# 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](https://attack.mitre.org/techniques/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.

|  |
| --- |
|  |

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

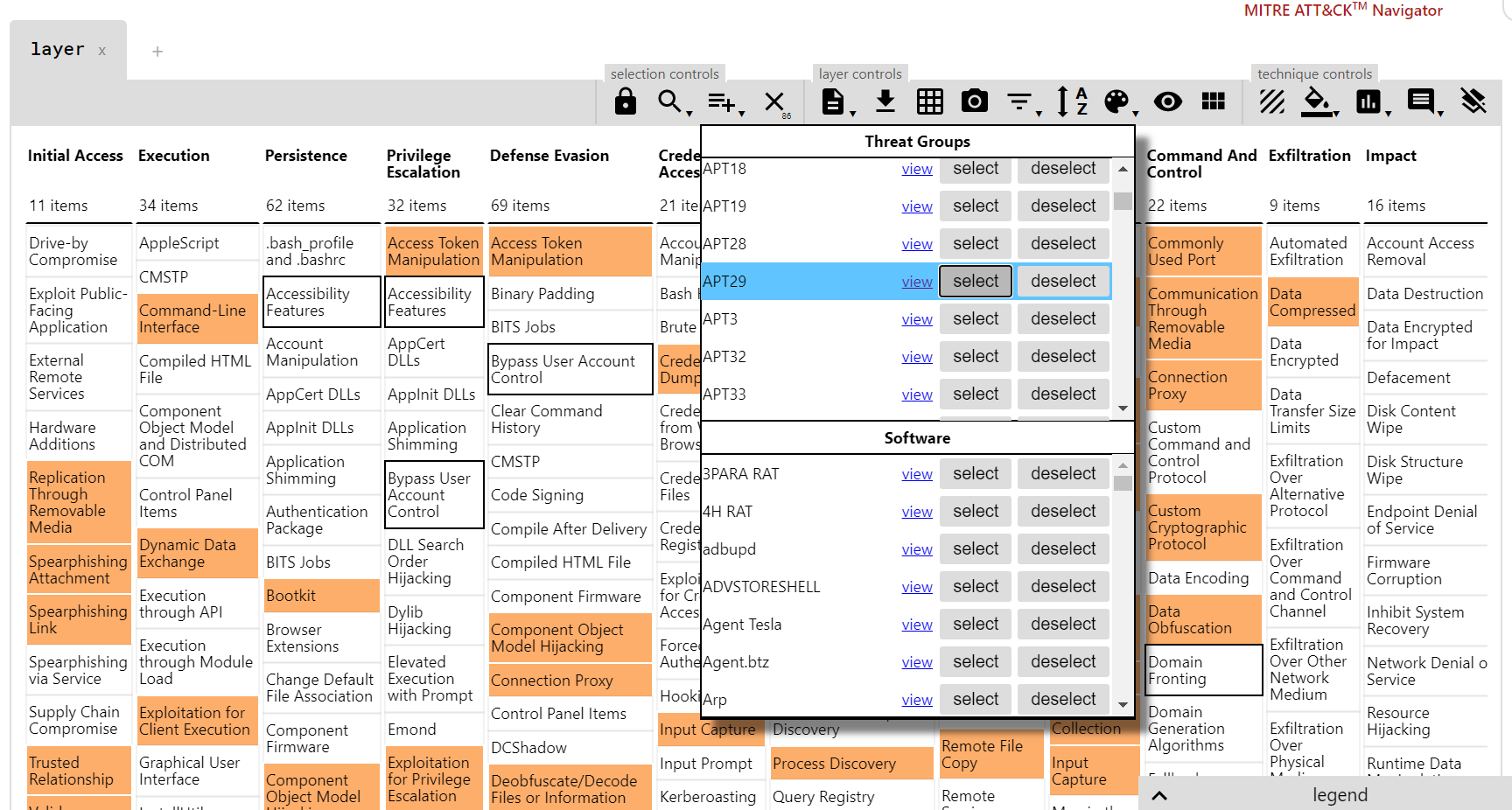
* It is curated. ATT&CK manages complexity by organizing techniques based on attacker objectives, grouping similar techniques together, and relating them to affected platforms.
* It is contributor friendly. In a recent release, most of the new techniques were contributed by researchers outside of MITRE. 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:

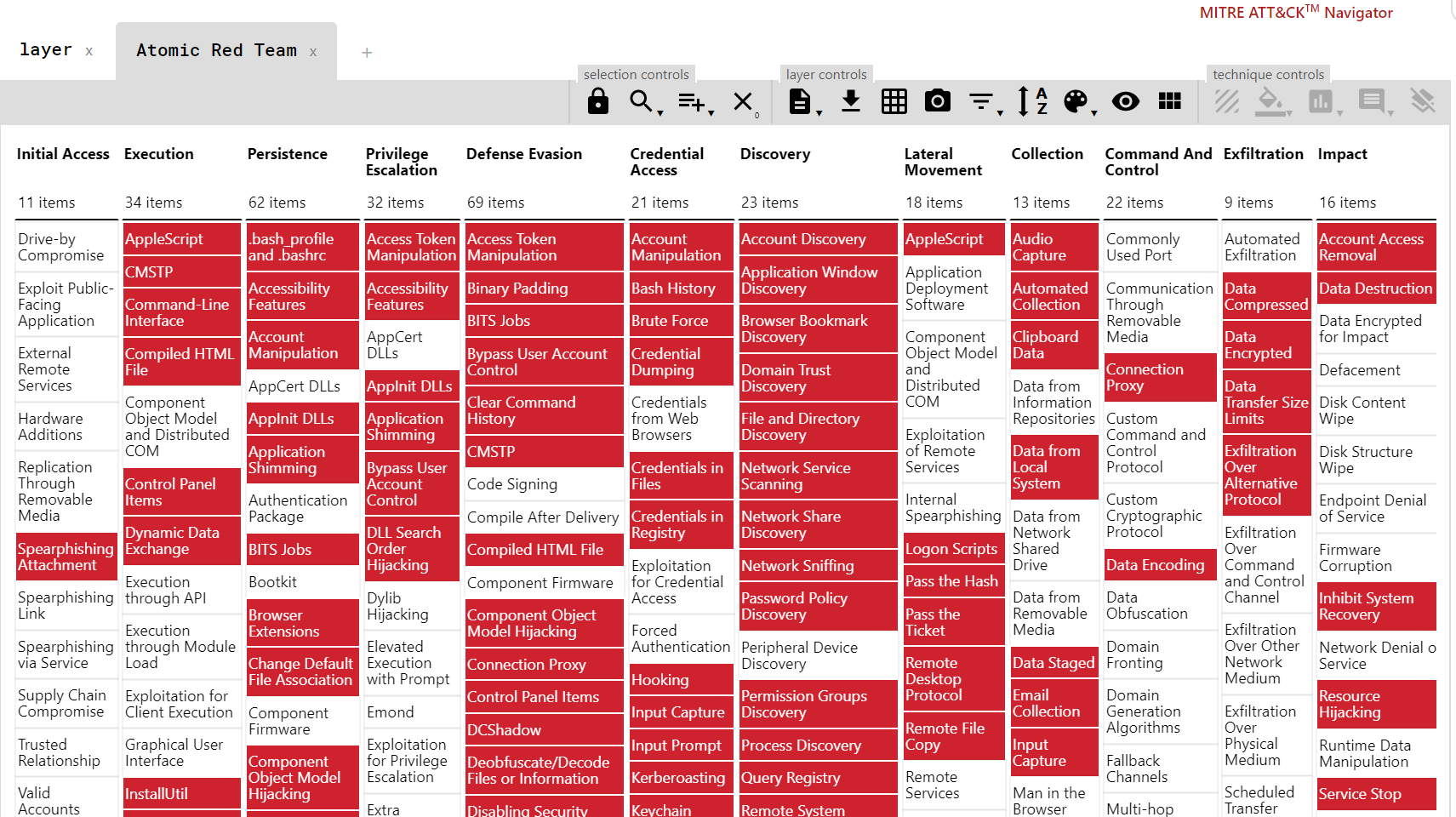
* Threat actors are described by the ATT&CK techniques they use. Defenders can then evaluate their defensive controls against the subset of techniques used by the specific threat actors they face. Here is an example of Palo Alto describing the ATT&CK techniques used by the [Sofacy](https://pan-unit42.github.io/playbook_viewer/?pb=sofacy) threat actor:



* The [ATT&CK navigator tool](https://mitre-attack.github.io/attack-navigator/enterprise/) 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](https://redcanary.com/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):



