



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Network Monitoring and Threat Detection In-Depth (Security 503)"  
at <http://www.giac.org/registration/gcia>

# Methods to Employ Zeek in Detecting MITRE ATT&CK Techniques

*GCIA Gold Certification*

Author: Mike McPhee, [mjmcphree@gmail.com](mailto:mjmcphree@gmail.com)

Advisor: *Jonathan Risto*

Accepted: *June 19, 2020*

## Abstract

MITRE ATT&CK techniques and their respective detections, while a significant step forward in democratizing threat intelligence, are predominantly focused on endpoint visibility through direct management or via agents. Some detection approaches leverage network sensors (e.g., Zeek) like BZAR (Fernandez, Wunder, Azoff, & Tylabs) in network-based detection of ATT&CK techniques. However, many of these earlier solutions focus on Microsoft Windows-specific protocols. They do not provide broad coverage of less-sophisticated endpoints, industrial systems, or infrastructure devices themselves (such as routers, switches, wireless devices). This paper will explore the feasibility of network-based detections using combinations of CLI utilities and Zeek IDS to augment or replace endpoint-focused detections and extend ATT&CK's utility to the rest of the network.

## 1. Introduction

MITRE ATT&CK's extensive database (<https://attack.mitre.org>) of Tactics, Techniques, and Procedures (TTPs) provides an open-source repository of adversarial behaviors that apply to the information security disciplines of Threat Intelligence, Adversary Emulation, Gap Analysis, & Detection & Analytics (Strom et al., 2018). The data sets used to populate ATT&CK were formerly and still are predominantly endpoint telemetry gained through the logs and alerts that incident responders have submitted from available antivirus, enhanced detection and response, antimalware, VPN, or other clients present on the impacted devices (Strom et al., 2017). While an endpoint presence is ideal, it is not always feasible or possible.

Heavily IoT-based environments (manufacturing, logistics, utilities, healthcare, and hospitality) leverage light-weight network stacks for cost-effectiveness. These devices are, with few exceptions, capable of running the same software suites as the end-user workstations and laptops within the environment. Additionally, guest and mobility networks used in hospitality, retail, or sports and entertainment industries are prohibited or strongly discouraged from gaining a presence on their tenants. Lastly, infrastructure devices, including routers, modems, wired and wireless network access devices, and other elements, offer attackers safe harbor when misconfigured or compromised due to a lack of consistent monitoring and telemetry on their operations. These limitations introduce blind spots that attackers use to disguise their efforts. The Mirai Botnet (Bursztein, 2017), Sandworm's CRASHOVERRIDE (Slowik, 2018), and evolving VPNFilter (Largent, 2018) threat are three of the many attack methods that have taken advantage of these blind spots with devastating consequences.

It is this researcher's opinion that expanding the envelope of ATT&CK can improve the outcomes for these environments that may lack full endpoint visibility or control. The ICS, Mobile, and Enterprise matrices all have tactics that at least a portion of their techniques result in some discernable network footprint. This research will investigate the feasibility of using open-source Network Service Extraction using Zeek IDS to uncover these techniques in-flight. This work will also explore the limitations of this approach and offer recommendations for further research.

## 2. Research Method

A sandbox environment and packet captures of real-world attacks available to the public tested the methods discussed in this research. While a controlled environment is undoubtedly useful, ensuring broad applicability for the scripts and techniques explored here necessitated the inclusion and development of real-world network captures. The public captures offered two distinct advantages: 1) ensured public accessibility and repeatability of the findings, and 2) rapid testing of a much more extensive set of scenarios and use cases than was possible in an isolated sandbox.

In addition to this relatively pure standard, shifting the Zeek sensor to the researcher's live network offered a more complete array of clients and protocols in use. These captures and the resulting Zeek logs provided significant value in the diversity of findings. They were used to help tune the Zeek detections and eliminate false positives throughout the scripting process.

### 2.1. Simulation Environment

The lab supporting this research was built in a VMWare ESXi environment and consisted of a single Ubuntu Linux server running Zeek IDS and using a secondary interface to receive mirrored traffic from a hybrid sandbox environment. This sandbox environment consisted of two sites connected to a simulated internet. The Headquarters Site included a Cisco virtual router and two physical Cisco Catalyst switches, each offering either data center or campus access. In turn, these switches trunked back into the ESXi host to provide access to several enclaves of Linux and Windows VMs. A second CSR provided connectivity for a simulated branch Windows VM. Both CSRs peered over a Sandbox Internet, with limited outside connectivity via a virtual firewall. An attacker VM, running Kali Linux 2020.2 (<https://www.kali.org>), was also resident on the Sandbox Internet to offer command and control (C2) and act as an exfiltration target. This process was instructive in helping to characterize what a healthy, low-client count network might approximate. Figure 1 depicts the sandbox environment:

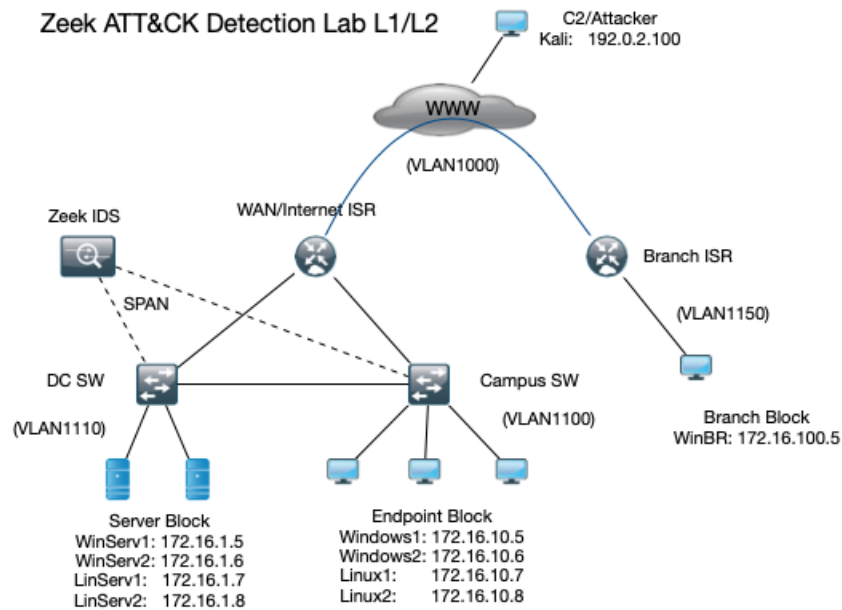


Figure 1: Sandbox Environment

### 2.1.1. Server and Client Capabilities

Windows 10 VMs and Ubuntu 18 hosts acted as clients. Windows 2016 R2 servers provided most of the data center traffic, but a single Ubuntu server running a demo environment of ELK and Samba emulated a mixed OS environment. Simple shell scripts were written to simulate typical network traffic: web browsing and application access by the client VMs, domain controller activities by the Windows Server VMs, and a mix of file transfers and regular management traffic (like GPO push and SMBv2 shares) to establish a baseline of traffic. All essential services provided typical traffic patterns including NTP, DHCP, DNS, simple SMTP-based email, file sharing, and domain services. VNC/RDP access was enabled from a single "employee" client to the servers. "Contractor," "Branch," and "Guest" VMs were each allowed varying levels of access.

### 2.1.2. Network Infrastructure Provisioning

Cisco Configuration Guides and software version 17.1

(<https://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-xe-amsterdam-17-2->

[1/model.html](#)) guided the deployment of network infrastructure devices. The primary sections of the configuration guides used were IP Addressing, IP Routing, Security Configuration, and Basic System Management. The guidance provided in these sections is not as secure as the NSA's recommendations for configuration, but adherence to those standards varies greatly. Manufacturer recommended settings tend to better approximate those encountered in real-world deployments.

