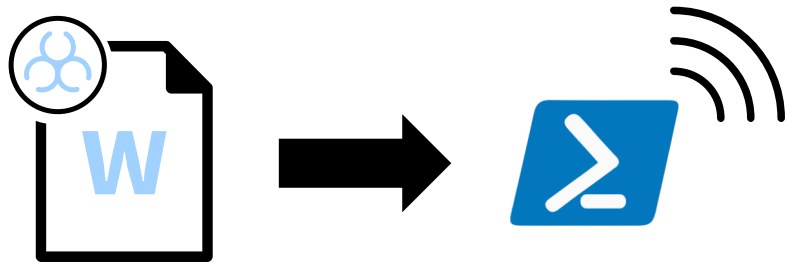
F-Secure | LABS

LAB 1:
POWERSHELL MACRO

F-Secure | LABS

Lab 1: Maldoc that uses PowerShell to establish a C2

We're going to craft a very basic VBA payload that uses PowerShell to retrieve a stager from a remote server. Real world adversaries abused this in campaigns that delivered Emotet. Open source tools like Empire employ similar strategies to retrieve stagers.



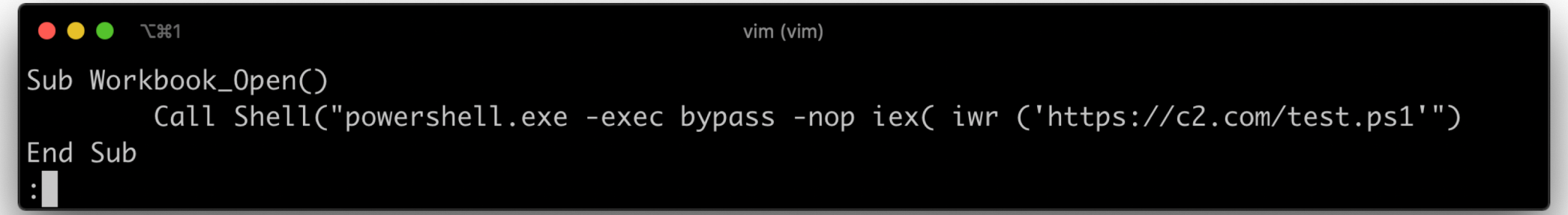
https://github.com/EmpireProject/Empire

https://blog.f-secure.com/hunting-for-emotet/

T1193 – Spearphishing Attachment
T1204 – User Execution
T1086 – PowerShell
T1071 – Standard App Layer Protocol
T1043 – Commonly Used Port

F-Secure LABS

The code will be something similar to this:



What this code does is:

- Spawns PowerShell.exe
- Retrieves the content of the file test.ps1 hosed at c2.com using Invoke-WebRequest (iwr)
- Passes the content of the ps1 file to the Invoke-Expression cmdlet (iex) that interprets the content as PowerShell code.

F-Secure | LABS

DEMO TIME!