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

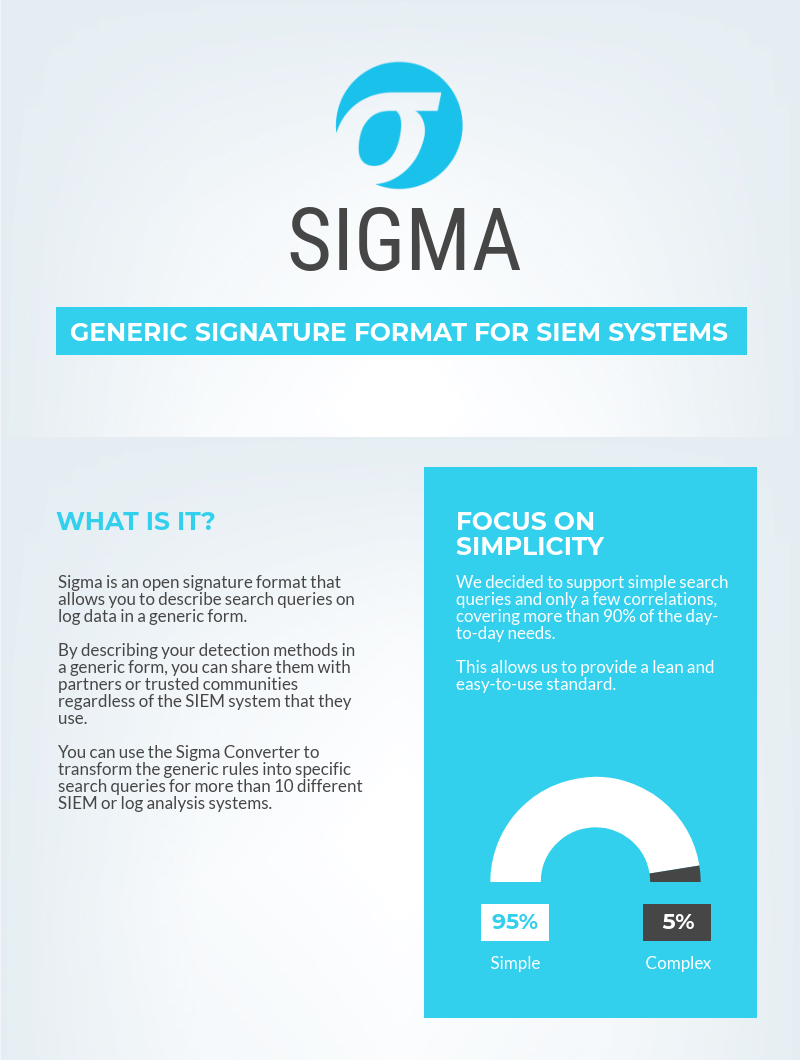
"Every contact leaves a trace” -- Locard's exchange principle

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 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 their 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](https://github.com/Neo23x0/sigma). 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](https://socprime.com/) 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.



With T1015, how can one use Sigma to write detection in a generic way? [This Sigma rule](https://github.com/Neo23x0/sigma/blob/master/rules/windows/sysmon/sysmon_stickykey_like_backdoor.yml) shows how to write a generic detection for both the setting of the registry keys and the invocation of the attack.

|  |
| --- |
| action: global  title: Sticky Key Like Backdoor Usage  id: baca5663-583c-45f9-b5dc-ea96a22ce542  description: Detects the usage and installation of a backdoor that uses an option to register a malicious debugger for built-in tools that are accessible in the login      screen  references:      - https://blogs.technet.microsoft.com/jonathantrull/2016/10/03/detecting-sticky-key-backdoors/  tags:      - attack.privilege\_escalation      - attack.persistence      - attack.t1015      - car.2014-11-003      - car.2014-11-008  author: Florian Roth, @twjackomo  date: 2018/03/15  detection:      condition: 1 of them  falsepositives:      - Unlikely  level: critical  ---  logsource:      product: windows      service: sysmon  detection:      selection\_registry:          EventID: 13          TargetObject:              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe\Debugger'              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe\Debugger'              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\osk.exe\Debugger'              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Magnify.exe\Debugger'              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Narrator.exe\Debugger'              - '\*\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DisplaySwitch.exe\Debugger'          EventType: 'SetValue'  ---  logsource:      category: process\_creation      product: windows  detection:      selection\_process:          ParentImage:              - '\*\winlogon.exe'          CommandLine:              - '\*cmd.exe sethc.exe \*'              - '\*cmd.exe utilman.exe \*'              - '\*cmd.exe osk.exe \*'              - '\*cmd.exe Magnify.exe \*'              - '\*cmd.exe Narrator.exe \*'              - '\*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](https://medium.com/@cyb3rops/an-overlooked-but-intriguing-sigma-use-case-221987f7b588) 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](https://github.com/trustedsec/unicorn) to disguise their intent and evade detective controls. [This notebook](https://mybinder.org/v2/gh/JohnLaTwC/Shared/master?filepath=notebooks%2FPowershell%20Shellcode%20Analysis%20with%20CyberChef.ipynb) 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](https://gchq.github.io/CyberChef/)’s disassembled output to annotate the functionality in the shellcode.

|  |
| --- |
| [Obfuscated PowerShell command](https://www.virustotal.com/gui/file/0c30d700b131246e302ff3da1c4180d21f4650db072e287d1b9d477fe88d312f/community) flagged by a rule: |
| After decoding the Base64 command, the following shellcode is found: |
| Searching for strings to find the callback domain: |
| Disassemble the shellcode and annotate the APIs to discover its functionality: |
| 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](https://github.com/trustedsec/unicorn): “Magic Unicorn is a simple tool for using a PowerShell downgrade attack and inject shellcode straight into memory.” -- Dave Kennedy (@HackingDave) |

