Fight crime.
Unravel incidents... one byte at a time.

Interested in learning more?
Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (FOR508)
ATT&CK-Based Live Response for GCP CentOS Instances

GIAC (GCFA) Gold Certification

Author: Allen Cox, alcox.infosec@gmail.com
Advisor: Sally Vandeven

Accepted: 24 June 2020

Abstract

As organizations increasingly invest in cloud service providers to host data, applications, and services, incident responders must detect and respond to malicious activity across several major platforms. With nearly one-third of the cloud infrastructure market share, Amazon Web Services (AWS) dominates the information security scientific literature. However, of the other major cloud providers, Google Cloud Platform (GCP) experienced the most significant annual growth in 2019 (Canalys, 2020), and as a result, defenders can expect to respond more frequently to incidents in GCP. This research examines the data sources available to responders on GCP CentOS compute instances and within the cloud platform. Using MITRE ATT&CK to identify attacker tactics and Red Canary’s Atomic Red Team to generate test data, this research proposes a live response script to collect the essential data that responders will need to identify the discussed tactics.
1. Introduction

The public cloud is an increasing aspect of many organizations’ information technology portfolios. Developers and administrators, especially those empowered by their organization to make on-the-fly changes or implement test environments, can inadvertently compromise the security of their organization’s networked resources. In the SANS 2019 Cloud Security Survey, 40.3% of respondents listed “unauthorized (rogue) application components or compute instances” as a “major concern” for their organization (2019). Furthermore, 46.3% also listed “poorly configured or insecure interfaces or APIs” (Shackleford, 2019, p. 5). These results indicate that many organizations are not confident in their ability to review and secure their cloud resources. When a successful attack makes these fears a reality, incident responders must attempt to reconstruct the attacker’s actions, regardless of the monitoring capabilities available.

Among the major players in the cloud infrastructure industry, Amazon Web Services (AWS) held an estimated 32.3% market share lead in 2019. Microsoft’s Azure trailed with 16.9% of the market. In third place, Google Cloud Platform (GCP) held a 5.8% share. Notably, GCP’s annual growth from 2018 was an impressive 87.8% (Canalys, 2020). Information security literature understandably reflects AWS’s market dominance. Of the 21 “Cloud Computing” papers published in the SANS Reading Room during 2019, 14 papers were AWS-specific investigations (SANS Institute, 2020). However, as GCP’s market share continues to grow, defenders will need to be prepared to respond to future incidents on the platform.

In *Incident Response & Computer Forensics, Third Edition*, live response is highlighted as an evidence collection technique that can yield quick answers to “investigative questions” (Luttgens, Pepe, & Mandia, 2014). Live response is usually an automated process, and the data collected may or may not be volatile. In addition to providing some initial confirmation that a system is compromised, evidence from live response results can assist responders in determining the scope of an incident, expose new leads to investigate, enable indicator development, and illustrate an attacker’s activity (Luttgens, Pepe, & Mandia, 2014, p. 37). Using MITRE ATT&CK to explore specific adversarial techniques, this research will identify the live response data that

Allen Cox, alcox.infosec@gmail.com
responders should collect from GCP CentOS instances to identify an attacker’s use of these techniques. This research assumes a default GCP configuration and should be useful for responders regardless of the security controls or monitoring in place on a compromised GCP instance.

2. Google Cloud Platform

Like its competitors, GCP provides a staggering number of tools and features. This research will not attempt to review all these capabilities and their implications but will instead focus on the commonly used GCP Compute Engine and Cloud Storage services. This research will also assume that defenders respond to these capabilities in a default state, meaning that the owning organization has not architected or implemented security controls and monitoring that may assist responders during an incident. The following section will review background information relevant to understanding the interactions between compute instances and other GCP resources.

2.1. The Resource Hierarchy

GCP resources, such as compute instances and storage buckets, are the building blocks that form the GCP resource hierarchy, shown in Figure 1. The organization is the root of the hierarchy, and the project is the mandatory “base-level organizing entity” for utilizing resources (Google Cloud, 2020-a). Organizations can group projects using folders, which they can optionally leverage to implement access controls according to the business unit or team. According to Google, benefits of the resource hierarchy include attaching a resource to the lifecycle of its parent and enabling policy inheritance for all resources belonging to an organization (Google Cloud, 2020-a).
Cloud Identity and Access Management (IAM) is the mechanism through which organizations can set access controls on their resources. Groupings of permissions, called a role, grant resource access to the member. Members are the accounts that utilize resource access, such as user accounts and service accounts. The roles are assigned to members using policy. Cloud IAM supports three types of roles: primitive, predefined, and custom (Google Cloud, 2020-b). Most organizations will want to utilize predefined and custom roles, which allow more granularity. Of interest to this research, however, are the default primitive roles: owner, editor, and viewer. These roles grant broad permissions; organizations that have not configured IAM policy will utilize them by default in newly created projects.

2.2. Instances and Service Accounts

All new projects come with a “Compute Engine default service account” (Google Cloud, 2020-c). This account is assigned the primitive editor role, granting broad account permissions to create, modify, or delete most resources in the project (Google
Cloud, 2020-d). By default, Google assigns this service account to newly created GCP instances. On these instances, however, the account is not configured with the appropriate access scopes to utilize cloud platform APIs. When creating a new instance in the GCP Compute Engine, the user may select a new instance’s service account and set the service account’s API access, called “access scopes.” As shown in Figure 2, GCP selects the “Allow default access” option by default.