Cisco's Switched Port Analyzer (SPAN) feature mirrored traffic with encapsulation replication to ensure the Zeek VM received full streams of characterized traffic. With a baseline established, several scenarios were run to simulate compromise. Also, less-secure features were enabled on network devices to simulate their compromise (e.g., Telnet, HTTP, TFTP, plain-text FTP, and SNMPv1/v2).

## 2.2. Packet Capture Approach

Packet captures sourced from various publicly available archives were used to formulate the detection methods and test their applicability. Played back using `tcpreplay` (<https://tcpreplay.appneta.com>) on the loopback interface, a freshly deployed Zeek sensor with the TLS extension was used to parse logs and provide suitable datasets for the wide variety of techniques considered for this research. The Appendix provides links to the publicly available packet capture libraries.

## 3. Findings and Discussion

### 3.1. 3.1 MITRE ATT&CK

#### 3.1.1. ATT&CK's Relationship Model

It is useful to explain their relationship model to better understand MITRE's approach to characterizing advanced persistent threats (APTs). Adversary groups tend to build habits or preferences (TTPs), which are observable behaviors that arise during operations against their targets. Whether using known software or Techniques to achieve their end-goals (Tactics), these behaviors leave digital artifacts that can be detected. By addressing the problem earlier in the timeline, defenders can indeed mitigate the exposure of these techniques using defensive solutions, features, or detections. Figure 2 illustrates this relationship model (Strom et al., 2018).

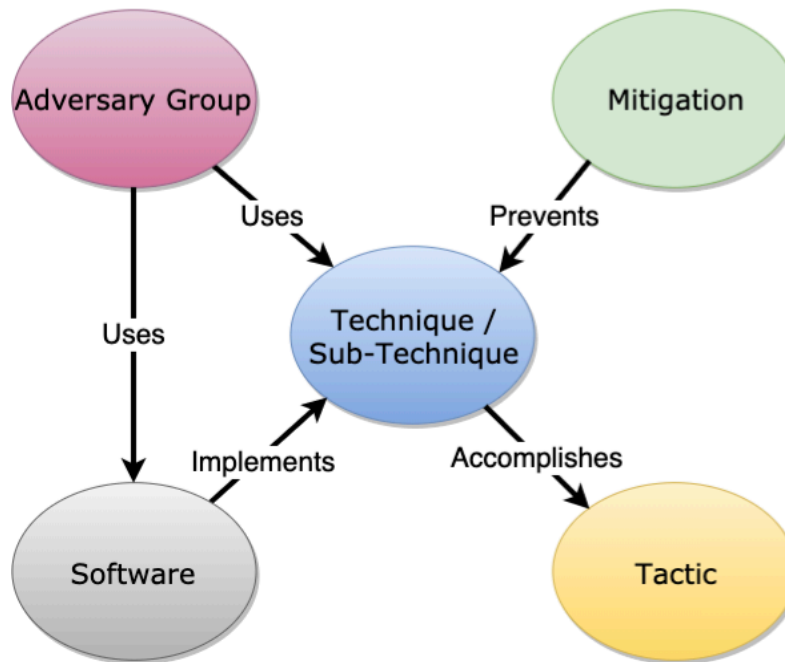


Figure 2: MITRE ATT&CK's Relationship Model (Strom, et al. 2018)

### 3.1.2. Detection Techniques

ATT&CK detection techniques based on endpoint telemetry focus on specific and unique artifacts to identify a particular technique's use or presence on the device. These artifacts or tests include any number of discreet indications but often focus on registry keys, file hashes of particular system files, Powershell, net/netsh command polling results, etc. Tools like Red Canary's Atomic Red Team (<https://atomicredteam.io>) and MITRE's Caldera (<https://github.com/mitre/caldera>) that leverage these detections to both reveal compromise or validate defenses are sufficiently protecting against these techniques. In the network, these specific, atomic indicators are not available. This precludes using network telemetry and service extraction for some techniques (and, in fact, entire tactics) in the matrices.

### 3.1.3. Pros & Cons of an Endpoint Focus for Detection

Insights gained from endpoints are preferred over those elsewhere in the environment for several reasons. Access to the operating system's kernel and other services and daemons offers details that cannot be revealed elsewhere. The widespread adoption of encryption in transport also obscures data exchange and intent, leaving other

detection mechanisms guessing. Most endpoint suites do not require a hardware footprint to support, resulting in a more flexible range of deployment options available to the system administrators. Perhaps most significantly, detection and protection or remediation can best be accomplished together on the endpoint. Defensive solutions like Intrusion Detection/Prevention Systems, Firewalls, and Network Traffic Analytics (NTA) tools can indicate vulnerability or attack and even some blocking action to prevent its spread. However, the root cause will still reside within the targeted systems. Endpoint solutions largely address these gaps.

The quantity and quality of endpoint telemetry and forensics from incident response activities have driven much of the characterization of TTPs within MITRE ATT&CK and for a good reason. There are limits to the industry's focus on the endpoint and tools to both execute and detect these same techniques.

Endpoint-focused approaches to detecting techniques assume the defenders' administrative control, rely on properly configured telemetry, and coordinate some form of intelligence to deliver detection and protection. Many environments, however, include a variety of essential endpoints or network nodes that are unable or inaccessible to endpoint solution sets. Administrative policies are one form of barrier – in guest, BYOD, or multi-tenant networks, it is unlikely that a network system's operators can be granted access to endpoints due to privacy, safety, or liability concerns.

The operating systems themselves – a vital source of any endpoint-focused approach's strength, can prevent their use. Network operating systems like Cisco's IOS, Juniper's JunOS, or others do not support third-party tools from providing similar access and insight. IoT vendors' focus on cost-effectiveness and low power/size/weight dictate bare-minimum network stacks and low compute resources that preclude using any commercial suites.

### **3.2. Network Service Extraction for Detection**

Network-based detection of malicious activity is nothing new – Cliff Stoll's famous The Cuckoo's Egg (Stoll, 1989) documents early attempts at using a hybrid network and operating-system based approach to detecting and prosecuting a threat. Intrusion Detection Systems (IDSs), especially NTA solutions, can offer a wide variety



of endpoint-independent detections, but many do not yet directly map their approaches to the ATT&CK Framework. A more generic approach to seeing ATT&CK's TTPs from the network's perspective would alleviate many of the key limitations of an endpoint-only approach. Monitoring the use of network telemetry allows better coverage of the non-corporate user machines, mobile devices, and IoT equipment on the network. It can also unveil suspect activities implicating infrastructure devices (routers, switches, security appliances) without impacting their configuration.

### **3.2.1. Traditional Applications of NSE**

NTA detections run the gamut between highly tuned, context-dependent devices and drop-in appliances that require little or no tuning or configuration for their environment. The underlying detection mechanisms are usually categorized as signature-based or behavioral. Most behavioral methods also use signatures but take a broader view and leverage context to improve outcomes. Behavioral tools compare behavioral patterns on the network and blend this pattern matching with heuristics or anomaly detection to broaden coverage rather than relying on atomic events or event combinations.

Unlike strictly signature or behavioral-based detection solutions, approaches using Network Service Extraction (NSE) don't merely look for threats. An NSE begins by observing all traffic between every host pair on a network and applies protocol analysis and mapping techniques to characterize everything that happened. In its purest form, NSE-based outputs do not automatically generate alarms. Initially, the operator will interpret results and make decisions. As the operator becomes more aware of the environment, they can begin to tune and configure NSE tools to implement simple scripts and signatures. These are consciously implemented to help automate the analysis for encountered patterns and events. As a result of their context-awareness and all-seeing scope of traffic, NSE-driven approaches can quickly surpass strictly signature-based solutions in efficacy and accuracy.

### **3.2.2. Challenges in NSE**

The lack of turn-key detections and response is a barrier to entry for NSE approaches. NSE is usually overlooked as a suitable monitoring solution in most environments in favor of off-the-shelf products that immediately begin to generate alert

traffic. NSE monitoring approaches can be quickly deployed to catalog and process flow and service information for the segments of interest. However, until an operator or analyst can look over the information, NSE is unable to provide immediate alert and automation value short term.