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](https://posts.specterops.io/threat-hunting-with-jupyter-notebooks-part-1-your-first-notebook-9a99a781fde7) on how to use them for that. Netscylla also has [a blog](https://www.netscylla.com/blog/2019/10/28/Jupyter-Notebooks-for-Incident-Response.html) 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](https://github.com/JohnLaTwC/Shared/tree/master/notebooks) indicated by the launch binder icon:

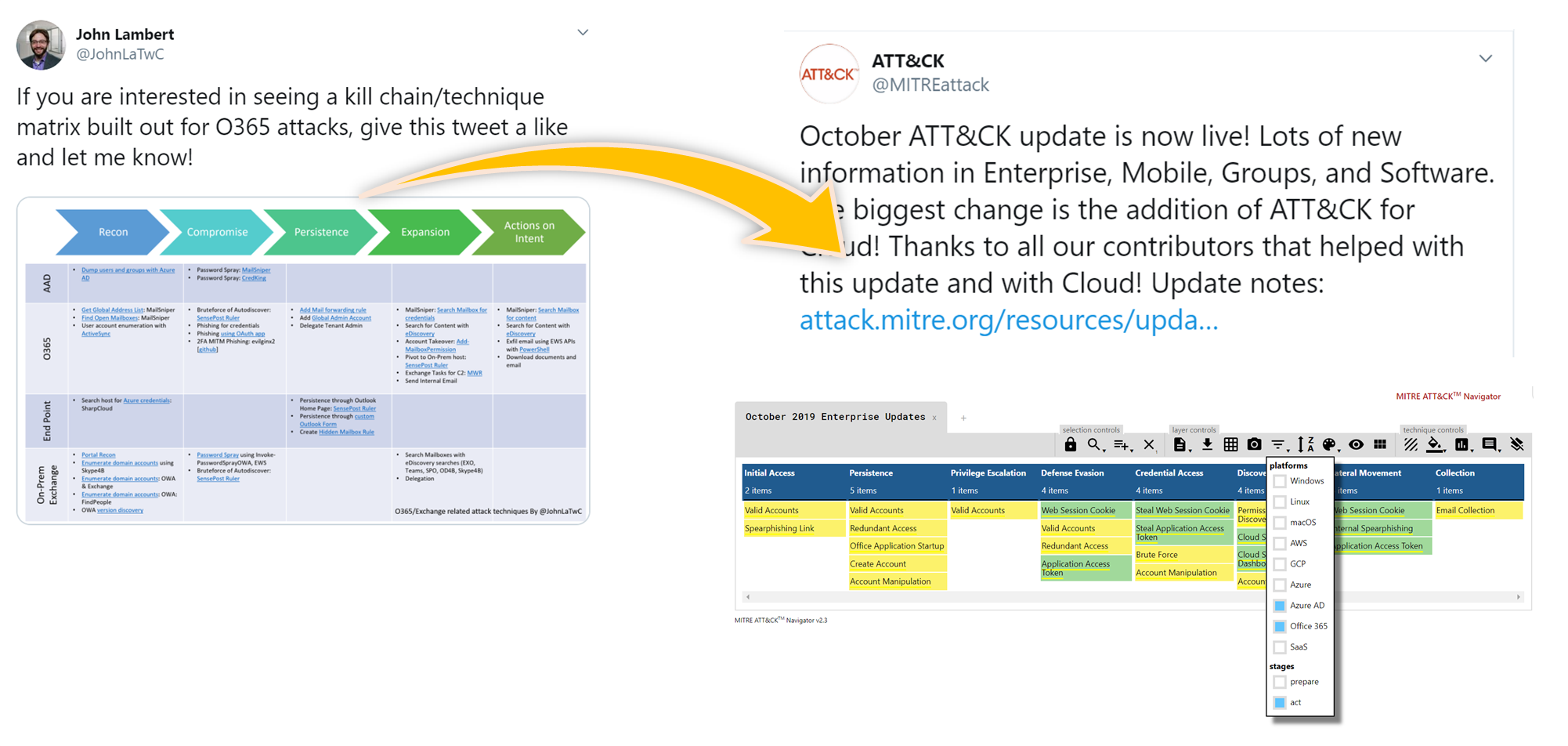


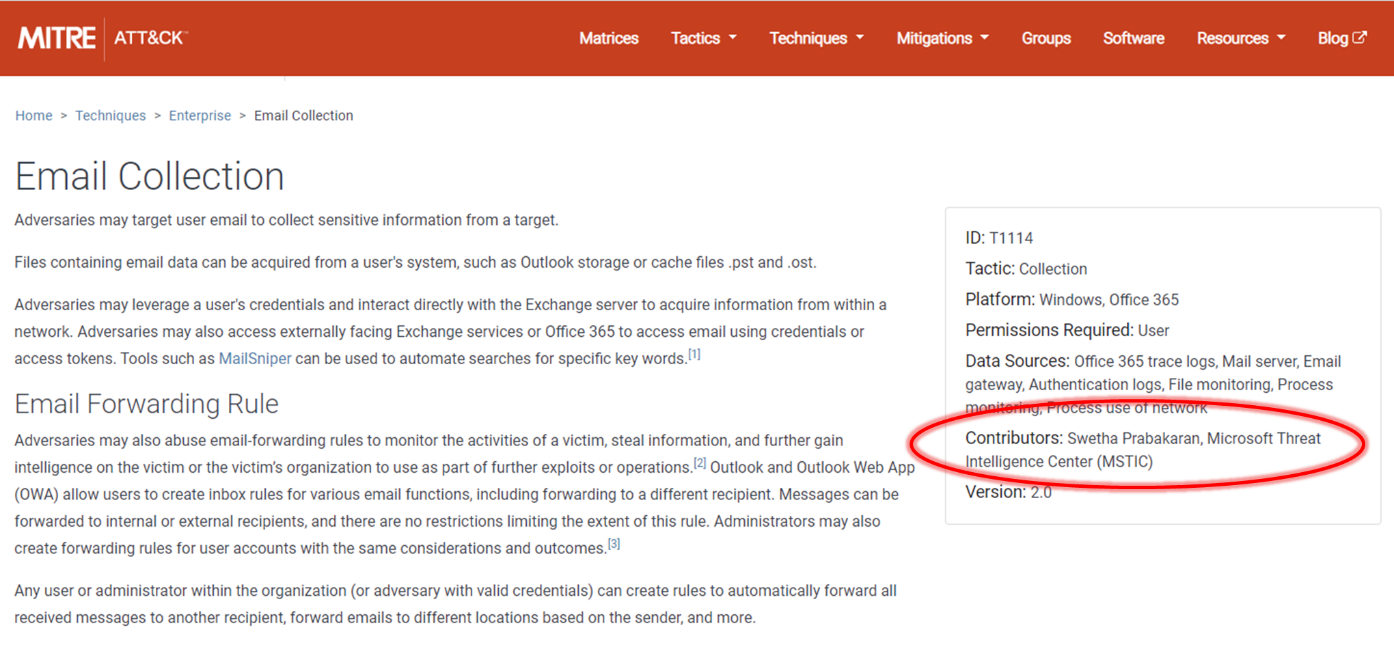
Building on the Work of Others

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.