Figure 2 – Instance Creation Identity and API Options

Google recommends selecting “Allow full access to all Cloud APIs” as a best practice. Rather than relying on access scopes as a security control, administrators should use predefined or custom IAM roles to limit an account’s access to other services (Google Cloud, 2020-c).

When a user creates a new instance in a project, that instance joins the default Virtual Private Cloud (VPC) network, and the instance receives an internal and external IP address. Instances located in the same geographical region share the same subnet for internal addresses (Google Cloud, 2020-e). The default VPC also applies four default “allow” firewall rules on ingress traffic: any internal connections, any port 22 (SSH), any port 3389 (RDP), and any ICMP (Google Cloud, 2020-f).

2.3. GCP Command-Line Tools

The gcloud command-line interface (CLI) enables users to “create and manage Google Cloud resources” (Google Cloud, 2020-g). The tool organizes its commands into product-related or functional groupings, like the Microsoft Windows “net” command or the AWS CLI. Google includes the gcloud CLI by default on new instances created from

Allen Cox, alcox.infosec@gmail.com
public images. While it is a powerful tool for project administrators, gcloud can equally empower attackers to interface with a project’s resources. Consider a scenario where an attacker has gained access to a compute instance with full API access and a service account with the editor role, the default service account configuration. The attacker could move laterally across all project resources, create new instances or accounts, modify roles and policies, and destroy resources.

In addition to gcloud, gsutil is an open-source application that only interacts with Cloud Storage, enabling manipulation of objects stored in a bucket and configuration of the buckets themselves (Google Cloud, 2020-h). Of note, gcloud does not offer equivalent features to interact with storage buckets. Google includes gsutil by default on new public image instances. The tool’s subcommands follow familiar Linux command-line conventions for interacting with filesystems, such as `cat`, `ls`, `mv`, and `rm`.

3. MITRE ATT&CK Testing

MITRE ATT&CK is an industry-standard collection of attacker techniques; each technique describes how attackers may implement it, identifies threats that have employed it, and finally, recommends mitigations and detections (The MITRE Corporation, 2020). While ATT&CK does not identify every possible technique an attacker may utilize, it provides a common baseline of tactics to which organizations should strive to mitigate, detect, and respond. This research explored nine ATT&CK techniques from among the Execution, Persistence, Discovery, and Collection tactics groups.

3.1. Testing the ATT&CK Techniques

The research experiment generated test data for each of the techniques using Red Canary’s open-source Atomic Red Team, a collection of “small, highly portable detection tests” that Red Canary has mapped against ATT&CK (Red Canary, 2020-a). Using these targeted tests in a laboratory environment helped illustrate which data defenders should collect to detect a given technique.

Allen Cox, alcox.infosec@gmail.com
While Atomic Red Team contains numerous tests for Linux-specific techniques, it does not contain any cloud technique tests. As discussed previously, the body of information security literature on GCP is limited. Fortunately, Chris Moberly recently authored an overview of how an attacker may be able to leverage access from a compromised GCP instance (2020). The experiment uses Moberly’s suggested commands to generate test data when exploring cloud-centric techniques.

3.2. GCP Laboratory Environment

The laboratory environment, illustrated in Figure 3, mimicked a hypothetical, default project that a developer would create for testing. The environment consisted of two GCP Compute Engine virtual machines and one storage bucket in a project called “STI MITRE Research.” The machines were general-purpose n1-standard-1 (1 vCPU, 3.75 GB memory), and both instances used the CentOS 7 GCP public image with a default 20 GB boot disk. GCP assigned the Compute Engine default service account to both instances, and the developer set API access scopes to “Allow full access to all Cloud APIs.” While GCP does not select this access scope by default, it is Google’s recommendation and a feature that a developer would likely enable in a testing environment (Google Cloud, 2020-c). Failing to select the option would prevent the user on that instance from accessing that majority of the gcloud command-line tool’s features.

![Figure 3 – GCP Laboratory Environment](image)

Allen Cox, alcox.infosec@gmail.com
The first CentOS 7 instance was called “clean-instance” and was used to generate baseline results using the live response script. The second instance, called “compromised-instance,” was used to run the attacker tests described in the previous section and collect evidence using the live response script. Finally, the environment had a storage bucket with a file that one of the test techniques stole. This experimental scenario assumes that the attacker gained access to an account with *sudo* privileges on the compromised instance.

4. Techniques, Evidence, and Live Response

Using a selection of techniques from MITRE ATT&CK, this section will explain each technique, describe the test used to implement the technique in the laboratory environment, and discuss the data sources a defender should collect as evidence that an attacker utilized the technique. The discussion of each technique includes a block with Bash commands that responders could include in a live response script.

When faced with a situation where a GCP Compute Engine Instance is compromised, defenders must consider the evidence available on both the instance itself and any logs preserved in the cloud platform. The ATT&CK tactics discussed represent a selection of techniques an attacker may utilize on a compromised GCP Linux instance.

4.1. T1059 – Command-Line Interface

The command-line interface, or shell, enables administrators to interact with an operating system, often remotely. Once an adversary gains access to a system, they can leverage the command-line to execute code, discover resources, and collect data of interest (The MITRE Corporation, 2020).

For technique T1059, “Atomic Test #1 – Command-Line Interface” uses *curl*, a data transfer tool frequently used for accessing files on HTTP and other file servers. The adversary downloads a shell script from a web server and executes the script in Bash via the pipeline (Red Canary, 2020-b). The implementation of this test illustrates two benefits adversaries gain when interacting with the command-line. First, most Linux distributions contain a common subset of powerful tools, such as *curl*. Using tools

Allen Cox, alcox.infosec@gmail.com
natively available on the operating system is frequently referred to as “living off the land” (Tancio, 2020) and may help the attacker blend in with regular administrator activity. Second, the implementation leverages Bash’s capabilities as a scripting language. Using a script, the attacker can quickly deploy pre-written, customized actions that are less prone to error. Section 4.3 further explores the use of scripting.