In that initial phase, human operators can parse the multitude of logs, but only after familiarizing themselves with the baseline can live detection value be gained. Catching attacks as they happen is possible once baselines are established. Scripts or even signatures may be written to mark deviations in that baseline or other patterns of interest as a malicious or alert-worthy activity, but only after the baseline has been captured and parsed.

Solutions rooted in NSE rarely include built-in response capabilities. Where a response is desired, operators often rely on integration to occur inside of a Security Information and Event Manager (SIEM) and potentially inform the automated response via integrations with enforcement tools or a Security Orchestration & Automated Response (SOAR) tool. These integrations themselves do not occur without some customization and tuning. Enterprises without dedicated security analysts trained in NSE are more likely to acquire security products that include detection and protection capabilities together, or quickly offer detections that can be readily consumed by response tools without significant tuning or customization.

Experienced operators of NSE-based tools can accelerate the tuning and deployment, but this expertise is not as widely available as that of next-generation firewall (NGFW) operators, IPS/IDS analysts, etc. Training offerings are also much more widely available for the more main-stream detection toolsets (e.g., Snort & Suricata, as well as commercial IDS/IPS vendors), whereas NSE solution training is much scarcer.

### **3.2.3. Zeek IDS**

Vern Paxson began the development of the leading open-source NSE-based tool, Bro IDS, in 1995. Built as a Linux/BSD-based software suite, Bro was made for scale, customization, and performance. It has continued to evolve and expand its capabilities through an active open-source community and the support of Corelight (a premium offering and support company created by Paxson and other significant contributors) as

well as major customers who leverage its capabilities. Bro was renamed Zeek in 2018. The architecture simply parses every packet mirrored to it from a segment of interest. Zeek then applies multiple parallel Event Engines to generate a large variety of events to mark the presence of every discernable type of transaction. A file transfer, no matter who it is between or for what purpose, will generate an event that passes all relevant data on the transaction. These events provide the context that Zeek then logs across a wide variety of protocols and data set types. These same events are also useful in the Event Scripting tier of the Zeek architecture, in that operators can use these events, and the information they pass, to trigger signatures or handoff information to scripts for correlation, alerting, or external integration. It is in this Event Scripting tier that the operators can imbue Zeek's policy-neutral capabilities with the power to observe and act based on the site's scripted policy.

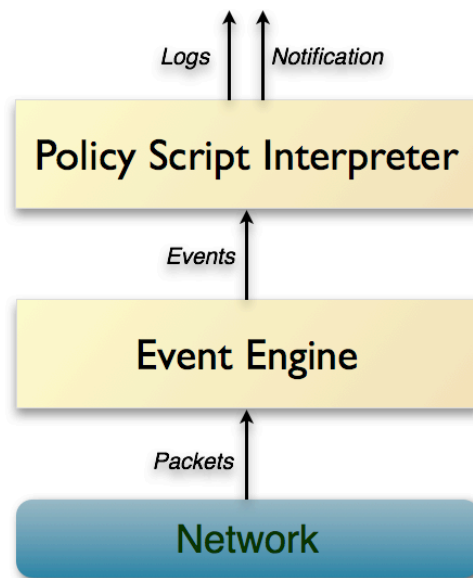


Figure 3: Zeek's Internal Architecture

Zeek's simple architecture also allows it to scale well to massive environments using a management daemon called `zeekctl` which manages multiple "workers." Each worker operates the event engines to process events and parse flows sent to them by a load-balancing front-end solution (e.g., `PF_RING` or `cPacket`). Their logging and generated events can be sent along to a centralized Manager or Logger process for correlation, centralized analytics, and reporting.

### 3.3. Choosing Candidate TTPs

The Bro/Zeek ATT&CK-based Analytics and Reporting (BZAR, <https://github.com/mitre-attack/bzar>) project tackles network-driven detection by combining Zeek's SMB and DCE-RPC protocol analyzers with its ability to perform file extraction. While BZAR has demonstrated network detection of ATT&CK techniques is possible, BZAR's coverage is focused on Windows-based management protocols. Expanding that concept to a broader base of non-Windows endpoints and infrastructure devices requires a more generalized approach.

To identify the most applicable tactics from the ATT&CK Enterprise matrix, a survey of techniques within each led to the logical selection of three tactics as prime candidates for this research. The Exfiltration (TA0010), Command and Control (TA0011), and Lateral Movement (TA0008) tactics offer ample opportunities to detect corresponding activity over the wire. All include a significant number of techniques that implicate the network in their execution. Zeek provides a rich number of event types to work with, and it sees and parses all data. Therefore, it should suit this role well. Figure 3 outlines some of the more "Zeek-ready" techniques for detection.

Lateral Movement (TA0008)	Command And Control (TA0011)	Exfiltration (TA0010)
Exploitation of Remote Services (T1210)	Commonly Used Port (T1043)	Automated Exfiltration (T1020)
Remote Desktop Protocol (T1076)	Domain Fronting (T1172)	Data Encrypted (T1022)
Remote File Copy (T1105)	Port Knocking (T1205)	Exfiltration Over Alternative Protocol (T1048)
Remote Services (T1021)	Remote Access Tools (T1219)	Exfiltration Over Command and Control Channel (T1041)
SSH Hijacking (T1184)	Remote File Copy (T1105)	Scheduled Transfer (T1029)
Windows Remote Management (T1028)	Uncommonly Used Port (T1065)	Transfer Data to Cloud Account (T1537)

Figure 3: Potential Enterprise ATT&CK Techniques for Zeek

Of the techniques above, those in blue were selected as representative techniques that may be deployed elsewhere in the matrix and are discussed in this research to demonstrate the methodology proposed. Three of these techniques (in green, with Remote File Copy appearing in both the Lateral Movement and Command & Control Tactics) have working scripts that can be used to illustrate a Zeek user's ability to meet ATT&CK detection needs.

Author Name, email@address

### 3.3.1. Lateral Movement (TA0008)

Lateral movement is comprised of the techniques most commonly used to pivot and move within a network. The techniques it covers are most likely to reveal themselves in three major ways: brute-force attacks, strange/deprecated protocol abuse, and unlikely host pair transactions.

First, less sophisticated adversaries or those lacking sufficient reconnaissance work will likely attempt password spraying or brute-force attacks to enumerate credentials, gain access, or map the environment. Efforts to uncover these using Zeek were initially attempted at a macro-level. However, as the various techniques each contained multiple dissimilar protocols, this overarching method was demoted in favor of a per-protocol approach. SSH as part of Remote Services (T1021 & T1210), and both techniques have potential given SSH's broad applicability to most environments, support within default Zeek protocol analyzers, and relative stability in implementation according to standards or published guidelines. Windows Remote Management (WinRM, T1028) could likewise be tackled using Zeek, but this research would only reinvent the hard work done by the BZAR project and others.

Most network and security operators will have some idea as to the protocols expected on their environment, and ideally, strict policies and enforcement in place to ensure adherence. Identifying abuses of deprecated or unwanted protocols on the network can quickly uncover offending users or attackers. While it is unfortunately not the case everywhere, Telnet, unsecured FTP, and plain text HTTP are still widely used protocols, especially in infrastructure systems such as routers, switches, and application servers. Attackers are also likely to attempt the use of these protocols or even their more secure replacements (SCP, Rsync, SFTP) over non-standard ports or unusual host pairings to evade detection or signature-based alarms. Remote File Copy (T1105) was selected as a candidate for detection using Zeek, leveraging the file extraction, connection monitoring, and protocol analysis capabilities.

Identifying unlikely host pairs can depend heavily on the context for some varieties of abuse. Defining roles can be a huge boost to any toolset, but here it can be used to pinpoint endpoints that should not typically request or orchestrate management

connections, yet do. Without that context, there is still more that can be uncovered. In the event of RDP (T1076) or SSH Hijacking (T1184), it should be possible to pin security associations such that a hijacking host could be identified without having a priori knowledge of those roles.