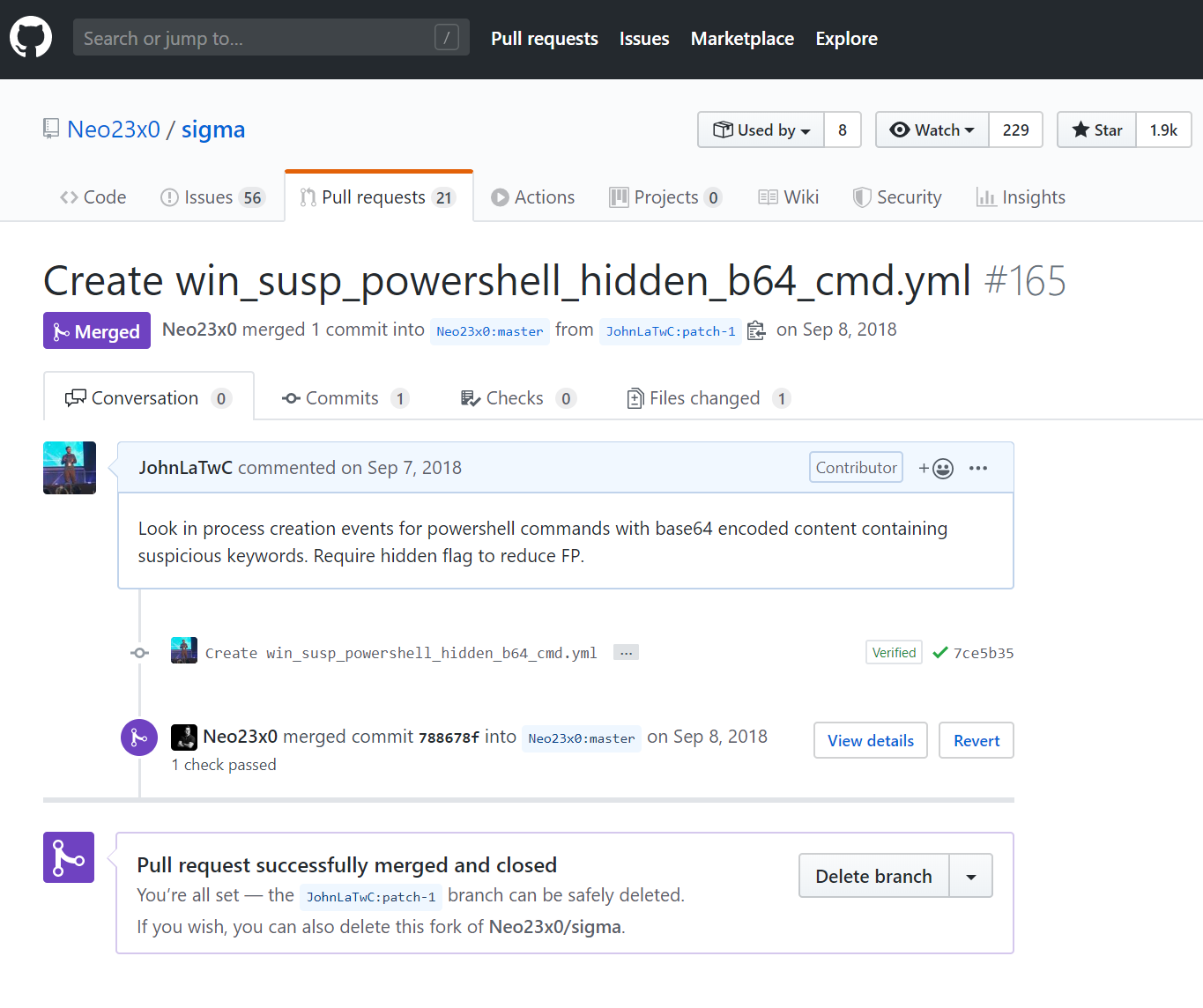
# Promoting Community

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](https://twitter.com/JohnLaTwC/status/1187604286064209921)) were almost entirely sourced from the community.