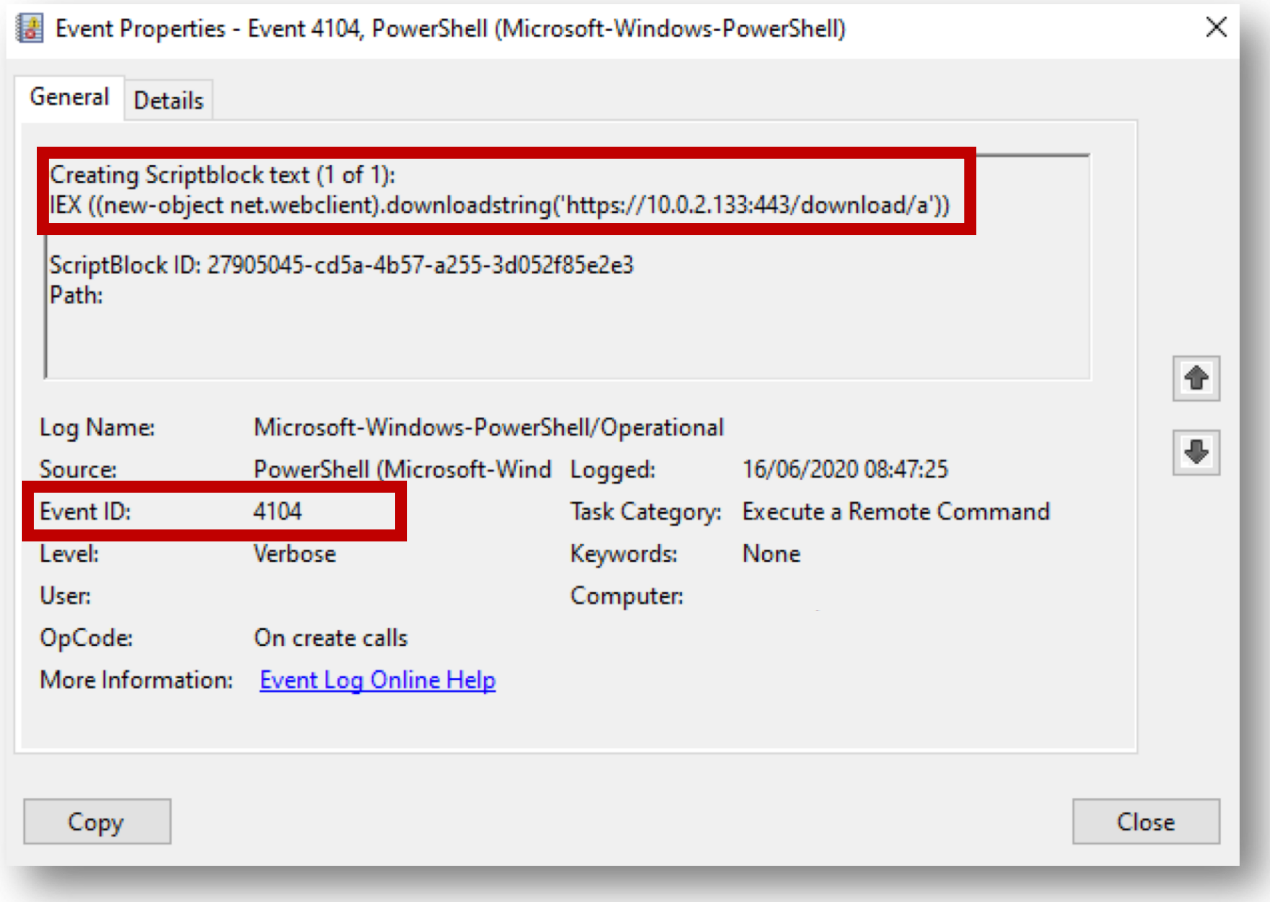
F-Secure | LABS

What did we observe?

- The macro spawned PowerShell → Sysmon Event ID 1 (Process Create)
- The PowerShell code retrieved remote content from an external source → Sysmon Event ID 3 (Network Connection)
- Additional PowerShell code was executed in memory → PowerShell Logs

The most anomalous event here was the process **Excel spawning PowerShell.exe**
The **parent-child process analysis** can be used as a quick win against less sophisticated actors.

F-Secure | LABS



https://devblogs.microsoft.com/powershell/powershell-the-blue-team/

F-Secure | LABS

What did we observe?

- The macro spawned PowerShell → Sysmon Event ID 1 (Process Create)
- The PowerShell code retrieved remote content from an external source → Sysmon Event ID 3 (Network Connection)
- Additional PowerShell code was executed in memory → PowerShell Logs

The most anomalous event here was the process **Excel spawning PowerShell.exe**
The **parent-child process analysis** can be used as a quick win against less sophisticated actors.

F-Secure | LABS

Problem solved? Not really…

Actors have a number of ways for decoupling the execution of a program and circumvent the parent-child analysis. Examples include:

- Execution via WMI
- Execution via COM Objects
- Parent PID Spoofing

https://blog.f-secure.com/dechaining-macros-and-evading-edr/

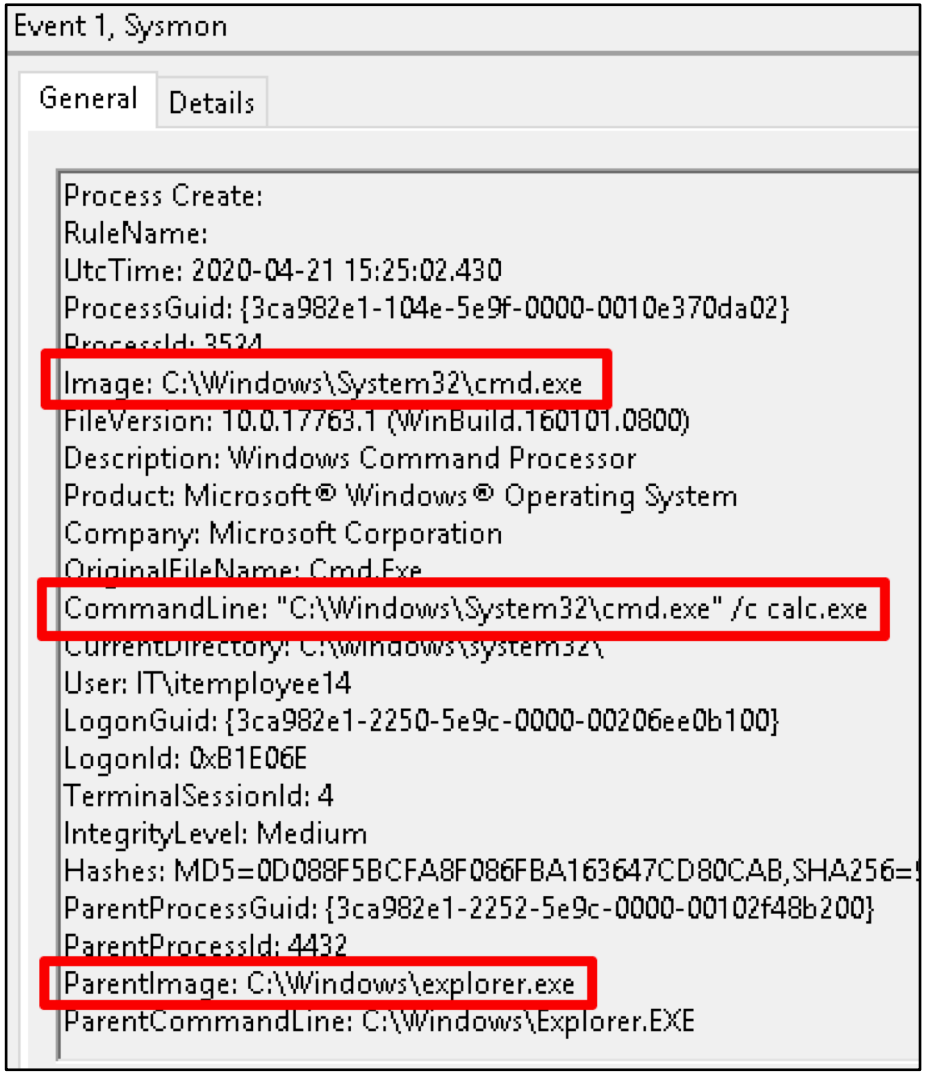
https://blog.christophetd.fr/building-an-office-macro-to-spoof-process-parent-and-command-line/

F-Secure | LABS

Execution via COM Objects:

Component Object Model (COM) is an inter-process communication mechanism that allows a software to expose methods that can be invoked by external applications. *W. Shakespeare*

```
Set obj = GetObject("new:C08AFD90-F2A1-11D1-
8455-00A0C91F3880")
obj.Document.Application.ShellExecute
"calc",Null,"C:\\Windows\\System32",Null,0
```



F-Secure | LABS

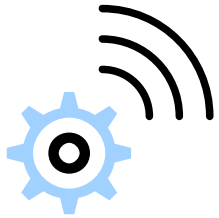
LAB 2:
KOADIC HTA

F-Secure | LABS

Lab 2: Malicious HTA that spawns a Koadic implant

Microsoft HTML Application (HTA) files were quite popular amongst threat actors.
HTA is an extension for HTML compiled applications that allows execution of JavaScript and VBScript without the normal sandboxing features offered by browsers.

When double-clicked, the behavior is similar to the execution of a normal EXE.