### **3.3.2. Command & Control (TA0011)**

An essential capability of any APT, Command & Control (C2) by its very nature requires the engagement of network resources. C2 is also a focus for obfuscation techniques and deception, as lapses in the C2 approach contribute to attribution and eventual reaction. Though many approaches may be used to identify C2 channels in use, this research focuses on detecting deception methods using two key techniques.

Domain Fronting (T1172) has seen widespread use with the mandated adoption of TLS 1.2+ in most environments coinciding with the heavy dependence on responsive web services on Content Delivery Networks (CDN). MITRE's summary of the technique inspired this approach, which attempts to detect "different domain names in the SNI field of the TLS header and the Host field of the HTTP header" and uncover any potential sessions compromised (MITRE ATT&CK®, 2020).

APTs will often map port use early on to establish their C2 networks and gain persistence within an environment. Port Knocking (T1205) should be detectable, as non-standard TCP exchanges and flag use can be used to identify malicious access attempts or mapping activities.

### **3.3.3. Exfiltration (TA0010)**

While early viruses and worms affected environments primarily via disruption and damage, modern threats are usually motivated by the desire to profit or compromise the users, their organization, or some other entity. While these goals are not mutually exclusive to destruction, APTs will gain persistence and access to steal, copy, or alter information over significant timeframes, improving the impact and return on their investment in the operation. Efforts in detecting exfiltration will benefit from similar role-based context as do other tactics. However, suitable intelligence may also be gleaned through threat feeds & geolocation, anomaly detection, and abnormal transfer fingerprints.

For this research, Exfiltration Over Alternative Protocol (T1048) is explored for compatibility with Zeek's strengths in identifying transactions regardless of expectations or assumptions. Every environment will demonstrate patterns of protocol use between expected host groups. This detection attempt will focus on finding those that occur between unlikely hosts. This technique should be expandable to cover Transfer Data to Cloud Account (T1537) and Exfiltration Over Alternative Protocol (T1048) – hopefully, both are low prevalence activities.

Additionally, while end-users will commonly leverage data transfers during business hours, and servers may employ data transfer in bulk for backup and replication activities, any deviation from those normal patterns should be investigated – especially those that appear to be encrypted. Email and normal application flow to external hosts will often rely on TLS or IPSec encryption for protection. However, attackers are more likely to encrypt specific files before transport to avoid proxies and encrypted session logging. While beacon detection programs like RITA can offer some insight from a pattern perspective, this research explores the Data Encrypted (T1022) technique as a possible detection using Zeek's file extraction capabilities.

### 3.4. Test Approach

The test approach varied by technique, as some test cases were more difficult to perform live in the sandbox and were thus exclusively tested against captures. In either case, Zeek configurations were consistent, except for which scripts were in place during each iteration.

#### 3.4.1. Preparation of VM

Testing was performed using Zeek 3.1.3 installed on an Ubuntu 19.04 VM. Zeek Package Manager was used to install the Zeek TLS Log Alternative (<https://github.com/0xxon/zeek-tls-log-alternative>), which was used to observe and expose the additional TLS details that may be needed for the Domain Fronting technique. Regardless of the test approach used at the time (packet capture playback or live traffic), the Zeek process was started using the `zeekctl deploy` command to ensure all scripts were reloaded from the configuration files and a new set of logs were captured for only that particular scenario. After the playback or live stream had completed, logs were

copied to a use-case-specific directory to facilitate repeatable analysis, and after confirming the logs' archival, the `zeekctl stop` command was used to cease collection. The logs were purged to ensure a clean start before the next run.

For live enactments of the test cases, a sensing interface was designated (`ens192` for this particular VM) and placed on the SPAN interface fed by the Cisco switches to capture live traffic. Live enactment in this lab was used much less than anticipated due to the more complete and repeatable results possible using real-world packet captures.

### 3.4.2. Playback & Logging of Capture

For each technique of interest, a suitable packet capture was located that could offer real-world telemetry for use. These captures were replayed against the loopback interface of the Zeek VM as the sensor without modification or excess command-line switches. For example, `tcpreplay -i lo wmi_exec.pcap` was used to play a particular WMI Exec capture against the testbed. Once logs were collected for each packet capture, these were archived, helping to avoid rerunning them until much further along in the scripting process. Subsequent processing of captures (e.g., during script refinement) took advantage of the Zeek command-line tool.

### 3.4.3. Parsing of Logs using Zeek-Cut

Parsing logs using the straightforward tool Zeek-Cut acted as a form of pseudocode in brainstorming each technique's detection process using Zeek. Using the Domain Fronting technique again as an example, it involves using data available via two different logs – `http.log` and `ssl.log`. Comparing the hostname in `http.log` to that of the `server_name` (SNI) in `ssl.log` allows us to compare the two values and uncover any mismatches that may indicate the nefarious use of CDNs to redirect traffic. For the same Unique ID (`uid`) field, all information should agree, and the certificate used should closely track, if not match, that of the cert-presenting hostname logged by HTTP protocol analyzer. The `zeek-cut` sequences used to winnow the many fields down to those of interest here provide a three-step process to detect:

- 1) Pull `host` and relevant information from HTTP log:



```
cat http.log | zeek-cut -c uid id.orig_h id.resp_h host
>> webdata.txt
```

- 2) Pull `server_name` and relevant information from the SSL log:

```
cat ssl.log | zeek-cut -c uid ts id.orig_h id.resp_h
server_name subject issuer >> certdata.txt
```

- 3) Script a comparison of the `host` field and `server_name` fields per UID to identify potential issues with Domain Fronting. This research leveraged Python to run this comparison.

Once a reasonable algorithm was captured, a representative selection of these were designated for transition to script.

#### 3.4.4. Transition from Zeek-Cut to Script

Transitioning from the Zeek-Cut processes described above to a deployable Zeek script proved to be a varied experience. While the concepts in the prior phase may have been straightforward when parsing the logs, the variety of events and details in handling them proved elusive to a novice scripting practitioner.

Most algorithms generated in the Zeek-Cut phase required data across multiple logs. This wider dataset foretold that a refined script would need to engage multiple Event Engines to aid in the conviction of a flow. Adding to the complexity, each of the Event Engines' generated events had a wide variety of formats and differing levels of detail provided – artifacts that likely arise from multiple contributors offering protocol analysis modules in their own style and image. These intricacies precluded this research from providing finished scripts for all techniques, and in many cases, required each script to focus on addressing only a single protocol or subset of the technique's coverage for demonstration purposes.

### 3.5. Analysis Per Technique – Selected Scripts

An exhaustive breakdown of every technique in the MITRE ATT&CK Enterprise matrix would far exceed this research scope. Instead, what follows are potential methods to approach use cases using each technique, with three pursued to the point of a workable, yet rudimentary script, and preliminary findings and concepts presented for the remaining techniques.

### 3.5.1. Lateral Movement: SSH (T1021) Brute-Force

As part of the Remote Services technique, (<https://attack.mitre.org/techniques/T1021/>) SSH abuses are best mitigated through proper configuration of access lists, timeouts, credential strength, and more rigorous processes built around the sharing of SSH keys between authorized users and the systems they access. Real-world implementations, however, rarely implement more than a couple of those provisions for SSH security. Attackers, knowing that these measures are often misconfigured or missing altogether, will attempt to gain access using brute-force or password spraying techniques.