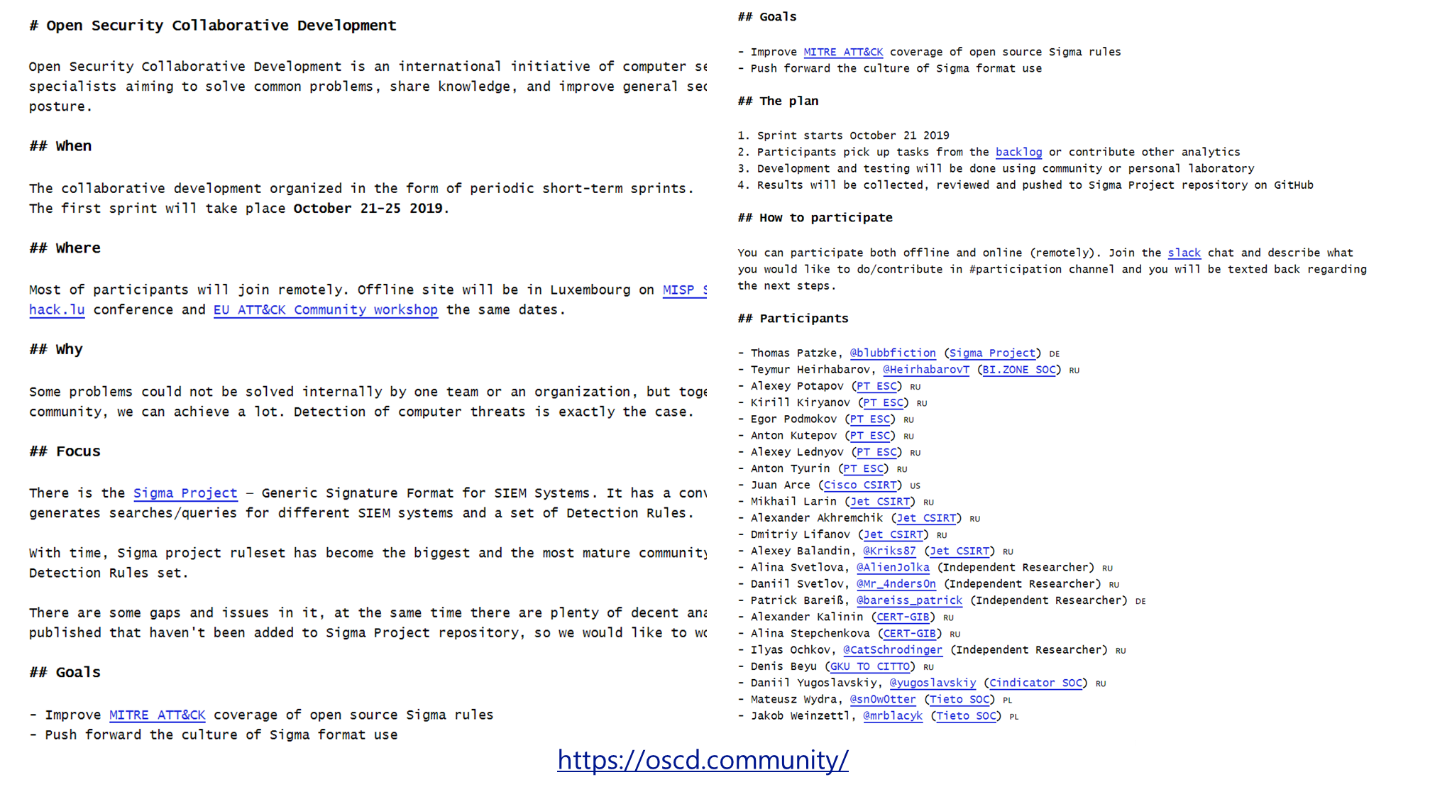




Florian Roth (@cyb3rops) has an [open source repository](https://github.com/Neo23x0/signature-base/tree/master/yara) of Sigma rules on GitHub. Contributing to them is as simple as creating a “pull request,” a request to incorporate a submission. In this example of a [pull request](https://github.com/Neo23x0/sigma/pull/165) to add a new Sigma rule.



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.