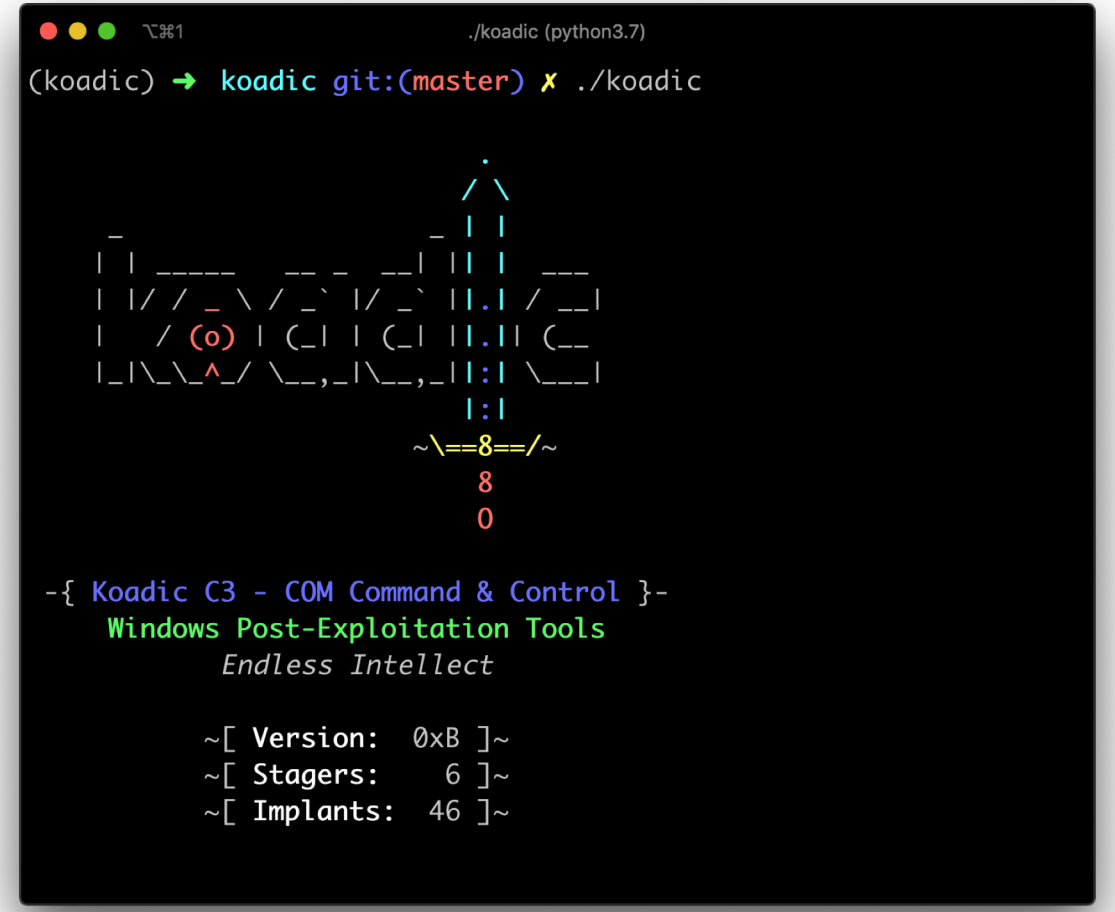


T1193 – Spearphishing Attachment
T1204 – User Execution
T1170 – Mshta
T1122 – COM Hijacking
T1071 – Standard App Layer Protocol
T1065 – Uncommonly Used Port

https://blog.malwarebytes.com/cybercrime/2016/09/surfacing-hta-infections/

F-Secure | LABS

We are going to use ***Koadic***, a Command and Control (C2) framework that relies on COM object to achieve code execution, lateral movement, privilege escalation and persistence.



https://github.com/zerosum0x0/koadic

F-Secure | LABS

DEMO TIME!

F-Secure | LABS

Taking things further…

We are going to cover one of the initial access techniques used by **Operation Cobalt Kitty**.

The main takeaway being that we can **detect part of the attack using the same type detections** that we already showed.

We will be able to combine the first two cases to detect a completely fileless attack campaign delivered by a real APT.

https://cdn2.hubspot.net/hubfs/3354902/Cybereason Labs Analysis Operation Cobalt Kitty.pdf

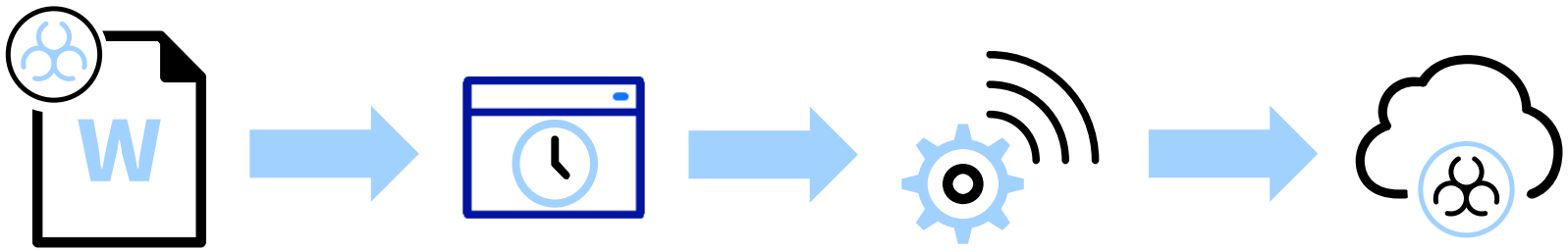
https://www.cybereason.com/blog/operation-cobalt-kitty-apt