For this detection method, the pseudocode could be implemented leveraging a single log, the `ssh.log`. Variables like `auth_attempts` vs. `auth_success` offer a potential focal point, but a script should offer some customization to ensure that false positives and sensitivity are correctly balanced. Timestamps (`ts`) and the `uid` values will offer indices that can be used to uncover potential brute-force attempts. A sample `zeek-cut` approach to this might simply pull these fields from the log:

```
cat ssh.log | zeek-cut -c uid ts id.orig_h id.resp_h
auth_attempts auth_success >> ssh_brute_data.txt
```

While researching the SSH protocol analyzer, it was noted that an additional log, `notices.log`, also receives messages when suspected SSH Brute-Forcing is detected and will appear in there if detected. In order to facilitate tuning the parameters that determine whether an alert is generated, a script can load the built-in script available in the base Zeek install (`/policy/protocols/ssh/detect-bruteforcing`). Redefining these variables, it is possible to modify the `SSH::guessing_timeout`, ignore known and approved guessers (`SSH::ignore_guessers`) and define the guessing limit (`SSH::password_guesses_limit`). As a stroke of luck, a simpler version of this script was used as an example in the Notice Framework documentation (`/frameworks/notice`). With some modifications, a script like the following can inject all three tailored parameters into the `detect-bruteforcing.zeek` package and ensure tailored thresholds per the environment's needs:

```

1  ### T1021_ssh_brute.zEEK
2
3  @load base/frameworks/notice
4  @load protocols/ssh/detect-bruteforcing
5
6  ### Redefine the guessing timeout, don't forget the interval units (default minutes)
7
8  redef SSH::guessing_timeout=3 mins;
9
10 ### Identify the authorized guessors (VAM tools, internal pen testers, etc.)
11 ### Use CIDR notation, index is Client, value is Server (table [subnet] of subnet)
12
13 redef SSH::ignore_guessers[172.16.1.0/24] = 172.16.2.0/24;
14
15 ### Redefine the guess limit per unit time - balance between this
16 ### and guessing timeout drives sensitivity
17
18 redef SSH::password_guesses_limit=5;
19
20 event NetControl::init()
21 {
22     local debug_plugin = NetControl::create_debug(T);
23     NetControl::activate(debug_plugin, 0);
24 }
25
26 ### Uses Notice::Policy framework to provide alerting outside of event
27 ### processing engines(https://docs.zEEK.org/en/current/frameworks/notice.html)
28
29 hook Notice::policy(n: Notice::Info)
30 {
31     if ( n$note == SSH::Password_Guessing )
32         add n$actions[Notice::ACTION_LOG];
33 }
34

```

Figure 4: T1021\_ssh\_brute.zEEK

A brute-force attempt was simulated against a switch using incorrect credentials using the host 172.16.11.31 and the data center's default gateway, 172.16.21.1. Random strings were used for password attempts, and four tries were made in each attempt to provide the correct password. Attempts were also made to trigger the alert using each switch as the pivot machine. The notice appeared in notice.log and is seen below. The action in this script can be replaced or augmented by other Notice Framework options, allowing emails to be sent, alerts to be forwarded to SIEMs, or even using APIs to drive automated response.

```

~/Dropbox/SANS_MSISE/ResearchTopic1/Captures/SSH/SSHHome [?] cat notice.log | zEEK-cut -c uid ts note msg sub
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path notice
#open 2020-05-19-13-12-54
#fields uid ts note msg sub
#types string time enum string string
Ciqsf3i2ljXVM3vV6 1589907368.904788 SSH::Invalid_Server_Cert SSL certificate validation failed with (unable to ge
t local issuer certificate) CN=*.events.data.microsoft.com,OU=WSE,O=Microsoft,L=Redmond,ST=WA,C=US
- 1589907473.126231 SSH::Password_Guessing 172.16.11.31 appears to be guessing SSH passwords (seen in 5 connections).
Sampled servers: 172.16.21.1, 172.16.21.1, 172.16.21.1, 172.16.21.1, 172.16.21.1
~/Dropbox/SANS_MSISE/ResearchTopic1/Captures/SSH/SSHHome [?]

```

Figure 5: Logging SSH Brute-Force Attempts notice.log

As per the documentation for the driving event (</base/protocols/ssh/main.zeek.html>), this event involves guesswork while evaluating the length of associated packets. The engine will err on the side of not raising the alert and thus may not seem to register the event. This research concludes that more tuning is required for the underlying SSH protocol analyzer to ensure it can adequately feed scripts offering this coverage.

T1021 acts as an umbrella technique for multiple protocols used in common management schemes. VNC and Telnet are called out specifically. SSH's unique approach above, leveraging built-in brute-forcing detection, could be similarly attempted for VNC with the remote framebuffer protocol (RFB) plugin to offer visibility ([/base/bif/plugins/Zeek\\_RFB.events.bif.zeek](/base/bif/plugins/Zeek_RFB.events.bif.zeek)). Telnet is the odd protocol out here, as there is no dedicated analyzer provided in Zeek. Its plain text and rudimentary nature should, however, be similarly detected using the combination of connection events, port mapping, and unusual host pairs.

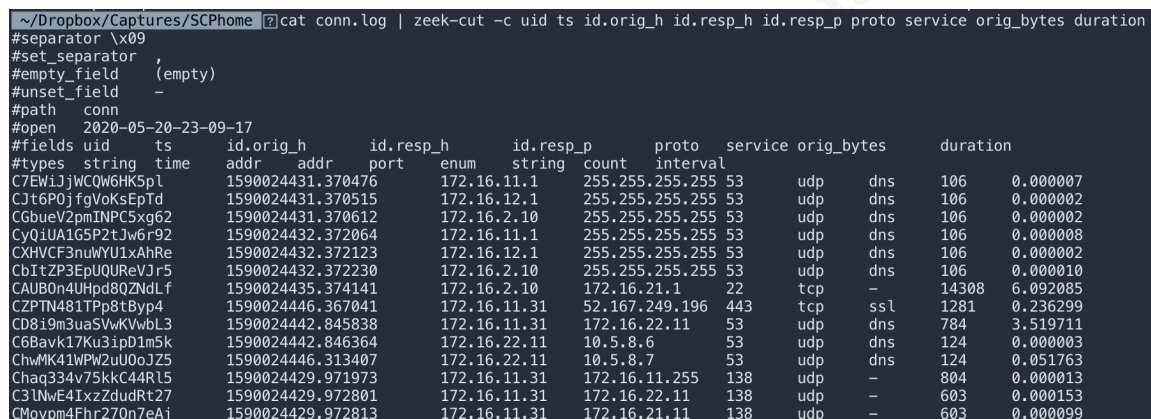
### **3.5.2. Lateral Movement & Command and Control: Remote File Copy (T1105)**

The transfer of files (<https://attack.mitre.org/techniques/T1105/>) using SCP, Rsync, or SFTP is covered in Technique T1105. As with T1021, these are all disparate protocols requiring different approaches to handling. Secure Copy (SCP) uses SSH and can be handled with the same SSH protocol analyzer and help from the `conn.log` file. As SSH hides the traffic within it, there is no possibility of combining it with File Events and other analyzers. The most straightforward method explored would evaluate single SSH connections and evaluate the length, in bytes, and duration of the corresponding flow. For those connections that are not whitelisted and exceed an arbitrarily chosen threshold, the script will flag these connections in the `notice.log`. This threshold should take into account typical SSH workloads, which tend to be on the order of several bytes per minute rather than multiple kilobytes over a few seconds.