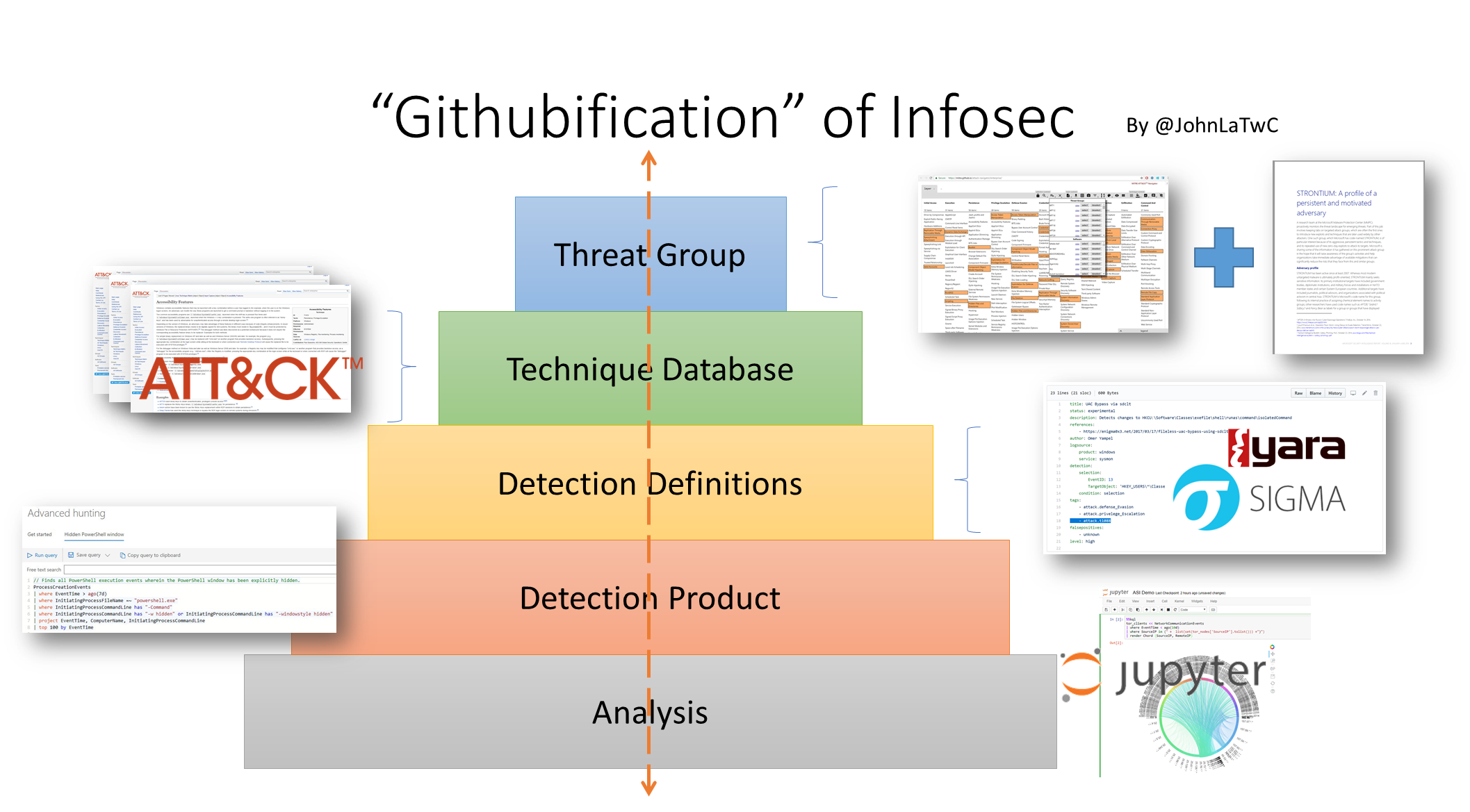
# The Githubification of Infosec

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 | * Write and apply a [Sigma](https://github.com/Neo23x0/sigma) rule * Contribute a rule back to an [open source repository](https://github.com/Neo23x0/sigma/tree/master/rules/windows/sysmon) * Try out a [Jupyter notebook](https://mybinder.org/v2/gh/JohnLaTwC/Shared/master?filepath=notebooks%2FEnvironmental%20Key%20Login.ipynb) on mybinder * Take an [online](https://attack.mitre.org/resources/contribute/) [course](https://www.pluralsight.com/browse/software-development/python) on learning Python |
| Security Product Engineer | * Support Sigma rules in your product such as [JoeSecurity has done](https://www.joesecurity.org/blog/8225577975210857708) * Publish a notebook that uses data from your product * Support Python interfaces to your data |
| Security Researcher | * Publish a notebook demonstrating a technique * Contribute Sigma rules to a repository * Add new attack techniques or examples to [MITRE ATT&CK](https://attack.mitre.org/resources/contribute/) * Publish data-sets useful for testing Sigma rules such as the [MORDOR](https://github.com/hunters-forge/mordor/) project. |