Unfortunately for defenders, GCP does not natively log command execution on Compute Engine instances. The defender must either leverage existing operating system features or install specialized tools to monitor command-line interaction. GCP’s default instance of CentOS contains three potential files of interest for monitoring command-line activity: audit.log, secure, and .bash_history. In its default configuration, the information in audit.log is insufficient to identify attacker activity. While defenders could configure auditd to monitor the relevant data, a discussion on adequately configuring this service is beyond the scope of this research. The secure log is useful for identifying the execution of commands that a user has run with sudo but does not log commands that users run outside of the sudo context. Finally, the .bash_history file may hold a wealth of information about commands that users have run in their user context. The .bash_history file is not protected, however, and attackers do not require any elevated permissions to delete or tamper with the file.

Each technique discussed in this section relies on some command-line component. Consequently, responders should invest significant effort towards reconstructing the commands an adversary ran on the compromised host. Figure 4 contains the Bash commands utilized in the experimental live response script.
Collecting this data was partially successful in the experimental results. While Figure 5 shows that the `.bash_history` file recorded the attacker’s commands in sequence, it does not provide enough information to create a full timeline of the attacker’s commands.

Since CentOS does not timestamp the history file by default, responders may not be able to correlate events when the attacker has switched between multiple user contexts, like running `sudo su -`, for example. However, switching user contexts with `sudo` did create a timestamped entry in the `secure` log (shown in Figure 6) and could be useful in cases where confirmed attacker commands precede the user context switch.

### 4.2. T1168 – Local Job Scheduling

An adversary may utilize `cron` to achieve command execution at a specified interval. They may choose this technique to establish persistence on system boot,
conduct scheduled check-ins, or execute code in a different user context for privilege escalation (The MITRE Corporation, 2020).

For technique T1168, “Atomic Test #2 – Cron – Add script to cron folder” adds an unwanted script to the /etc/cron.daily/ folder (Red Canary, 2020-b). As written, this Atomic Test does not give execute permissions to the script. The attacker would need to make this modification to implement the tactic successfully. Consequently, the experimental test added this step. Using this technique could allow an attacker to reestablish persistence on a system every day, as long as the attacker has the root privileges necessary to write a file to the cron.daily folder. While overcoming file protections is trivial in the case of administrator credential compromise, it may create a barrier in cases where the attacker has exploited a correctly configured web application that follows the principle of least privilege.

By default, the GCP CentOS installation contains three files in the cron.daily folder: 0yum-daily.cron, logrotate, and man-db.cron. Defenders can collect a current listing of files in the cron.daily folder and compare it to these defaults or another known-good state for their organization. Collecting a copy of any anomalous files could greatly assist defenders in identifying how the attacker was leveraging this technique. Figure 7 contains the Bash commands utilized in the experimental live response script.

```
1 # T1168 - Local Job Scheduling
2 mkdir /tmp/liveresponse/T1168
3
4 # Collect cron.daily file listing
5 ls -la /etc/cron.daily > /tmp/liveresponse/T1168/cron.daily_listing.txt
6
7 # Get hashes for each of the cron.daily files
8 touch /tmp/liveresponse/T1168/cron.daily_hashes.txt
9 for TASK in /etc/cron.daily/*; do
10    if [ -f $TASK ]; then
11       sha256sum $TASK >> /tmp/liveresponse/T1168/cron.daily_hashes.txt
12     fi
13 done
```

Figure 7 – Collect Local Job Scheduling Evidence

Allen Cox, alcox.infosec@gmail.com
Collecting the hashes and file listing of the `cron.daily` folder was sufficient to identify an anomalous script relative to the baseline. Any modification to one of the files will result in a change to the hash. Therefore, the hashes enabled confirmation that the attacker did not tamper with the default scripts.

### 4.3. T1064 – Scripting

As discussed in Section 4.1, many shells, like Bash, are also scripting languages. As such, adversaries can leverage this capability to automate tasks or take rapid and simultaneous actions. Depending on the scripting language used, attackers may bypass security monitoring (The MITRE Corporation, 2020). For example, an organization with extensive monitoring on the user’s login shell may have less thorough reporting on processes executed from a Python interpreter shell.

For technique T1064, “Atomic Test #1 – Create and execute bash shell script” writes a simple script to a file called `art.sh`, grants the file execute permissions, and runs the file (Red Canary, 2020-b). While the title of the test indicates this implementation uses a Bash script, the actual test provided utilizes a Bourne shell script. The difference is an important distinction since running the script from the Bourne shell may impact the artifacts available to the defender. However, the Atomic Test, as written, should still capture the attacker’s execution of the script in `.bash_history`.

Unless an existing logging source captures the script’s initial creation, modification, or execution, identifying the script without a file name may be challenging using default tools. Defenders could consider collecting a recursive listing of file names, permissions, and timestamps from writeable directories, such as `/tmp` or `/home`, looking for new files with execute permissions set. Then, defenders could take subsequent action to collect files of interest. This technique’s success depends on several factors, including the length of time since the original attacker activity and the frequency with which applications and users write scripts to disk. Identifying the script and collecting a copy may not be feasible in a single live response action. In such cases, taking a snapshot of the GCP instance and conducting more in-depth forensic analysis may be necessary.