The pseudocode process used to locate the parameters needed focuses on finding sessions in the `conn.log` file that look abnormal or unexpected (using a vector of hosts or subnets to be excluded) and filtering those sessions first to eliminate non-SSH sessions

and then filtering out any approved host pairings. The basic zeek-cut processes inspiring this script are shown below:

```
> cat conn.log | zeek-cut -c uid ts id.orig_h id.resp_h id.resp_p
proto service orig_bytes duration
> cat ssh.log | zeek-cut uid ts id.orig_h id.resp_h id.resp_p
```



```
~/Dropbox/Captures/SCPhone [?] cat conn.log | zeek-cut -c uid ts id.orig_h id.resp_h id.resp_p proto service orig_bytes duration
#separator \x09
#set_separator
#empty_field (empty)
#unset_field -
#path conn
#open 2020-05-20-23-09-17
#fields uid ts id.orig_h id.resp_h id.resp_p proto service orig_bytes duration
#types string time addr addr port enum string count interval
C7EWijjWCQW6HK5pl 1590024431.370476 172.16.11.1 255.255.255.255 53 udp dns 106 0.000007
CJt6P0jfgVoKsEpTd 1590024431.370515 172.16.12.1 255.255.255.255 53 udp dns 106 0.000002
CGbueV2pmINPC5xg62 1590024431.370612 172.16.2.10 255.255.255.255 53 udp dns 106 0.000002
CyQiUA1G5P2tJw6f92 1590024432.372064 172.16.11.1 255.255.255.255 53 udp dns 106 0.000008
CXHVCf3nuWYU1xAhRe 1590024432.372123 172.16.12.1 255.255.255.255 53 udp dns 106 0.000002
CbItZP3EpUQUReVjr5 1590024432.372230 172.16.2.10 255.255.255.255 53 udp dns 106 0.000010
CAUBOn4UHpd8QZNdLf 1590024435.374141 172.16.2.10 172.16.21.1 22 tcp - 14308 6.092085
CZPTN481TPp8tByp4 1590024446.367041 172.16.11.31 52.167.249.196 443 tcp ssl 1281 0.236299
CD8i9m3uaSVwKVwbL3 1590024442.845838 172.16.11.31 172.16.22.11 53 udp dns 784 3.519711
C6Bavk17Ku3ipD1m5k 1590024442.846364 172.16.22.11 10.5.8.6 53 udp dns 124 0.000003
ChwMK41WPW2uU0oJZ5 1590024446.313407 172.16.22.11 10.5.8.7 53 udp dns 124 0.051763
Chaq334v75kkC44RL5 1590024429.971973 172.16.11.31 172.16.11.255 138 udp - 804 0.000013
C3lNwE4IxzZdudRt27 1590024429.972801 172.16.11.31 172.16.22.11 138 udp - 603 0.000153
CMovpm4Fhr270n7eAj 1590024429.972813 172.16.11.31 172.16.21.11 138 udp - 603 0.000099
```

Figure 6: Looking for SCP sessions with zeek-cut

Connection events were first evaluated based on whether they contained SSH and were then filtered for any that involved approved management hosts to transition the code into a script. Some sizing and rate-checks were then evaluated to flag flows where a file may have been moved over SSH (thus via SCP or some other nefarious means). The `connection_state_remove` event, as the name should indicate, will occur at the end of the connection and thus include the full population of the many fields that additional analyzers can render. A notional, far-from-production script used in testing this method follows below:

```

1  ##! T1105_remote_file_copy.zEEK
2
3  @load base/frameworks/notice
4  @load base/protocols/ssh
5  @load base/protocols/conn
6
7  ##! Set of variables to tackle this problem
8  ##! Define list of local management boxes approved for SCP/SSH
9  global auth_managers: set[addr] = { 172.16.11.1 };
10
11 ##! Define constraints for what entails a high-rate, suspect flow
12 const max_ssh_duration: interval = 2 min &redef;
13 const max_ssh_size: count = 2300 &redef;
14 const max_ssh_rate: count = 1000 &redef;
15
16 ##! Will ID SSH connections and determine if they are too big or otherwise likely SCP candidates.
17 event connection_state_remove(c: connection) {
18
19     ##! Filter out non-SSH noise looking for presence of appropriate ssh information
20     if ( c?$ssh == F ) {
21         return;
22     }
23     ##! Filter out management noise looking for approved SSH users
24     if ( c$id$orig_h in auth_managers ) {
25         return;
26     }
27     ##! For every connection id associated with a SSH session and not approved,
28     ##! evaluate size and duration to determine if likely SCP.
29
30     ##! Calculate the byterate, aim is to be flag sessions higher than CLI normal, indicating SCP or tunneling
31     local byterate: double;
32
33     ##! protect from Divide-by-Zero
34     if ( |c$duration| == 0 ){
35         byterate = 0;
36     }
37     if ( |c$duration| != 0 ){
38         byterate = c$conn$orig_ip_bytes / |c$duration|;
39     }
40     ##! If an SSH session has a ton of bytes, takes too long, or looks too big, it probably is - send an alert
41     and log it!
42     if ( (c?$ssh == T) && ((c$duration > max_ssh_duration) || (byterate > max_ssh_rate) ||
43         (c$conn$orig_ip_bytes > max_ssh_size))) {
44
45         ##! Print out alert to terminal
46         print fmt ("[WARNING] Potential unauth SCP detected (ATT&CK T1105)");
47         print fmt ("Time: %s TX: %s RX: %s Bytes: %s Duration: %s Connection: %s", strftime("%Y/%M/%d
48             %H:%m:%S", c$conn$time), c$id$orig_h, c$id$resp_h, c$conn$orig_ip_bytes, c$duration, c$uid );
49
50         ##! And capture for posterity in notice.log
51         NOTICE([
52             $note = Weird::Activity,
53             $msg = "Potential unauth SCP detected (ATT&CK T1105)",
54             $conn = c
55         ]);
56     }
57 }

```

Figure 7: T1105\_rfc\_scp.zEEK (SCP use case only)

The results from this code, run against the sandbox switch-to-switch scenario results in a triggered terminal message and a simple entry in `notice.log` with the appropriate details. Multiple thresholds were iterated to verify the script's adjustable sensitivity, and a successful run result is depicted in Figure 8.

```

~/Dropbox/SANS_MSISE/ResearchTopic1/Captures/LateralMover/SCPhome [?] zEEK -C -r scp.pcapng local
WARNING: No Site::local_nets have been defined. It's usually a good idea to define your local networks.
netcontrol debug (Debug-All): init
[WARNING] Potential unauth SCP detected (ATT&CK T1105)
Time: 2020/27/20 21:05:15 TX: 172.16.2.10 RX: 172.16.21.1 Bytes: 149592 Duration: 6.092085 Connection: C2WKVauhayfRTeIT7
~/Dropbox/SANS_MSISE/ResearchTopic1/Captures/LateralMover/SCPhome [?] cat notice.log | zEEK-cut -c uid ts note msg id.orig_h id.resp_h
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path notice
#open 2020-05-21-15-26-19
#fields uid ts note msg id.orig_h id.resp_h
#types string time enum string addr addr
C2WKVauhayfRTeIT7 1590024446.505256 Weird::Activity Potential unauth SCP detected (ATT&CK T1105) 172.16.2.10 172.16.21.1
ChghpV2AMgNCqoXE19 1590024446.505256 SSL::Invalid_Server_Cert SSL certificate validation failed with (unable to get local issuer
certificate) 172.16.11.31 52.167.249.196
~/Dropbox/SANS_MSISE/ResearchTopic1/Captures/LateralMover/SCPhome [?]

```

### Figure 8: T1105 Detection Triggers

It may be tempting to rely on SSH's usual TCP port assignment of 22, but that is what attackers are hoping defenders will do. A complete approach would look for only those uid that correspond with an SSH session as recognized by the protocol analyzer, which does not rely on port mapping but rather a much more thorough parsing of the packet. While Figure 6 shows an obvious candidate for SCP, this research will attempt to build in SSH analyzer hooks to ensure non-standard port use is covered.

Possible expansions of this approach may include detection methods for the other remote file copy processes of interest. Combined with a more dynamic means of feeding context (address mapping to roles, additional correlation with another telemetry, or a richer alert and logging engine), this technique seems ideally suited to the capabilities of Zeek for a production-quality script.

More sophisticated statistical analysis using the `Sumstats` module could help reveal these discrepancies and guard against attackers who anticipate these detection methods by leaving their connections open longer to bring down the average bit rate. Long-term connections are a related issue, assuming those are also being monitored, and this complication can be mitigated. Projects like RITA from Active Countermeasures assist with detecting beacons and long-flows

(<https://www.activecountermeasures.com/free-tools/rita/>).