T1193 – Spearphishing Attachment
T1204 – User Execution
T1001 – Data Obfuscation
T1053 – Scheduled Task
T1170 – Mshta
T1086 – PowerShell
T1071 – Standard App Layer Protocol
T1043 – Commonly Used Port

F-Secure | LABS

One of the initial access strategies adopted by Cobalt Kitty was the following:

- Word document with a malicious macro
- The macro **created a scheduled task** using the schtasks binary
- The scheduled task executed **mshta.exe that downloaded a stager** from a remote server
- The stager **spawned PowerShell.exe** that downloaded and executed an obfuscated version of a Cobalt Strike PowerShell launcher



F-Secure | LABS

DEMO TIME!

F-Secure | LABS

Using the concepts that we previously introduced, we have at least three detection opportunities here:

- Winword that spawns a known LOLBin (schtask) → Sysmon Event ID 1
- New scheduled task created → EID 4698
- Mshta that downloads content from internet → Sysmon Event ID 3
- A LOLBin (mshta) that spawns PowerShell → Sysmon Event ID 1

https://lolbas-project.github.io/

F-Secure | LABS

```
title: Rare Schtasks Creations
id: b0d77106-7bb0-41fe-bd94-d1752164d066
description: Detects rare scheduled tasks creations that only appear a few times per time frame and could reveal password dumpers, backdoor installs or other types of malicious code
status: experimental
author: Florian Roth
date: 2017/03/23
tags:
    - attack.execution
    - attack.privilege_escalation
    - attack.persistence
    - attack.t1053
    - car.2013-08-001
logsource:
    product: windows
    service: security
    definition: 'The Advanced Audit Policy setting Object Access > Audit Other Object Access Events has to be configured to allow this detection (not in the baseline recommendations by Microsoft). We also recommend extracting the Command field from the embedded XML in the event data.'
detection:
    selection:
        EventID: 4698
    timeframe: 7d
    condition: selection | count() by TaskName < 5
falsepositives:
    - Software installation
    - Software updates
level: low
```

https://raw.githubusercontent.com/Neo23x0/sigma/master/rules/windows/builtin/win_rare_schtasks_creations.yml

F-Secure | LABS

Using the concepts that we previously introduced, we have at least three detection opportunities here:

- Winword that spawns a known LOLBin (schtask) → Sysmon Event ID 1
- New scheduled task created → EID 4698
- Mshta that downloads content from internet → Sysmon Event ID 3
- A LOLBin (mshta) that spawns PowerShell → Sysmon Event ID 1

https://lolbas-project.github.io/

F-Secure | LABS

LAB 3:
EXCEL 4.0 + SHELLCODE

F-Secure | LABS

Lab 3: Excel 4.0 Macro that executes a shellcode

We're going to craft an Excel macro that uses Win32 APIs to execute ***shellcode***.
Shellcode can be defined as:

> *"A small and self-contained piece of software often used to download and execute additional malware code or to deploy an implant."*

Most of the time, shellcode is written in Assembly, but we will use a few frameworks to help us speed up the process.

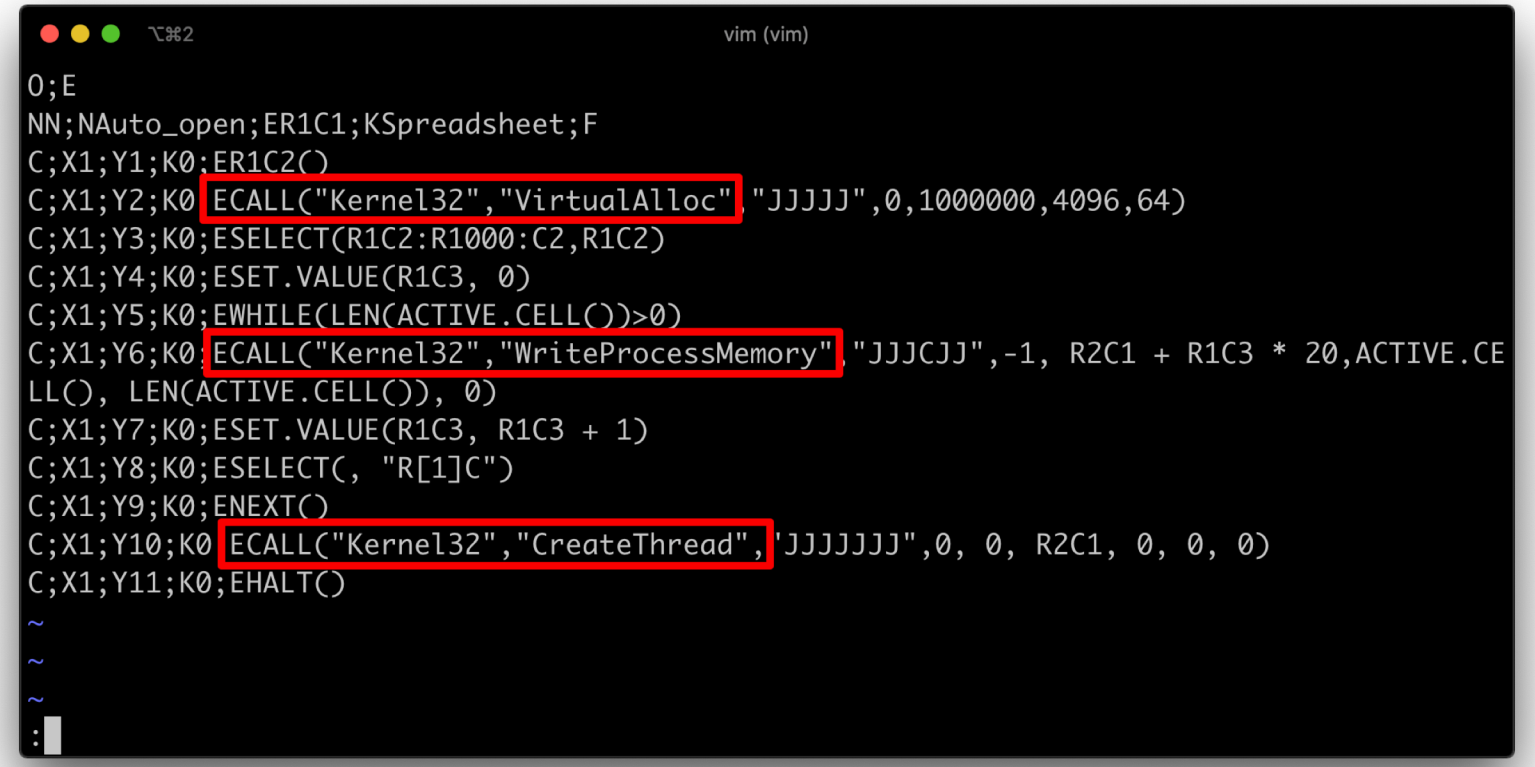
https://outflank.nl/blog/2018/10/06/old-school-evil-excel-4-0-macros-xlm/
https://www.mdsec.co.uk/2019/02/macros-and-more-with-sharpshooter-v2-0/

T1193 – Spearphishing Attachment
T1204 – User Execution
T1055 – Process Injection
T1071 – Standard App Layer Protocol
T1043 – Commonly Used Port

F-Secure | LABS

The following code is taken from one of the templates available within the ***SharpShooter*** payload generation framework.

Understanding every single bit is not necessary, but take note of the highlighted function calls as they are commonly used to execute shellcode.



https://github.com/mdsecactivebreach/SharpShooter/blob/master/modules/excel4.py

F-Secure | LABS

DEMO TIME!

F-Secure | LABS

We can’t use the previously described detection techniques because:

- **No process is created -** Even if the malware author decides to spawn a child process to perform the code injection, the process can be another Excel.exe which would appear legitimate;

- **No suspicious network-process anomalies -** As Excel communicates frequently with external addresses.

https://github.com/FSecureLABS/C3

F-Secure | LABS

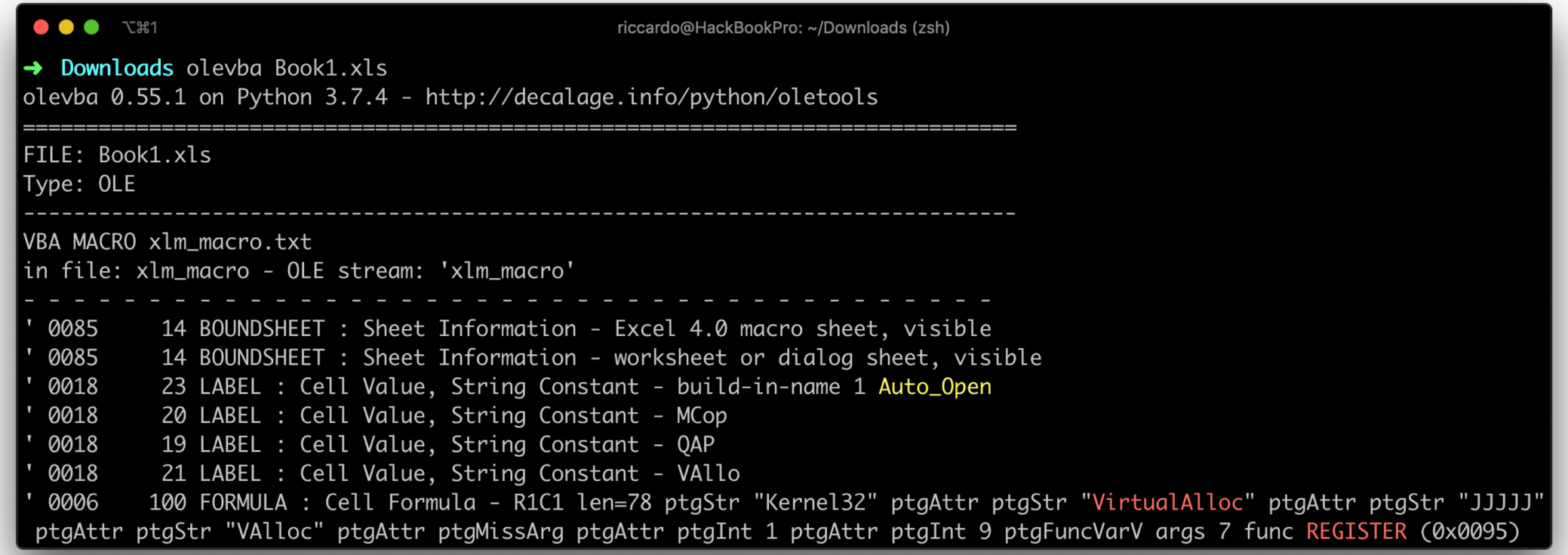
Detection-wise, we can approach this problem from three different angles, each one of them with its own strengths and weaknesses:

- Build a signature for the macro code ⟶ Probably done by your mail gateway
- API Monitoring ⟶ Probably done by your EDR
- Memory Analysis ⟶ Probably done by your EDR

F-Secure | LABS

Signature-based approach

It is possible to build signatures based on the macro content, but actors can obfuscate them. That being said, the amount of false positives is typically low.



F-Secure | LABS

API Monitoring approach

As we already discussed, the execution of the shellcode uses specific API calls that can be monitored. Usually, EDR product have these type of features.

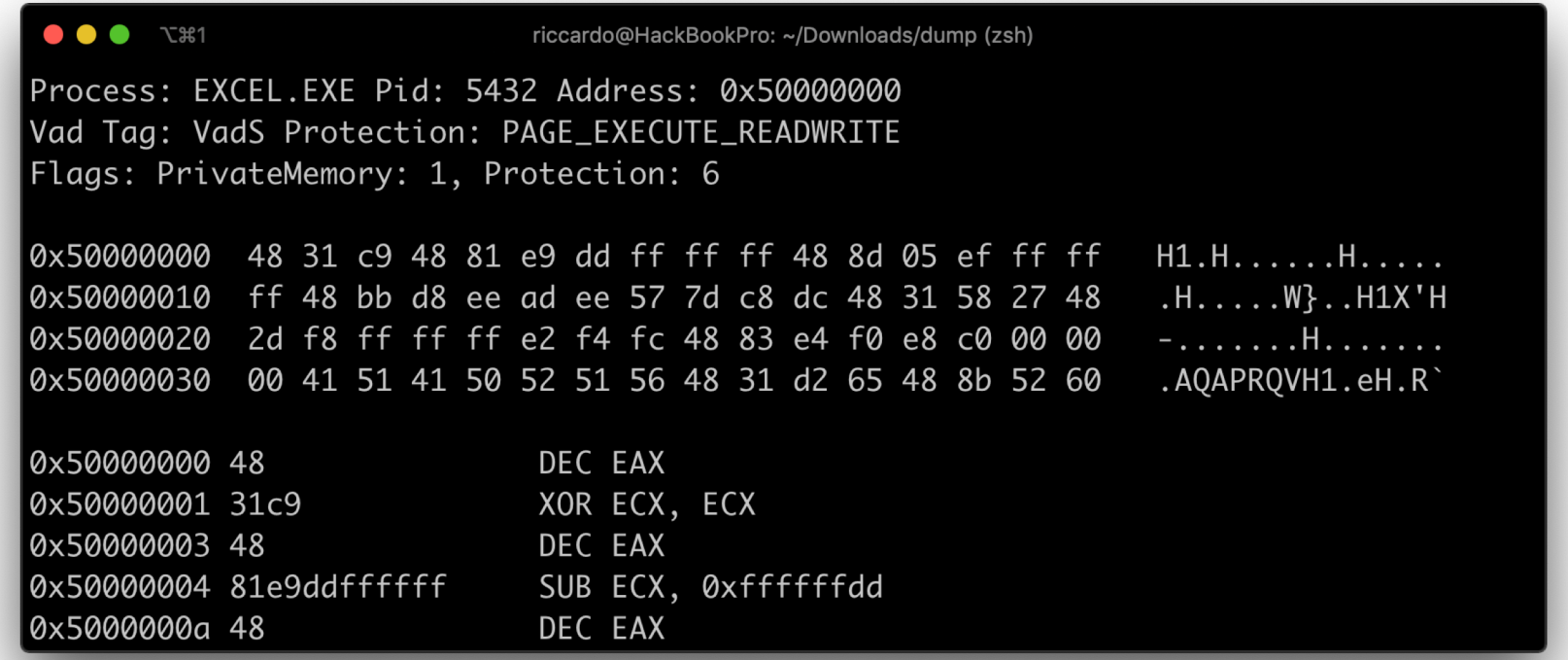
#	Time of Day	Thread	Module	API	Return Value	Error	Dura...
1	3:46:16.347 ...	2	clr.dll	VirtualAlloc (NULL, 65536, MEM_COMMIT, PAGE_READWRITE)	0x051c0000		0.00...
2	3:46:16.347 ...	2	clr.dll	VirtualFree (0x051c0000, 0, MEM_RELEASE)	TRUE		0.00...
3	3:46:16.347 ...	2	clr.dll	VirtualAlloc (0x009b5000, 4096, MEM_COMMIT, PAGE_READWRITE)	0x009b5000		0.00...
4	3:46:16.347 ...	2	clr.dll	VirtualAlloc (NULL, 106, MEM_COMMIT, PAGE_EXECUTE_READWRITE)	0x051c0000		0.00...
5	3:46:16.347 ...	2	clr.dll	CreateThread (NULL, 0, 0x051c0000, NULL, 0, 0x0057ef30)	0x000002d4		0.00...
6	3:46:16.347 ...	2	KERNELBASE...	NtCreateThreadEx (0x0057ebf8, THREAD_ALL_ACCESS, NULL, GetCurrentProcess(), 0x051...			0.00...

However, a huge amount of **legitimate applications** use the same APIs.
Dealing with false positives is one of the main challenges that AV/EDR vendors struggle with.

F-Secure | LABS

Memory analysis approach

Anomalies within the host memory can be identified using tools, such as Volatility. The example shows the detection of a suspicious memory section within the Excel process after we executed our malicious code:



F-Secure | LABS

SUMMARY

F-Secure | LABS

- We explored techniques commonly used for creating Maldocs to obtain *Initial Access*.

- Identified multiple telemetry sources and detection opportunities:
  - Parent-child process relationships.
  - Processes establishing network connections.
  - Suspicious use of Win32 APIs.
  - Use of encoded PowerShell.
  - Memory artifacts.

- We utilised Covenant, Koadic, MetaSploit and Cobalt Strike frameworks.

F-Secure | LABS

NEXT TIME...

F-Secure | LABS

Code execution and Persistence – July 1st 2020 - Anartz Martin

Get to know Code Execution and Persistence tactics. Follow along with demos that illustrate the techniques used by attackers in the wild to:

- Run malicious code to gain foothold on a target's system (Code Execution)
- Maintain this system access consistently through reboots, credential changes, and other operational interruptions (Persistence)
- Consultants will also cover how code execution and persistence can be detected before attackers advance further down the kill chain.

https://www.f-secure.com/en/consulting/events/attack-detection-fundamentals-workshops

F-Secure | LABS