Figure 8 contains the Bash commands utilized in the experimental live response script.

Allen Cox, alcox.infosec@gmail.com
For this experiment, collecting a listing of executable scripts was sufficient to identify the attacker script, `art.sh`. Note that the live response script will also appear in the results if saved as an executable. On hosts that have a higher volume of script creation and deletion, as responders might expect on a development system, using this evidence would be only partially successful.

### 4.4. T1156 – `.bash_profile` and `.bashrc`

The `.bash_profile` and `.bashrc` files are scripts located in a user’s home directory. The `.bash_profile` script executes when a user logs in to the console interactively; whereas, the `.bashrc` file executes whenever the user opens a new shell. An adversary could modify these files to execute commands that establish persistence (The MITRE Corporation, 2020). For example, consider a user establishing a remote session through SSH on a compromised host. The attacker-modified `.bash_profile` script will run, executing a malicious binary the attacker placed in the user’s home directory.

For technique T1156, “Atomic Test #1 – Add command to `.bash_profile” appends a command calling an arbitrary Python script at the end of the current user’s `.bash_profile` script (Red Canary, 2020-b). Since each user has a unique copy of the `.bash_profile` script, elevated permissions are not necessary to modify this file.

Defenders should collect copies of the `.bash_profile` and `.bashrc` files when gathering live response data, checking for any anomalous additions or modifications to the files. The default GCP `.bash_profile` and `.bashrc` files are quite short. The `.bash_profile` script only runs `.bashrc` and exports the PATH, while the `.bashrc` file only runs the `/etc/bashrc` file. The `/etc/bashrc` file is significantly longer but well-commented. To modify the `/etc/bashrc` file, the attacker would need to use root

---

Allen Cox, alcox.infosec@gmail.com
permissions. Figure 9 contains the Bash commands utilized in the experimental live response script.

```
# T1156 - .bash_profile and .bashrc
mkdir /tmp/liveresponse/T1156

# Collect .bash_profile and .bashrc files and hashes from all users with home directories
cd /home
touch /tmp/liveresponse/T1156/bash_hashes.txt
for FOLDER in *; do
  echo $FOLDER >> /tmp/liveresponse/T1156/bash_hashes.txt
  if [-f "$FOLDER/.bash_profile" ]; then
cp $FOLDER/.bash_profile /tmp/liveresponse/T1156/bash_profile_$FOLDER.txt
  sha256sum $FOLDER/.bash_profile >> /tmp/liveresponse/T1156/bash_hashes.txt
  fi
  if [-f "$FOLDER/.bashrc" ]; then
cp $FOLDER/.bashrc /tmp/liveresponse/T1156/bashrc_$FOLDER.txt
  sha256sum $FOLDER/.bashrc >> /tmp/liveresponse/T1156/bash_hashes.txt
  fi
done

# Root
if [-f "/root/.bash_profile" ]; then
cp /root/.bash_profile /tmp/liveresponse/T1156/root_profile_root.txt
  sha256sum /root/.bash_profile >> /tmp/liveresponse/T1156/bash_hashes.txt
  fi
  if [-f "/root/.bashrc" ]; then
cp /root/.bashrc /tmp/liveresponse/T1156/rootrc_root.txt
  sha256sum /root/.bashrc >> /tmp/liveresponse/T1156/bash_hashes.txt
  fi
```

Figure 9 – Collect .bash_profile and .bashrc Evidence

Identifying the modified script was trivial using the collected hashes. As illustrated in Figure 10, the .bashrc file, which the experimental testing did not modify, has the same hash between the two user accounts.

```
$ cat bash_hashes.txt

$cat compromised-instance T1156 /

<table>
<thead>
<tr>
<th>Hash</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0A7c216e9938367fcd2f072285f6d53eefb2a4baf8835415b965e029ca20</td>
<td>.bash_profile</td>
</tr>
<tr>
<td>898ed3bdc749c665866ee2750ab50d7ac5da6b666564fcd952cfc44c0033b4</td>
<td>.bashrc</td>
</tr>
<tr>
<td>ea5b6e00e8791f626fbca9c3d6aa348336bca49321bfcdd00929630146dd97</td>
<td>.bash_profile</td>
</tr>
<tr>
<td>898ed3bdc749c665866ee2750ab50d7ac5da6b666564fcd952cfc44c0033b4</td>
<td>.bashrc</td>
</tr>
</tbody>
</table>

```

Figure 10 - .bash_profile and .bashrc Hashes from User Folders

The .bash_profile script, which the test modified on the definitelynotahacker account, has a different hash than the default file in the developer’s account. Since the live response script also collected the files, responders could then investigate the attacker’s changes to the script.

Allen Cox, alcox.infosec@gmail.com
4.5. T1136 – Create Account

While adversaries often leverage legitimate accounts when credentials are available, some actors may choose to create local accounts to retain access on a system without deploying “persistent remote access tools” (The MITRE Corporation, 2020). An attacker will need elevated privileges to create a local user account. The default SSH server configuration on the GCP CentOS image prohibits password-based authentication. Therefore, an adversary wishing to utilize their newly created account will need to either enable password authentication in `/etc/ssh/sshd_config` or generate SSH keys. An attacker may also attempt to create a cloud service account to access the cloud platform APIs and thereby an organization’s other GCP resources.

For technique T1136, “Atomic Test #1 – Create a user account on a Linux system” calls the `useradd` command to create the account `evil_user`. As mentioned previously, a privileged user must run this command (Red Canary, 2020-b).

Defenders can use the contents of `/etc/passwd` and a listing of the `/home` directory to identify anomalous local accounts. In cases where an actor has created an anomalous local account, defenders may want to look for changes to the `/etc/ssh/sshd_config` file or SSH keys stored in the home directory of the account. As with other techniques, responders can identify the commands leveraged to create the account using the evidence discussed in T1059 – Command-line Execution.

In Moberly’s GCP Red Team exercise, he provides a script to generate and add a new user account to the GCP instance metadata. Running the script will create the user account and grant `sudo` permissions. Modifying the instance metadata for other instances in the project can allow the attacker to move laterally (Moberly, 2020).

Responders can look for evidence of SSH key creation and check the gcloud logs for the string “`gcloud.compute.instances.add-metadata`” in addition to the terms “`metadata-from-file`” and “`ssh-keys`.” The gcloud logs are discussed further in Section 4.8. Finally, the attacker-created username should appear in `/etc/passwd`.

Moberly’s script will also create a folder for the new user in the home directory. Figure 11 contains the Bash commands utilized in the experimental live response script.
ATT&CK-Based Live Response for GCP CentOS Instances

Figure 11 – Collect Create Account Evidence

With a baseline or a listing of user accounts expected in a GCP project (found in the GCP console under IAM & Admin → IAM), the /etc/passwd file should help responders identify anomalous user accounts. As shown in Figure 12, using gcloud logging to identify the attacker tampering with the instance metadata is trivial.

The log does not include the account name, but it does specify that gcloud loaded the metadata content from meta.txt. Responders can search for this file, which will contain the username. Of note, the GCP default activity log (found in the GCP console under Operations → Logging → Logs Viewer) recorded that the metadata was modified but did not provide enough contextual information to distinguish the attacker’s modification from any routine changes the platform made to the instance.

4.6. T1501 – Systemd Service

Like cron in T1168 – Local Job Scheduling, adversaries can leverage systemd to achieve persistence during system initialization on modern Linux OSs. While creating new services or making system-wide modifications to existing services usually requires elevated privileges, an attacker could create services in a user’s home directory with that user’s permissions (The MITRE Corporation, 2020).

For technique T1501, “Atomic Test #1 – Create Systemd Service” creates a system_service_file in /etc/systemd/system that describes the service’s behavior. In this case, the service only creates files in the /tmp directory, naming the file after the service’s associated action. For example, the service will create a file named

Allen Cox, alcox.infosec@gmail.com
/tmp/art-systemd-execreload-marker when the service restarts. After creating the file to describe the service, the test uses the systemctl command to enable and start the service (Red Canary, 2020-b).

MITRE recommends exploring the /etc/systemd/system/, /usr/lib/systemd/system/, and /home/<user>/.config/systemd/user/ directories for files that an adversary may have created or modified. Additionally, responders should consider collecting a live listing of services using the command systemctl list-units --type=service -all. For each of these recommendations, a system baseline can help defenders identify anomalous services (The MITRE Corporation, 2020). Instances started from the same image in GCP should provide defenders with a good reference point. Figure 13 contains the Bash commands utilized in the experimental live response script.

```
1 # T1501 - System Service
2 mkdir /tmp/liveresponse/T1501
3
4 ## Collect a live service listing
5 systemctl list-units --type=service -all > /tmp/liveresponse/T1501/live_service_list.txt
6
7 ## Collect MITRE-recommended file listings
8 ls -la /etc/systemd/system/ > /tmp/liveresponse/T1501/etc_systemd_listing.txt
9 ls -la /usr/lib/systemd/system/ > /tmp/liveresponse/T1501/usr_systemd_listing.txt
10 cd /home
11
12 for FOLDER in *; do
13  if [ -d "FOLDER/.config/systemd/user" ]; then
14   ls -la FOLDER/.config/systemd/user > /tmp/liveresponse/T1501/config_systemd_listing_FOLDER.txt
15  fi
16 done
```

Figure 13 – Collect Systemd Service Evidence

Analyzing the results with diff illuminated the attacker’s new service when comparing listings from the compromised system to the clean baseline. In cases where developers have heavily modified the compromised system from the public image, utilization of the baseline may be less beneficial, and additional analysis would be required. Regardless, a baseline should significantly narrow the scope of the investigation.

4.7. T1018 – Remote System Discovery

Throughout an attack, adversaries may move laterally to other hosts in the environment. To support this movement, they may attempt to find other hosts on the

Allen Cox, alcox.infosec@gmail.com
network during the discovery phase. While attackers have access to a wide range of tools used to identify remote systems, cloud platforms offer APIs that can provide this information about the environment directly (The MITRE Corporation, 2020).

Moberly’s GCP Red Team exercise identifies two gcloud commands attackers can run to understand the environment. First, the attacker can request a listing of subnets contained in a project:

\$ gcloud compute networks subnets list

Second, gcloud will conveniently list all the compute instances in the project:

\$ gcloud compute instances list

Using these commands is significantly less noisy than using a port scanning tool to identify nearby hosts (Moberly, 2020). The next section will discuss techniques for finding evidence of this activity.

4.8. T1046 – Network Service Scanning

Once adversaries have discovered remote systems, they may attempt to discover services running on other nearby hosts. Often, attackers use a port scanner like nmap. However, when interacting with cloud-hosted systems, adversaries may have access to APIs or other tools that can more quietly assist in remote discovery (The MITRE Corporation, 2020).

Moberly has developed gcp_firewall_enum, a Python script that analyzes GCP firewall rules against a list of compute instances. The tool’s output is a file that shows the “external IP address, allowed TCP ports, [and] allowed UDP ports” for each compute instance name. The Python script runs against offline data, and the attacker needs to collect the appropriate inputs using the cloud platform API (2020). The test will run two gcloud commands to list firewall rules and instances, simulating an attacker utilizing this tool. The README file for the gcp_firewall_enum tool provides these commands.

Conveniently, the gcloud application logs all commands run with the tool and saves the results the tool returned. The application stores the logs in 
~/.config/gcloud/logs/, separating them into folders by date and then files by

Allen Cox, alcox.infosec@gmail.com
time. Unfortunately, the administrative logs collected in the GCP console, external to the instance, do not record these API calls by default. Figure 14 contains the Bash commands utilized in the experimental live response script.

```
# T1018 - Remote System Discovery / T1046 - Network Service Scanning
mkdir /tmp/liveresponse/T1046

## Collect the gcloud logs

cd /home

for FOLDER in *; do
    if [-d "$FOLDER/.config/gcloud/logs"]; then
        mkdir /tmp/liveresponse/T1046/gcloud_logs_$FOLDER
        cp -r "$FOLDER/.config/gcloud/logs/*" /tmp/liveresponse/T1046/gcloud_logs_$FOLDER/
    fi
done

if [-d "/root/.config/gcloud/logs"]; then
    mkdir /tmp/liveresponse/T1046/gcloud_logs_root
    cp -r "/root/.config/gcloud/logs/*" /tmp/liveresponse/T1046/gcloud_logs_root/
fi
```

Figure 14 – Collect Discovery Evidence

While responders could find gcloud commands in the .bash_history file, identifying the attacker activity in the gcloud logs provided greater detail. The gcloud log provided a timestamp for each command the attacker ran and listed the results the API returned. Figure 15 is an example log of the attacker running `gcloud compute instances list`.

```
2020-05-22 05:03:37,430 DEBUG root Running [gcloud.compute.instances.list] with arguments: {}
2020-05-22 05:03:37,952 INFO root Display format: "
name,
zone, hostname(),
machineType, machine_type, hostname(),
scheduling, preemptible, yes
(no=""),
networkInterface, networkIP, notnull(), list(), label="EXTERNAL_IP",
project, accessConfig[0], natIP, notnull(), list() : label="EXTERNAL_IP",
status,
"
2020-05-22 05:03:38,496 INFO root cache collection=compute.instances api_version=v1 params=["project",
"zone", "instance"]
```

Figure 15 – gcloud CLI Logs

Allen Cox, alcox.infosec@gmail.com
While the gcloud log is especially helpful, like the .bash_history file, it does not have any protections, and the attacker could delete or modify the log.

### 4.9. T1530 – Data from Cloud Storage Object

Like other cloud providers, GCP has a data storage solution – sensibly named “Google Cloud Storage.” These storage solutions can often contain a treasure-trove of data to interest adversaries, such as code for applications, web application credentials, and even private customer data. When storage access controls are too permissive, adversaries may be able to access the data without compromising other project resources (The MITRE Corporation, 2020). Once an attacker has gained access to a compute instance, gathering this data is often even easier.

Moberly’s GCP Red Team exercise recommends using the gsutil command to enumerate a project’s storage buckets. Once an attacker has identified the project’s buckets, they can list each bucket’s contents, read individual files, and copy the file to disk (2020). Figure 16 demonstrates the steps an attacker may take to identify and read a file from a bucket and the respective outputs from gsutil.

```bash
[definitelynotahacker@compromised-instance ~]$ gsutil ls gs://sti_research_bucket/
[definitelynotahacker@compromised-instance ~]$ gsutil ls -r gs://sti_research_bucket/
[definitelynotahacker@compromised-instance ~]$ gsutil cat gs://sti_research_bucket/steal_me.txt
This is a BIG secret.[definitelynotahacker@compromised-instance ~]$ 
```

**Figure 16 – Reading data from a Google Storage Bucket**

The gsutil tool does not require any specific access scopes since it does not interface with the cloud platform API. Unlike gcloud, using the tool successfully is possible from an instance with default access scopes. When a user runs gsutil for the first time, the tool creates a folder named .gsutil in the home directory. Unfortunately, gsutil only stores credentials in this folder and does not log how the user employed the tool. Even worse, GCP does not provide access logging on storage buckets by default. While configuring access logging on Google Storage buckets is relatively straightforward, if the owner has not configured it, the responder will be unable to identify which buckets and files an adversary accessed (Google Cloud, 2020-i). Figure 17 contains the Bash commands utilized in the experimental live response script.

Allen Cox, alcox.infosec@gmail.com
While collecting this data identified the attacker’s use of gsutil in the controlled laboratory environment, checking for the .gsutil folder will likely be a less effective detection in real development environments. In the best-case scenario, the attacker may create a new user account and run gsutil from that account. All this would confirm for the responder is that the attacker attempted to interact with a storage bucket. Gathering higher fidelity information would require enabling access logging on the storage bucket. As discussed, GCP does not collect this logging by default. However, if the responder identifies the name of an impacted bucket, they can run the command “gsutil logging get gs://<bucket-name>” to determine whether logging is configured and where the logs are stored.

5. Results

The experiment involved running the live response script against the “clean-instance” host to generate baseline data for default GCP CentOS 7 images. The complete live response script is available on GitHub: https://github.com/coxallen/gcp-centos-liveresponse. Next, the attacker tests ran against the “compromised-instance” host. Appendix A contains a full list of the attacker commands used in this experiment. Finally, the live response script ran against the “compromised-instance” host.

5.1. The Live Response Script

The live response script consisted of the commands presented in Section 4. Note that the live response script created for this research prioritized clarity over conciseness.

Allen Cox, alcox.infosec@gmail.com
and only collected the data relevant to detecting the nine ATT&CK techniques discussed in the research. A complete live response script should also collect other critical investigative data. Responders should consider including network connections, running processes, user activity logs, and application-specific logging (Luttgens, Pepe, & Mandia, 2014). Responders should also plan the mechanisms they will use to automate the script’s deployment and collection of the results once the script has run.

### 5.2. Experimental Results

Overall, the live response script successfully collected evidence that enabled responders to detect the attacker technique, except for T1530 – Data from Cloud Storage Object. Techniques heavily reliant on logging of command-line activity, like T1059 and T1064, were only partially successful. Here, partial success indicates that evidence exists that would be of use to a responder, but the evidence may be missing vital contextual information. GCP logged shockingly little of the activity that occurred on the instance, resulted in API calls, or involved storage bucket access. The sole exception was gcloud’s verbose logging in the home directory. Although, as previously mentioned, an attacker could manipulate these logs, and GCP does not record gcloud activity elsewhere. Table 1 summarizes the experimental results for each technique.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Linux-based Evidence (Default Settings)</th>
<th>GCP Logging and Tools (Default Settings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1059 – Command-Line Interface</td>
<td>Partially</td>
<td>No</td>
</tr>
<tr>
<td>T1168 – Local Job Scheduling</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T1064 – Scripting</td>
<td>Partially</td>
<td>No</td>
</tr>
<tr>
<td>T1156 - .bash_profile and .bashrc</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T1136 – Create Account</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T1501 – Systemd Service</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T1018 – Remote System Discovery</td>
<td>Partially</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### 5.3. Recommendations for Further Research

As discussed in the introduction to this research, the body of information security literature exploring methodologies to mitigate, detect, and respond to malicious activity in GCP is limited, and there are numerous topics from across multiple disciplines to explore. Recommendations for future research include:

- Determining the optimal configuration of GCP logging settings available in the cloud console. The research should also consider the balance between providing responders with enough information to complete an investigation while not placing an unrealistic data retention burden on organizations.
- Exploring other ATT&CK Cloud tactics not explored in this research. Consider attacks against identity and role management through IAM and how responders can detect abuse of these resources.
- Recommending tools and techniques to conduct live response on GCP instances, with a focus on identifying tools that enable detection of cloud resource abuse.

These recommendations specifically focus on detection and response capabilities in GCP. Future research should contribute to an eventual body of literature that will assist defenders in responding to GCP incidents, developing detections, and hardening the platform.

### 6. Conclusion

This research successfully implemented a live response script to help responders investigate eight of the nine discussed MITRE ATT&CK techniques. Using a live response script can help defenders quickly scope an incident and make initial discoveries into what the attacker accomplished. In situations where defenders are responding to a

Allen Cox, alcox.infosec@gmail.com
compromise in an environment with default settings, they should rely heavily on the data available on the Compute Engine instance that was compromised. By default, GCP does not log activity that occurs on the instance, or against any storage buckets, in the cloud console.

Defenders with GCP in their environment should encourage developers to create IAM roles that follow the principle of least privilege. The primitive IAM roles that GCP uses by default grant broad permissions that an attacker could easily abuse to gain significant access to an organization’s resources in GCP and possibly any resources logically connected to the platform.
References


Allen Cox, alcox.infosec@gmail.com


Allen Cox, alcox.infosec@gmail.com
Appendix A
List of Attacker Testing Commands

# T1136 - Create Account (via gcloud)
NEWUSER="definitelynotahacker"
ssh-keygen -t rsa -C "$NEWUSER" -f ./key -P ""
NEWKEY="$(cat ./key.pub)"
echo "$NEWUSER:$NEWKEY" > ./meta.txt
gcloud compute instances add-metadata compromised-instance --metadata-from-file ssh-keys=meta.txt
ssh -i ./key "$NEWUSER"@localhost

# T1059 - Command-Line Interface
curl -sS https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1059/echo-art-fish.sh | bash
cat /tmp/art-fish.txt

# T1168 - Local Job Scheduling
sudo su -
# Executed as root
echo "echo 'Hello from Atomic Red Team' > /tmp/atomic.log" > /etc/cron.daily/persistevil
chmod +x /etc/cron.daily/persistevil
ls -al /etc/cron.daily/
exit
# Return to definitelynotahacker context

# T1064 - Scripting
sh -c "echo 'echo Hello from the Atomic Red Team' > /tmp/art.sh"
sh -c "echo 'ping -c 4 8.8.8.8' >> /tmp/art.sh"
chmod +x /tmp/art.sh
sh /tmp/art.sh

# T1156 - .bash_profile and .bashrc
vim persist.py
echo "python /home/definitelynotahacker/persist.py" >> .bash_profile

# T1136 - Create Account (via useradd)
sudo useradd -M -N -r -s /bin/bash -c evil_account evil_user

# T1501 - Systemd Service
# Executed as root
sudo su -
echo "[Unit]" > /etc/systemd/system/art-systemd-service.service
echo "Description=Atomic Red Team Systemd Service" >> /etc/systemd/system/art-systemd-service.service
echo "" >> /etc/systemd/system/art-systemd-service.service
echo "[Service]" >> /etc/systemd/system/art-systemd-service.service
echo "Type=simple" >> /etc/systemd/system/art-systemd-service.service
echo "ExecStart=/bin/touch /tmp/art-systemd-execstart-marker" >> /etc/systemd/system/art-systemd-service.service
echo "ExecStartPre=/bin/touch /tmp/art-systemd-execstartpre-marker" >> /etc/systemd/system/art-systemd-service.service
echo "ExecStartPost=/bin/touch /tmp/art-systemd-execstartpost-marker" >> /etc/systemd/system/art-systemd-service.service
echo "ExecReload=/bin/touch /tmp/art-systemd-execreload-marker" >> /etc/systemd/system/art-systemd-service.service
echo "ExecStop=/bin/touch /tmp/art-systemd-execstop-marker" >> /etc/systemd/system/art-systemd-service.service
echo "ExecStopPost=/bin/touch /tmp/art-systemd-execstoppost-marker" >> /etc/systemd/system/art-systemd-service.service
echo "" >> /etc/systemd/system/art-systemd-service.service
echo "[Install]" >> /etc/systemd/system/art-systemd-service.service
systemctl daemon-reload
systemctl enable art-systemd-service.service
systemctl start art-systemd-service.service
ls /tmp
exit

# Return to definitelynotahacker context

# T1018 - Remote System Discovery
gcloud compute networks subnets list
gcloud compute instances list

# T1046 - Network Service Scanning
gcloud compute firewall-rules list --format="json(name,allowed[].map().firewall_rule().list(),network,targetServiceAccounts.list(),targetTags.list())" --filter="direction:INGRESS AND disabled:False AND sourceRanges.list():0.0.0.0/0 AND allowed[].map().firewall_rule().list():*" | tee ./firewall-rules.json
gcloud compute instances list --format="json(id,name,selfLink,networkInterfaces[].accessConfigs[0].natIP,serviceAccount.name,serviceAccount.email,serviceAccount.tags.items[],networkInterfaces[].network)" --filter="networkInterfaces[].accessConfigs[0].type:ONE_TO_ONE"
ATT&CK-Based Live Response for GCP CentOS Instances

Allen Cox, alcox.infosec@gmail.com

```bash
_NAT AND status:running" | tee ./compute-instances.json

# T1530 - Data from Cloud Storage Object
gsutil ls
gsutil ls -r gs://sti_research_bucket/
gsutil cat gs://sti_research_bucket/steal_me.txt

# End the session
exit
```
## Upcoming SANS Forensics Training

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Virginia Beach 2020 - Live Online</td>
<td>Virginia Beach, VA</td>
<td>Aug 30, 2020 - Sep 04, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Virginia Beach 2020</td>
<td>Virginia Beach, VA</td>
<td>Aug 30, 2020 - Sep 04, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Japan Bilingual Live Online</td>
<td>, Japan</td>
<td>Aug 31, 2020 - Sep 05, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London September 2020 - Live Online</td>
<td>London, United Kingdom</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Threat Hunting &amp; Incident Response Summit &amp; Training 2020</td>
<td>Virtual - US Eastern,</td>
<td>Sep 10, 2020 - Sep 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Munich September 2020 - Live Online</td>
<td>Munich, Germany</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Network Security 2020</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 25, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Australia Spring 2020</td>
<td>, Australia</td>
<td>Sep 21, 2020 - Oct 03, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Australia Spring 2020 - Live Online</td>
<td>, Australia</td>
<td>Sep 21, 2020 - Oct 03, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS San Antonio Fall 2020</td>
<td>San Antonio, TX</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Antonio Fall 2020 - Live Online</td>
<td>San Antonio, TX</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Northern VA - Reston Fall 2020</td>
<td>Reston, VA</td>
<td>Sep 28, 2020 - Oct 03, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>OIl &amp; Gas Cybersecurity Summit &amp; Training 2020</td>
<td>Virtual - US Central,</td>
<td>Oct 02, 2020 - Oct 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS FOR500 Milan 2020 (In Italian) - Live Online</td>
<td>Milan, Italy</td>
<td>Oct 05, 2020 - Oct 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Amsterdam October 2020</td>
<td>Amsterdam, Netherlands</td>
<td>Oct 05, 2020 - Oct 10, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS FOR500 Milan 2020 (In Italian)</td>
<td>Milan, Italy</td>
<td>Oct 05, 2020 - Oct 10, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Tokyo Autumn 2020</td>
<td>Tokyo, Japan</td>
<td>Oct 05, 2020 - Oct 17, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Amsterdam October 2020 - Live Online</td>
<td>Amsterdam, Netherlands</td>
<td>Oct 05, 2020 - Oct 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Orlando 2020</td>
<td>Orlando, FL</td>
<td>Oct 12, 2020 - Oct 17, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS FOR572 Australia Online 2020</td>
<td>, Australia</td>
<td>Oct 12, 2020 - Oct 23, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London October 2020 - Live Online</td>
<td>London, United Kingdom</td>
<td>Oct 12, 2020 - Oct 17, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS October Singapore 2020</td>
<td>Singapore, Singapore</td>
<td>Oct 12, 2020 - Oct 24, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Doha October 2020</td>
<td>Doha, Qatar</td>
<td>Oct 17, 2020 - Oct 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Riyadh October 2020 - Live Online</td>
<td>Riyadh, Kingdom Of Saudi Arabia</td>
<td>Oct 17, 2020 - Oct 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Doha October 2020 - Live Online</td>
<td>Doha, Qatar</td>
<td>Oct 17, 2020 - Oct 22, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS FOR500 (In Spanish) Online 2020</td>
<td>, United Arab Emirates</td>
<td>Oct 19, 2020 - Oct 30, 2020</td>
<td>vLive</td>
</tr>
</tbody>
</table>