### 3.5.3. Command and Control: Domain Fronting (T1172)

As outlined in Section 3.4.3, Domain Fronting (<https://attack.mitre.org/techniques/T1172/>) has become a popular technique in circumventing enterprise controls as many firewalls and proxies have had to make concessions for content delivery networks (CDNs). Adhering to the IETF RFCs 6066 (Eastlake, 2011) and 8446 (Rescorla et al., 2020), it should be possible to detect mismatches between the HTTP host field and the `server_name` (SNI) in the SSL information. A preliminary script is available in the repository associated with this research (see the Appendix for the link).

The longevity of any approaches here will be in question as draft-ietf-tls-esni-06 (<https://www.ietf.org/id/draft-ietf-tls-esni-06.txt>) makes recommendations on how to



protect the SNI using encryption (Rescorla, Oku, Sullivan, & Wood, 2020). Ironically, the call for the Encrypted SNI (ESNI) and its separation from the client hello message comes as a result of the ambiguity and lack of convention in dealing with SNI information in earlier versions of TLS. Zeek, however, appeared to take advantage of plentiful SNI visibility methods in earlier versions. With TLS 1.3+, Zeek's inability to proxy traffic will impede efforts to detect the ESNI. Further research will be necessary to find an approach that extends to TLS 1.3 and beyond.

The script provided in the Github repository generates checked results and was one of many that attempted to solve this problem within Zeek alone. It identifies simple TLS 1.1 transactions accurately without CDN involvement, but as the complexity of CDN-like behavior is introduced, false positives undermine the script's utility. The many CDN approaches to caching data and representing their own host and domain naming indicate a need for greater correlation and some deeper understanding of each CDN's conventions. This is certainly an area for later exploration.

### **3.6. Analysis Per Technique – Ideas for Additional Techniques**

While the previous section offered test scripts that delved into how one might detect those techniques, this research also identified several other techniques that are prime candidates for Zeek. This section discusses concepts for further investigation that may offer additional detections and bolster Zeek's value in supporting ATT&CK technique coverage.

#### **3.6.1. Lateral Movement: RDP (T1076)**

The use of RDP (<https://attack.mitre.org/techniques/T1076/>) for lateral movement has a myriad of possible applications. Adversaries may use RDP after gaining a pivot point within a network. Well-defined security policies and role assignment can assist in limiting the exposure by only permitting RDP access from a well-defended subset of hosts, which in turn should have strict access controls and monitoring in place. While enforcement controls are desired, for this detection, it is assumed that there is a means by which hosts can either be excluded from detection (administrative machines approved to use RDP) or included in the list of untrusted hosts.



Zeek's base install includes a detailed record for RDP session information (/base/protocols/rdp). It is envisioned that this record's detailed information could be used in conjunction with the connection record's top-level information to detect RDP sessions between unapproved host pairs (using whitelist/blacklist approach), using downgraded or non-existent security protocols, expired or invalid certs, or even more nefarious techniques, such as the use of non-native keyboard layouts or hijacked cookies from valid sessions elsewhere in the environment. A pseudocode query for this information could begin with the below:

```
Cat rdp.log | zeek-cut -c uid ts id.orig_h id.resp_h
id.resp_p cookie security_protocol keyboard_layout
client_name cert_type cert_count cert_permanent
encryption_level encryption_method
```

It should also be possible to further characterize RDP traffic for context that can better inform the future detection of RDP in a specific environment. A script may use statistical analysis to identify brute-forcing attempts, potential pivot points in the network, relay agents, and other undesired activity in the network.

### 3.6.2. Lateral Movement: SSH Hijacking (T1184)

SSH hijacking (<https://attack.mitre.org/techniques/T1184/>) would be demonstrated by an attacker wishing to take advantage of approved management channels while circumventing the brute-forcing process. Commonly, this will involve attackers gaining a foothold on a host and abusing the pre-existing trust relationships to establish SSH tunnels and perform their objectives. This method will follow similar parameters as those used in the pseudocode process for detecting brute-forcing of SSH credentials:

```
cat ssh.log | zeek-cut -c uid ts id.orig_h id.resp_h
auth_attempts auth_sucess >> ssh_hijacking_raw.txt
```

These data points can then be parsed against a list of approved host group pairings (using a table[subnet] of subnet) to look for client-server pairings that look abnormal or were not disclosed in the configuration of environmental variables.

### 3.6.3. Command and Control: Port Knocking (T1205)

Port knocking (<https://attack.mitre.org/techniques/T1205/>) is a common technique for the enumeration of services and their defenses within an environment. Nmap scans or custom-crafted packet sequences are leveraged to detect port status and assess versions, operating systems, and other valuable intelligence on target networks. Zeek's many protocol-focused analyzers can assist, but the Protocol Independent Protocol Detection (DPD) framework and resultant connection's service field may be areas of focus for future research.

### 3.6.4. Exfiltration: Exfiltration Over Alternative Protocol (T1048)

Adversaries use lesser known protocols or alternative protocol/port/service combinations (<https://attack.mitre.org/techniques/T1048/>) to evade detection while removing data from a target. Detecting these covert flows requires good baseline knowledge and understanding, combined with absorbing that information and using it to feed detections that identify significant data transfer in unusual places. Attention will need to be paid to the soft leaking of information, or the scheduled transfers of small portions of the data. Attackers may use these techniques to avoid triggering alarms or garnering attention.

### 3.6.5. Exfiltration: Data Encrypted (T1022)

As with Exfiltration Over Alternative Protocol (<https://attack.mitre.org/techniques/T1022/>), context and pattern discovery are critical to detection. In this particular case, however, it is possible to make some additional broad-based assumptions. Most legitimate encrypted file transfers will be conducted over observable – and thus, inspectable – protocols (TLS/SSL) or between well-understood host pairings (e.g., backup & storage traffic or mission-critical applications). A detection scheme might leverage the same SSH/SCP detection methodology used before but may be expanded to look for abnormal flow patterns. In addition to the use of context, statistical analyses would greatly improve the efficacy of this approach.

An additional note – Transfer Data to Cloud Account (<https://attack.mitre.org/techniques/T1537/>) would be a fantastic candidate for inclusion into this with the addition of dynamic domain/IP feeds to assist with offloading the logic

within the script and allow checking against the ever-changing list of IPs and domains used by both approved and blacklisted services.

## 4. Recommendations and Implications

Zeek's abilities far surpass the scope of any single project. This research can act as a humble launching point, providing some ideas and a simple methodology for tackling ATT&CK-focused detections. There are multiple paths to pursue from here. It is clear that using NSE techniques and the Zeek tool can create useful detections for several of the MITRE ATT&CK Tactics and their Techniques. Focusing on the underserved endpoint and node populations, such as IoT, ICS, and network infrastructure, this approach can complement more traditional endpoint-focused approaches as well.

### 4.1. Recommendations for Practice

There are issues common to both endpoint and network-based detection methods. Across the three tactics of focus for this research, multiple protocols are masked by deception through non-standard port use, rate limiting, session hijacking, or obfuscation. Much like an endpoint solution, differentiating between these malicious flows and good traffic is aided by context. Providing that context can either require some prior configuration of variables used in the scripts, or in additional processing and intelligence in the scripting to provide baselining of active network nodes and hosts to determine their role. In either instance, production Zeek users will likely need to interact with scripts to either confirm role assignments or provide them outright. Significant work has been performed in open-source and commercial products alike and beyond the scope of this research.

The amount of state required to process these complex relationships also must be considered. Production script approaches may wish to factor in the use of databases or additional helper functions (such as coded libraries, algorithms) to facilitate reuse, promote modularity, and consolidate like functions. While tables can be declared and passed within Zeek scripts, longer-term storage, grander scale, and advanced search, sort, and correlation can be achieved through the integration of a database. A valid, and for many, preferred approach is to parse and forward logs to SIEMs like Splunk or

Elasticstack, pushing all available log entries into the SIEM and then correlating appropriate fields. Native scripting within Zeek would still be preferred by engaging additional protocol analyzers or specialized/tailored frameworks, but strictly, for analysis, either can be used to significant effect. The determination will likely hinge on training, experience, and the correlation with external tools.

## 4.2. Implications for Future Research

There is potential to both converge and expand detection methods – to generalize them such that fewer scripts provide more comprehensive coverage of techniques and sub-techniques. This research explored specific use cases to infer acceptable detection rates and certainty. This narrow focus was not limited to the per-technique detections, but necessary to break the technique itself down on a per-protocol (e.g., SCP vs. Rsync or SFTP) or variant basis (e.g., brute-force vs. hijacking vs. unlikely hosts) for analysis. Expansion of the connection-based events and the expansion of its available dataset opens the door to making more broadly applicable scripts with minimal reinvention.

Much remains to be done in providing a multi-factor correlation of events. Sophisticated threat actors take advantage of myopic detection methods and enforcement tools. Building in more advanced analyses, cross-protocol validation of observations, and integration with threat feeds could ensure that a feature-rich implementation does not adversely impact the operator's workload or the system's security coverage. This research has barely scratched Zeek's extensive set of frameworks, policy analyzers, and data structures – increased familiarity and collaboration in this pursuit would likely accelerate with additional capable hands.

Lastly, Zeek's flexibility to log, alert, or otherwise integrate with down-stream consumers of its parsed information leaves the door open to the automation and standardization of those subsystems to meet a wide variety of use cases. Per-script notification methods were used in this research. However, script elegance and the user experience would be improved through a centrally configured and full-featured notice, alert, and logging approach that allows both stand-alone and highly integrated rollouts.

## 5. Conclusion

The focus on MITRE ATT&CK for a wide variety of information security needs, including solution efficacy, is well warranted. A very small number of organizations can afford to develop their threat intelligence, and even those organizations have turned to feeds and guidance from MITRE ATT&CK as a means of improving their outcomes. Manufacturers and suppliers of security solutions have quickly moved to adopt ATT&CK's many facets in their solutions and marketing. The sourcing of data correlated in this framework is heavily endpoint-based. Given the expansion of ICS and IoT use cases, more commercial and open-source solutions could stand to benefit by considering that wider array of potential targets. Not all techniques and tactics are attainable, yet many do offer some discernable network footprint. Using Zeek's vast capabilities and some ingenuity, it is possible to cover of significant portions of the MITRE ATT&CK Enterprise Matrix. This approach expands its applicability well beyond "enterprise" endpoints and assists environments of all types and compositions in achieving a more secure environment.

## References

- Bursztein, E. (2017, December 6). Inside Mirai the infamous IoT Botnet: A Retrospective Analysis. Retrieved May 13, 2020, from <https://elie.net/blog/security/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/>
- Eastlake, D. (2011, January). Transport Layer Security (TLS) Extensions: Extension Definitions. Retrieved from <https://tools.ietf.org/html/rfc6066>
- Fernandez, M., Wunder, J., Azoff, J., & Tylabs. (n.d.). mitre-attack/bzar. Retrieved March 2020, from <https://github.com/mitre-attack/bzar>
- The Honeynet Project. (1999-2019). The Honeynet Challenges. Retrieved March 11, 2020, from <https://www.honeynet.org/challenges/>
- Karimi, A. M., Niyaz Q., Sun, W., Javaid, A. Y. & Devabhaktuni, V. K., "Distributed network traffic feature extraction for a real-time IDS," *2016 IEEE International Conference on Electro Information Technology (EIT)*, Grand Forks, ND, 2016, pp. 0522-0526, doi: 10.1109/EIT.2016.7535295.
- Largent, W. (2018, May). New VPNFilter malware targets at least 500K networking devices worldwide. Retrieved May 13, 2020, from <https://blog.talosintelligence.com/2018/05/VPNFilter.html>
- MITRE ATT&CK®. (2020, March 20). Retrieved from <https://attack.mitre.org/>
- Rescorla, E. (2018, August). The Transport Layer Security (TLS) Protocol Version 1.3. Retrieved from <https://tools.ietf.org/html/rfc8446>
- Rescorla, E., Oku, K., Sullivan, N., & Wood, C. A. (2020, March 9). Encrypted Server Name Indication for TLS 1.3. Retrieved from <https://tools.ietf.org/html/draft-ietf-tls-esni-06>

- Slowik, J. (2018, October 12). Anatomy of an Attack: Detecting and Defeating CRASHOVERRIDE. Retrieved from <https://dragos.com/resource/anatomy-of-an-attack-detecting-and-defeating-crashoverride/>
- Stratosphere. (2015). Stratosphere Laboratory Datasets. Retrieved March 13, 2020, from <https://www.stratosphereips.org/datasets-overview>
- Stoll, C. (1989). The Cuckoos Egg: Tracking a Spy Through the Maze of Computer Espionage (1st ed.). New York, NY: DoubleDay.
- Strom, B. E., Battaglia, J. A., Kemmerer, M. S., Kupersanin, W., Miller, D. P., Wampler, C., ... Wolf, R. D. (2017, June). Finding Cyber Threats with ATT&CK™-Based Analytics. Retrieved April 2020, from <https://www.mitre.org/publications/technical-papers/finding-cyber-threats-with-attck-based-analytics>
- Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018, July). MITRE ATT&CK: Design and Philosophy. Retrieved March 2020, from [https://attack.mitre.org/docs/ATTACK\\_Design\\_and\\_Philosophy\\_March\\_2020.pdf](https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf)

## Appendix: Resources & Links

Publicly Available Packet Captures	
Publicly available PCAP files Aggregated by NETRESEC	<a href="https://www.netresec.com/?page=PcapFiles">https://www.netresec.com/?page=PcapFiles</a>
Over 1600 packet captures built by brad@malware-traffic-analysis.net	<a href="https://www.malware-traffic-analysis.net">https://www.malware-traffic-analysis.net</a>
Stratosphere Laboratory Datasets	<a href="https://www.stratosphereips.org/datasets-overview">https://www.stratosphereips.org/datasets-overview</a>
Honeynet Challenges from the Honeynet Project	<a href="https://www.honeynet.org/challenges/">https://www.honeynet.org/challenges/</a>
LIVE DOWNLOAD LINK for DefCon 23 ICS Village Packet Captures (RAR file)	<a href="https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20villages/DEF%20CON%2023%20ics%20village/DEF%20CON%2023%20ICS%20Village%20packet%20captures.rar">https://media.defcon.org/DEF CON 23/DEF CON 23 villages/DEF CON 23 ics village/DEF CON 23 ICS Village packet captures.rar</a>

MITRE & Zeek Documentation Links	
MITRE ATT&CK Project	<a href="https://attack.mitre.org">https://attack.mitre.org</a>
Zeek's Framework Documentation (including File Analysis, Logging, Notice)	<a href="https://docs.zeek.org/en/current/frameworks/index.html">https://docs.zeek.org/en/current/frameworks/index.html</a>
Zeek's Script Reference - in-depth dictionary of all data structures, events, analyzers, etc.	<a href="https://docs.zeek.org/en/current/script-reference/index.html">https://docs.zeek.org/en/current/script-reference/index.html</a>

Link to Repositories	
GitHub Repository	<a href="https://github.com/mjmcphree/zeek-attack">https://github.com/mjmcphree/zeek-attack</a>
Excellent Bro IDS programs collection by Michael Purzynski @ Mozilla	<a href="https://github.com/michalpurzynski/bro-gramming">https://github.com/michalpurzynski/bro-gramming</a>
MITRE Caldera Project	<a href="https://github.com/mitre/caldera">https://github.com/mitre/caldera</a>
Red Canary's Atomic Red Team Project	<a href="https://github.com/redcanaryco/atomic-red-team">https://github.com/redcanaryco/atomic-red-team</a>