| Infosec Manager | * 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](http://blog.joesecurity.org/2019/10/joe-sandbox-sigma.html), and Jupyter |
| Cyber security organization or CERT | * 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)

# 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/nteract/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>

## 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.

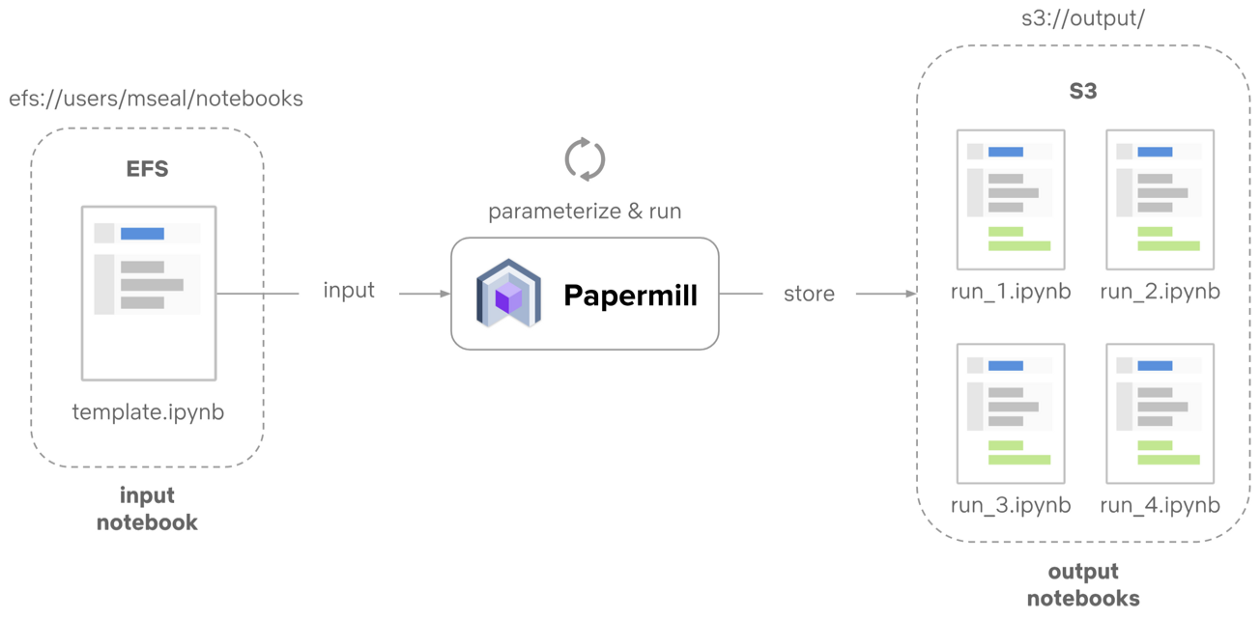
|  |  |
| --- | --- |
| Area | Improvement |
| ATT&CK | * Link to Sigma and Yara rules * Provide logs where the TTP was demonstrated such as done with the [MORDOR project](https://github.com/hunters-forge/mordor/). * Document attack examples for techniques that are lacking public information. * Increase coverage of network-based visibility of techniques * Improve the mitigation resources in the ATT&CK repository |
| SIGMA | * 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 | * 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](https://github.com/nteract/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](https://mybinder.org/) 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:

