

# Detection Engineering Demystified: Building Custom Detections for GitHub Enterprise

---

**David French**

Staff Adoption Engineer, Google Cloud

[@threatpunter](#)

# David French > About Me

- 18+ years in IT and cybersecurity
  - Blue Team life: Detection Engineer, Threat Hunter, SOC Analyst
  - Vendor life: Threat Research, Detection Engineering, building SIEMs & EDRs
- Currently at Google Cloud (Google Security Operations)
- Formerly Twilio, Elastic, Endgame, Capital Group
- Enjoys sharing knowledge & research:
  - Speaker at Black Hat, BSides, FIRST
  - [Blog](#), [community contributions](#), [Detection-as-Code](#), [Dorothy](#)
- When I'm not working, I'm hiking, fishing, cycling, etc



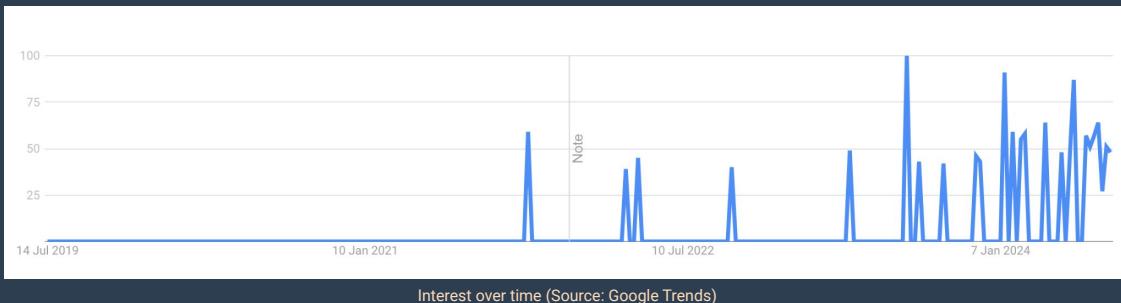
# Agenda

1. A Brief Introduction to Detection Engineering
2. Case Study: Intelligence-Driven Detection Engineering
3. Developing a New Detection
4. Monitoring Your Data Pipeline
5. Testing Detections
6. Wrap Up
  - a. Key Takeaways
  - b. Useful Resources
  - c. Q&A

# A Brief Introduction to Detection Engineering

# What is Detection Engineering?

- Specialization focused on implementing detective security controls that identify malicious or unauthorized activity
- Aims to detect potential incidents before they can cause significant damage
- Continuous process for developing, testing, and refining detections
- Complements preventative controls
- Emphasis on detecting behavior vs. indicators of Compromise (IOCs)



A screenshot of a LinkedIn search interface. The search bar contains "Detection Engineer" and "United States". Below the search bar are filters for "Jobs", "Past month", "\$160,000+", and "Experience level". The main results section is titled "Detection Engineer in United States" and shows "303 results". There is a "Set alert" button with a toggle switch. The LinkedIn logo is visible at the top left.

# Benefits of Detection Engineering (1)

- Reduced risk
  - Proactive detection identifies attacks early on
  - Gives defenders a chance to respond before significant damage occurs
  - Can save \$ and (sometimes) lives...
- Faster response times
  - Actionable detections provide the info needed to understand and respond to threats quickly
  - Helps maintain company's reputation & trust with customers

## From Risky Biz News:

### **Ransomware attacks increase hospital mortality rates:**

A whitepaper published last year by academics from the University of Minnesota's medical school has looked at the aftermath of ransomware attacks on US hospitals and found evidence to suggest that mortality rates typically increase by around 20%.

[Hacked to Pieces? The Effects of Ransomware Attacks on Hospitals and Patients](#)

**USD 4.45  
million**

The global average cost of a data breach in 2023 was USD 4.45 million, a 15% increase over 3 years.

[IBM Cost of a Data Breach Report 2023](#)

# Benefits of Detection Engineering (2)

- Increased visibility
  - Engineers proactively identify and integrate relevant data sources
  - Detection coverage is assessed against new threats/tactics
- Can help satisfy compliance/regulatory requirements
  - e.g. Detections related to data loss prevention or SWIFT compliance



## MY DETECTION RULES

SESSION COOKIE THEFT: ← ← → → ← + ⓐ

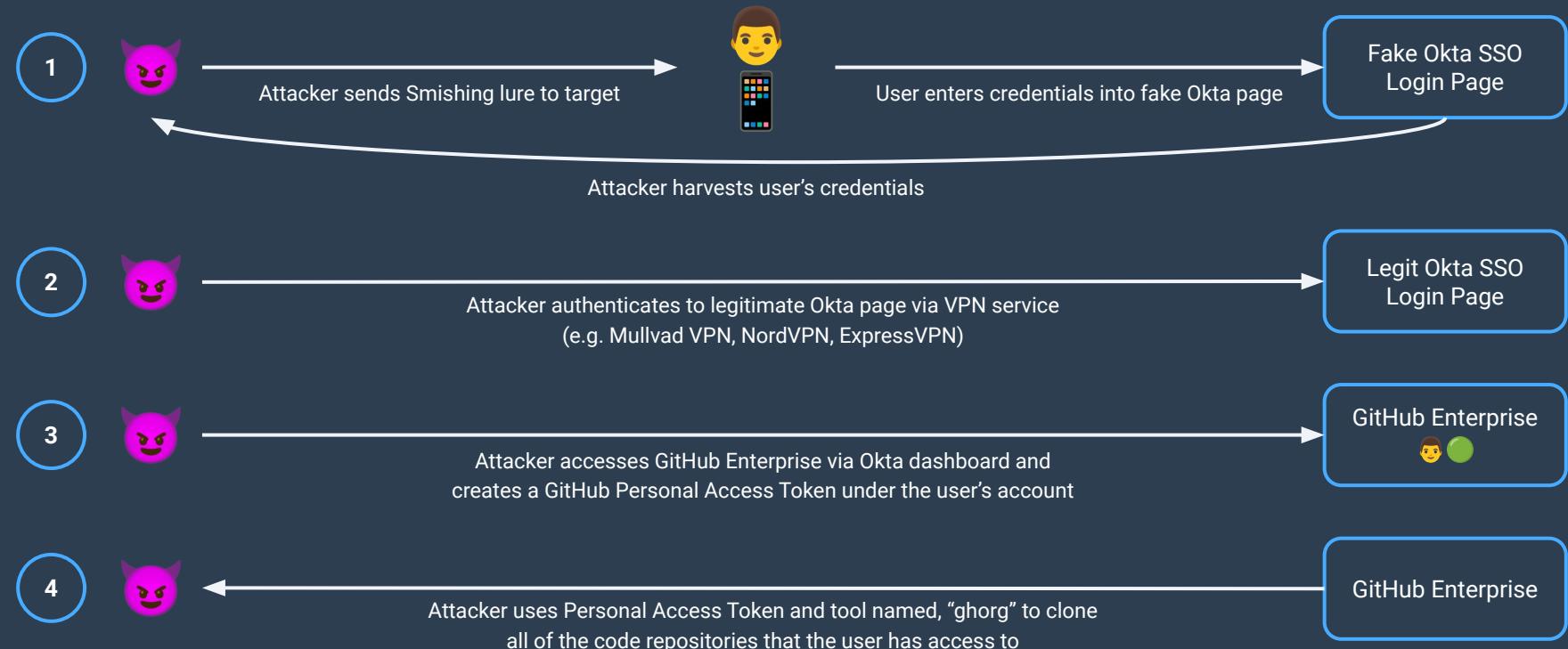
DATA COLLECTED FROM  
CODE REPOSITORIES : → → ↓ ↑ + ⓑ

TWO-FACTOR AUTHENTICATION 1 ↑ ← → + ⓐ  
DISABLED :

# Case Study: Intelligence-Driven Detection Engineering

# Case Study: Intelligence-Driven Detection Engineering

Intelligence received from fellow practitioner working in the same industry



# Why GitHub is Targeted by Attackers

- Code repositories may contain intellectual property
- Source code can be examined for vulnerabilities that can be exploited in subsequent attacks
- Harvested secrets can be used in follow up attacks
- Software supply chain attacks

**Report: Microsoft's GitHub Account Gets Hacked**



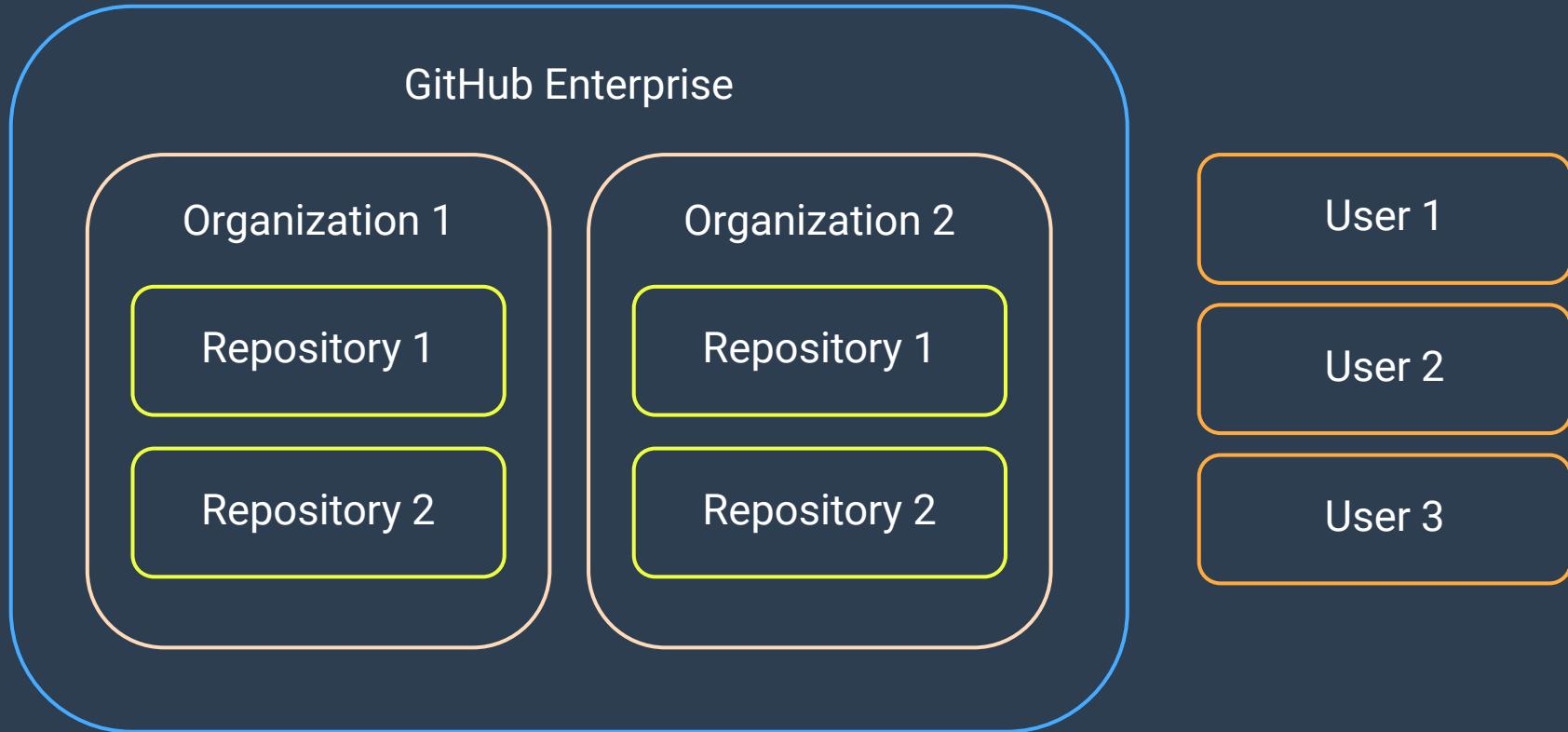
Okta's source code stolen after GitHub repositories hacked

By Ax Sharma

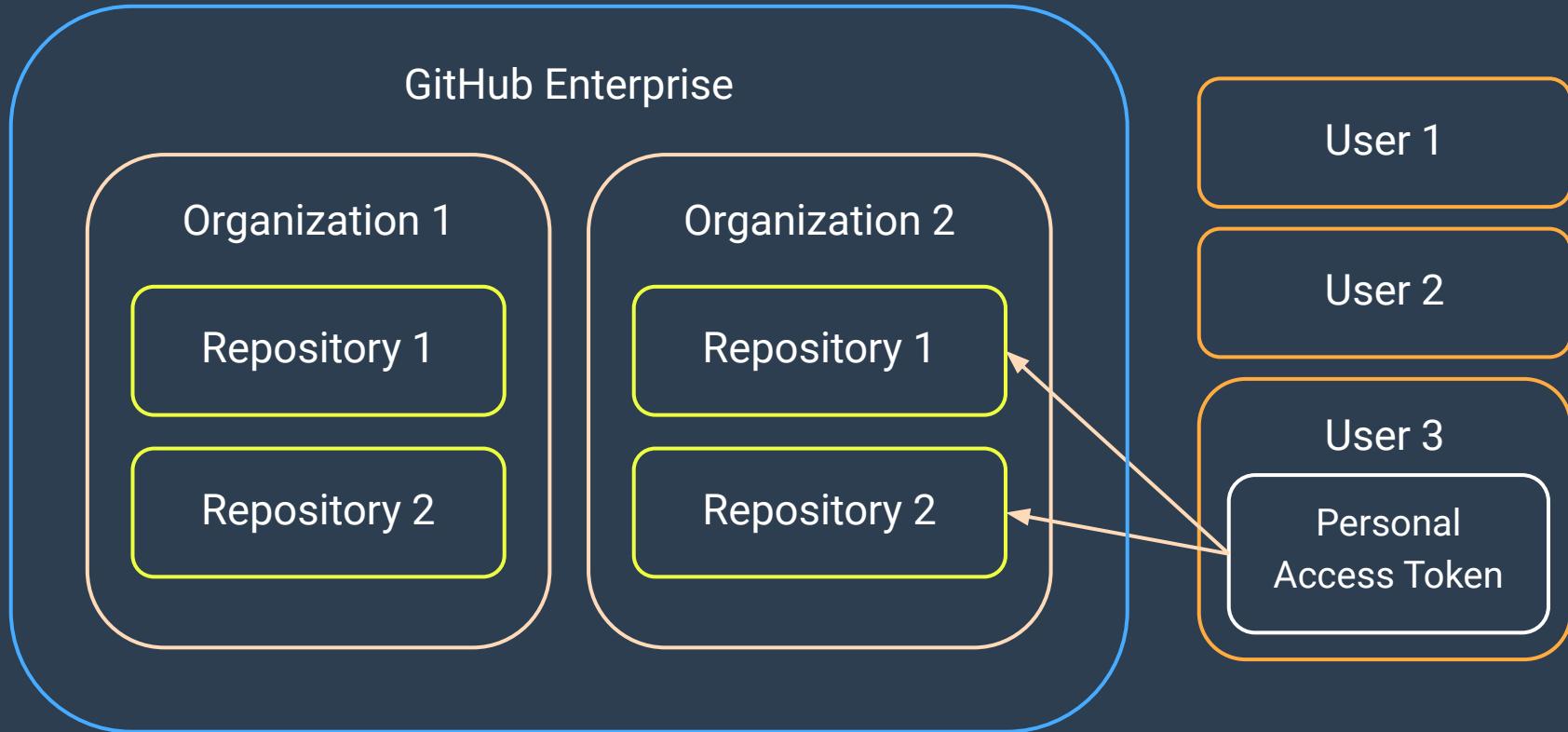
December 21, 2022 01:15 AM 1



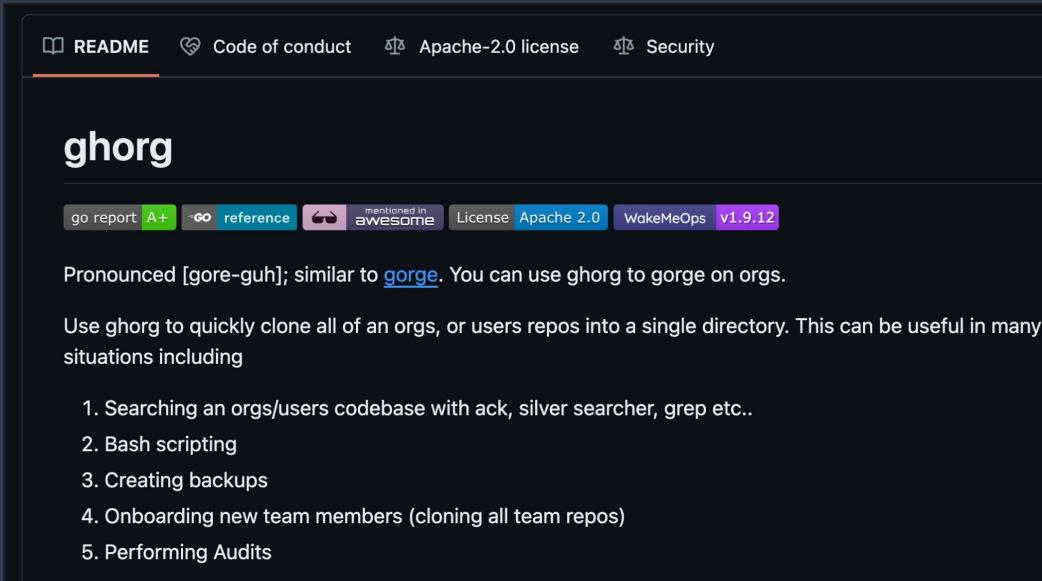
# GitHub: Key Terms & Concepts



# GitHub: Key Terms & Concepts



# Reviewing the Attacker's Tool of Choice



The screenshot shows the README page of the `ghorg` GitHub repository. The page includes a navigation bar with links to `README`, `Code of conduct`, `Apache-2.0 license`, and `Security`. Below the navigation bar, the title `ghorg` is displayed. A row of buttons includes `go report`, `A+`, `reference`, `mentioned in awesome`, `License`, `Apache 2.0`, `WakeMeOps`, and `v1.9.12`. The main content area contains the following text:

Pronounced [gore-guh]; similar to [gorge](#). You can use `ghorg` to gorge on orgs.

Use `ghorg` to quickly clone all of an orgs, or users repos into a single directory. This can be useful in many situations including

1. Searching an orgs/users codebase with ack, silver searcher, grep etc..
2. Bash scripting
3. Creating backups
4. Onboarding new team members (cloning all team repos)
5. Performing Audits

## 3. Clone a `users` repos

```
ghorg clone <github_username> --clone-type=user --base-url=https://<your-hosted-github>.com --token=XXXXXX
```



# Indicator vs. Behavior-Based Detection

## Indicator-Based Detection

Analyze tool and write signatures to alert on specific indicators (e.g. user agent)

Tools are often easily modified to evade this type of detection

What if the attacker uses a different tool?

## Behavior-Based Detection

Focus on detecting underlying actions that are required to achieve the desired outcome

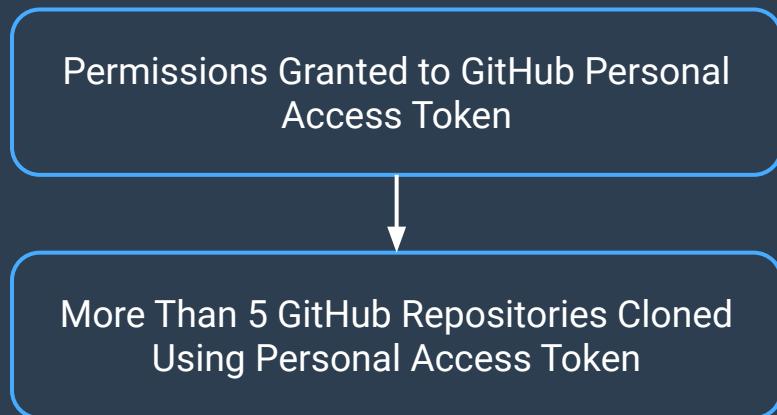
These detections often have a longer shelf life than indicator-based detections

Incorporates context and sequences/patterns of behavior

# Developing a New Detection

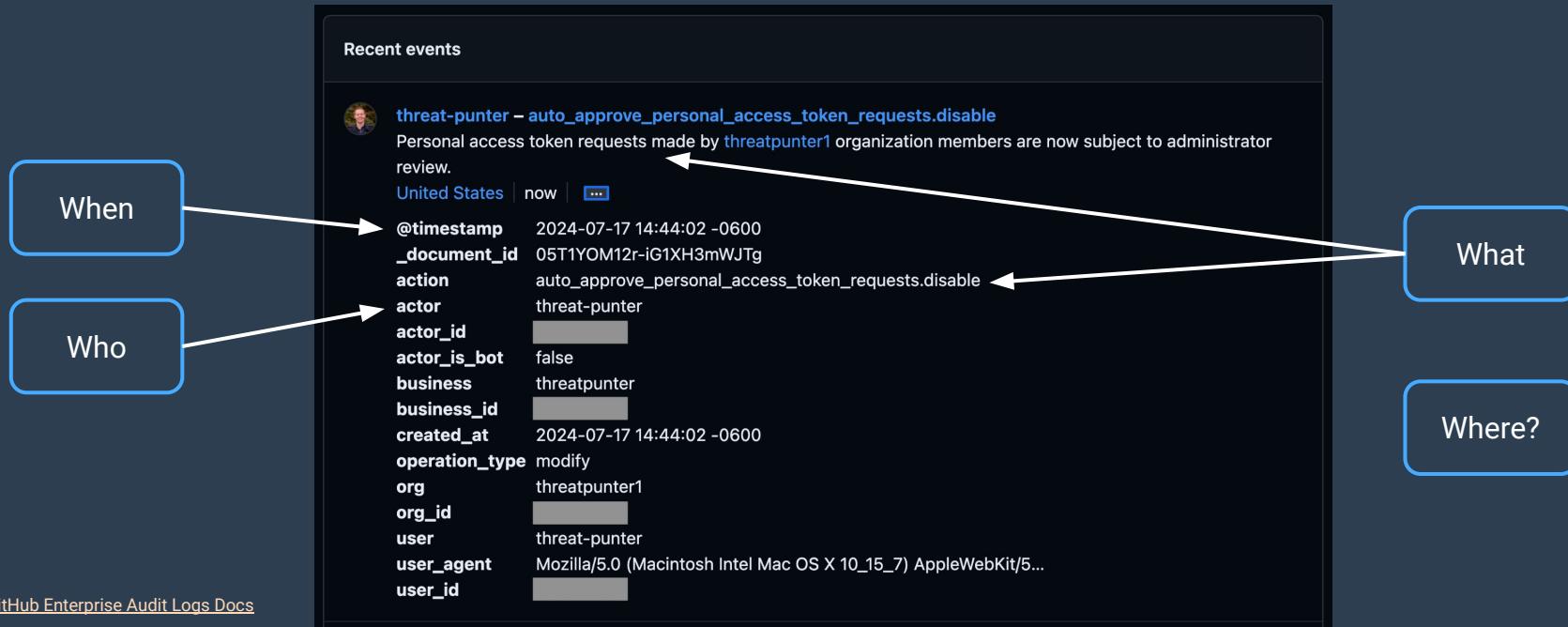
# Designing a New Detection

- Let's develop a detection that alerts us when a GitHub user performs the following sequence of behaviors
- We'll build a simple detection and refine it from there



# Understanding the GitHub Enterprise Audit Log

“The audit log lists events triggered by activities that affect your enterprise within the last 180 days. The audit log retains Git events for seven days.”



# Configuring the GitHub Audit Log

This data source must be configured before it can be used for many detection use cases

The screenshot shows the 'Audit log' settings page. At the top, there are three tabs: 'Events', 'Log streaming', and 'Settings'. The 'Settings' tab is selected. Below the tabs, the first section is titled 'Disclose actor IP addresses in audit logs' and contains a checkbox for 'Enable source IP disclosure'. A note explains that enabling this feature allows viewing IP addresses of current members for enterprise and organization audit log events, and advises reviewing it with legal teams. A 'Save' button is present. The second section is titled 'API Requests' and contains a checkbox for 'Enable API Request Events'. A note explains that enabling this feature streams API request events via audit log streaming with a 24-hour retention period. Another 'Save' button is present.

These options are not enabled by default

Audit log

Events Log streaming Settings

Disclose actor IP addresses in audit logs

Enable source IP disclosure  
Enabling will allow you to view IP addresses of current members for enterprise and organization audit log events. As this feature makes your users' IP addresses automatically available, you should review this change with your legal team to determine whether any user notification is required.

Save

API Requests

Enable API Request Events  
Enable API request events to be streamed via audit log streaming. These events have a retention period of 24 hours.

Save

# Configuring GitHub Audit Log Streaming

The audit log can be streamed to an external system for storage and further analysis

1

## Configure audit log streaming

Set up streaming of audit data from GitHub to an external database management system like Splunk or Azure Event Hubs.

Configure stream ▾

Amazon S3

Azure Blob Storage

Azure Event Hubs

Datadog

Google Cloud Storage

Splunk

2

## Configure Audit log streaming to Google Cloud Storage

Stream audit logs to a Google Cloud Storage endpoint. [Learn more.](#)

Bucket

github-audit-logs-1234

JSON Credentials

\*\*\*\*\*

Check endpoint



Endpoint check successful

3

## Configured stream

Streaming to Google Cloud Storage  
github-audit-logs-1234

# Ingesting GitHub Audit Logs into a SIEM

Enables log analysis, detection use cases, correlation with other security-relevant events, long term retention, etc

The screenshot shows the Google SecOps SIEM Settings - Feeds interface. On the left, a sidebar menu lists various settings: Profile, Users & Groups, Roles, Feeds (which is selected and highlighted in blue), and Forwarders. The main panel is titled 'FEEDS' and contains a table with the following data:

FEED NAME	FEED ID ↑	STATUS	SOURCE TYPE	LOG TYPE	LAST SUCCEEDED ON
Okta System Logs	a37ab324-0f75-41c7-b36d-cbfa45cc29ab	InProgress	Third party API	Okta	2024-07-17 21:12:14
GitHub Enterprise Audi...	t7rnpnbnxsjng4	Completed	Google Cloud Storage	GitHub	2024-07-17 21:11:14

The row for 'GitHub Enterprise Audi...' is highlighted with a red box.

# Simulating the Behavior to Detect

1. Create a new GitHub Personal Access Token (PAT)
2. Grant the token access to at least six GitHub repositories
3. Clone the six GitHub repositories using the PAT
4. Explore the events that were generated

EVENTS	PIVOT	Search events...					
TIMESTAMP	EVENT	PRINCIPAL.USER.USERID	METADATA.PRODUCT_EVENT_TYPE	PRINCIPAL.IP	EXTRACTED.FIELDS["REPO"]	EXTRACTED.FIELDS["PROGRAMMATIC_ACCESS_TYPE"]	
2024-07-17T04:34:23.925	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo6	threat-punter	git.clone		threatpunter1/repo6	Fine-grained personal access token	
2024-07-17T04:34:17.486	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo5	threat-punter	git.clone		threatpunter1/repo5	Fine-grained personal access token	
2024-07-17T04:34:11.092	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo4	threat-punter	git.clone		threatpunter1/repo4	Fine-grained personal access token	
2024-07-17T04:34:00.361	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo3	threat-punter	git.clone		threatpunter1/repo3	Fine-grained personal access token	
2024-07-17T04:33:51.889	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo2	threat-punter	git.clone		threatpunter1/repo2	Fine-grained personal access token	
2024-07-17T04:33:39.099	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - threatpunter1/repo1	threat-punter	git.clone		threatpunter1/repo1	Fine-grained personal access token	
2024-07-17T04:33:22.947	[USER_RESOURCE_UPDATE_CONTENT] threat-punter - unknown resource	threat-punter	personal_access_token.access_granted		[Unknown]	[Unknown]	

# Developing the Detection Logic (1)

```
18 events:
19   // GitHub Personal Access Token (PAT) access granted event
20   $github_pat.metadata.product_name = "GITHUB"
21   $github_pat.metadata.product_event_type = "personal_access_token.access_granted"
22
23   // GitHub repository clone event
24   $github_clone.metadata.product_name = "GITHUB"
25   $github_clone.metadata.product_event_type = "git.clone"
26   $github_clone.additional.fields["repository_public"] = "false"
27   $github_clone.additional.fields["programmatic_access_type"] = "Fine-grained personal access token"
28   $github_clone.target.resource.name = $github_repo_name
29
30   // Join GitHub PAT access granted event to GitHub repository clone event
31   $github_pat.principal.user.userid = $github_clone.principal.user.userid
32
33   // Placeholder for match section
34   $github_pat.principal.user.userid = $user_id
35
36   // Ensure PAT access granted event occurred before repository clone events
37   $github_pat.metadata.event_timestamp.seconds < $github_clone.metadata.event_timestamp.seconds
```

# Developing the Detection Logic (2)

```
40     match:
41         $user_id over 30m
42
43     outcome:
44         $github_repo_name_distinct_count = count_distinct($github_repo_name)
45
46     condition:
47         // Customize GitHub repo count to fit your environment
48         $github_pat and $github_clone and $github_repo_name_distinct_count > 5
49     }
```

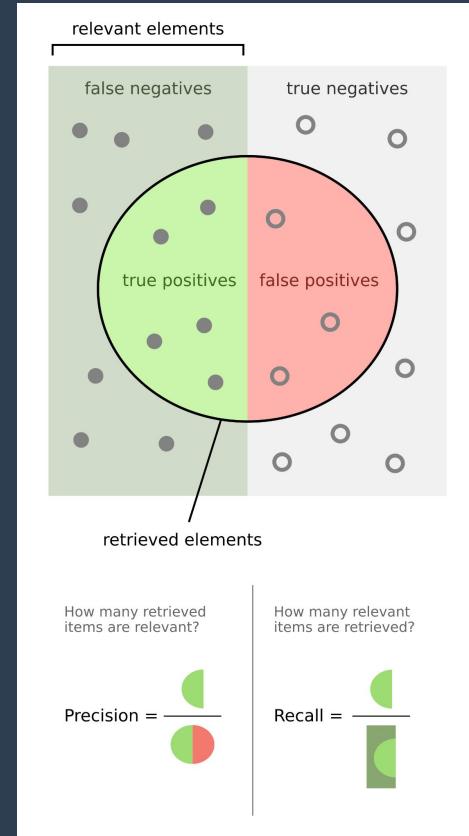
# Validating the Detection

- Simulate the behavior again and validate that the detection generates an alert
- Or run the detection logic over historical events and validate that it matches the intended behavior

TIMESTAMP ↓	DETECTION	GITHUB_REPO_NAME_DISTINCT_COUNT...	USER_ID (DETECTION)	METADATA.PRODUCT_EVENT_TYPE
2024-07-17T21:30:00.000	<b>DETECTION</b> user_id:threat-punter	6	threat-punter	personal_access_token.access_granted,...
EVENTS (7)				
2024-07-17T21:03:23.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - unknown resource	[Unknown]	[Unknown]	personal_access_token.access_granted
2024-07-17T21:03:28.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo1	[Unknown]	[Unknown]	git.clone
2024-07-17T21:03:33.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo2	[Unknown]	[Unknown]	git.clone
2024-07-17T21:03:38.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo3	[Unknown]	[Unknown]	git.clone
2024-07-17T21:03:43.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo4	[Unknown]	[Unknown]	git.clone
2024-07-17T21:03:48.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo5	[Unknown]	[Unknown]	git.clone
2024-07-17T21:03:53.000	<a href="#">USER_RESOURCE_UPDATE_CONTENT</a> [G] threat-punter - threatpunter1/repo6	[Unknown]	[Unknown]	git.clone

# False Positive Report from the SOC

- When a software engineer gets a new laptop, it's common for them to:
  - Create a new GitHub Personal Access Token
  - Clone the private code repos that they work on
- This behavior causes pesky false positives
- Let's look for a way to tune the detection to increase its precision



# Detection Engineering and Third-Party Data Feeds

- Modify the detection logic to alert when the behavior originates from an IP address that's attributed to a VPN service
  - Might introduce FPs if engineers are using VPN services
  - Users should be using the company-provided VPN
  - Create another detection to alert on non-approved VPN use
- Spur is a service that provides data on VPNs, residential proxies, and bots
  - Feeds can be ingested into SIEM for correlation or enrichment



31.171.153.66 -  
**Mullvad VPN**

31.171.153.66 belongs to the **Mullvad VPN** anonymization network. Mullvad VPN users route traffic through 31.171.153.66 to obscure their traffic from ISPs and mask their identity from servers on the internet.

---



103.62.49.195 -  
**Nord VPN**

103.62.49.195 belongs to the **Nord VPN** anonymization network. Nord VPN users route traffic through 103.62.49.195 to obscure their traffic from ISPs and mask their identity from servers on the internet.

# Modifying the Detection Logic

Modify the detection logic to match on events that originated from an IP address that Spur attributes to a VPN service

```
18    events:  
19      // GitHub Personal Access Token (PAT) access granted event  
20      $github_pat.metadata.vendor_name = "GITHUB"  
21      $github_pat.metadata.product_name = "GITHUB"  
22      $github_pat.metadata.product_event_type = "personal_access_token.access_granted"  
23  
24      // GitHub repository clone event  
25      $github_pat.metadata.vendor_name = "GITHUB"  
26      $github_clone.metadata.product_name = "GITHUB"  
27      $github_clone.metadata.product_event_type = "git.clone"  
28      $github_clone.additional.fields["repository_public"] = "false"  
29      $github_clone.target.resource.name = $github_repo_name  
30      $github_clone.principal.ip = $ip  
31  
32      // IP address is in Spur's VPN IP address feed  
33      $spur.graph.metadata.vendor_name = "Spur"  
34      $spur.graph.metadata.product_name = "Spur Feeds"  
35      $spur.graph.metadata.entity_type = "IP_ADDRESS"  
36      $spur.graph.entity.artifact.tags = "VPN"  
37      // Join GitHub events with Spur data  
38      $spur.graph.entity.artifact.ip = $ip
```

# Validating the Detection

Simulate the attacker behavior again and validate that the detection alerts on the intended behavior

TIMESTAMP ↓	DETECTION	METADATA.PRODUCT_EVENT_TYPE
2024-07-17T21:06:00.000	<b>DETECTION ALERT</b> user_id:threat-punter	personal_access_token.access_gran...
<b>ENTITIES (1)</b>		
2024-07-17T00:00:00.000	<b>ASSET</b> 110.170.24.226	[Unknown]
<b>EVENTS (7)</b>		
2024-07-17T21:03:23.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - unknown resource	personal_access_token.access_gran...
2024-07-17T21:03:28.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo1	git.clone
2024-07-17T21:03:33.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo2	git.clone
2024-07-17T21:03:38.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo3	git.clone
2024-07-17T21:03:43.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo4	git.clone
2024-07-17T21:03:48.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo5	git.clone
2024-07-17T21:03:53.000	<b>USER_RESOURCE_UPDATE_CONTENT</b> [GIT] threat-punter - threatpunter1/repo6	git.clone

2024-06-27 00:00:00

IP\_ADDRESS

Raw Log  Event/Entity

Entity Information

0 selected

00:00:00.000

Log Source: Spur data feeds

metadata.collected\_timestamp: "2024-06-27T00:00:00Z"  
 metadata.vendor\_name: "Spur"  
 metadata.product\_name: "Spur Feeds"  
 metadata.entity\_type: "IP\_ADDRESS"  
 metadata.interval.start\_time: "2024-06-27T00:00:00Z"  
 metadata.interval.end\_time: "2024-08-25T00:00:00Z"  
 metadata.threat[0].threat\_name: "VPN and proxy IP addresses"  
 metadata.event\_metadata.id: b"AAAAAN3Wk3N24+H1VuPKj1Y7Qb4AAAABwAAAAAAA="

metadata.event\_metadata.base\_labels.log\_types[0]: "SPUR\_FEEDS"

metadata.event\_metadata.base\_labels.allow\_scoped\_access: true

entity.ip[0]: "110.170.24.226"  
 entity.artifact.ip: "110.170.24.226"  
 entity.artifact.location.city: "Bang Rak"  
 entity.artifact.location.state: "Bangkok"  
 entity.artifact.location.country\_or\_region: "TH"  
 entity.artifact.as\_owner: "TRUE INTERNET Co Ltd"  
 entity.artifact.asn: 7470  
 entity.artifact.tags[0]: "VPN"  
 entity.artifact.tags[1]: "FILE\_SHARING"  
 entity.artifact.tags[2]: "MULLVAD\_VPN\_USER"

# Documenting the Detection (1)

## ADS Framework

By Wade Wells

This bot helps you write Alerting and Detection Strategies Framework documents for detections. Give it some detection logic or description of the detection you wrote and it will create a fully filled out ADS document <https://github.com/palantir/alerting-detection-strategy-framework>

Write me and ADS  
for this detection  
logic "process ==..."

Write me an ADS document for this new detection please. The detection identifies the following behavior in a GitHub Enterprise Environment:

1. A user account grants permissions to a GitHub Personal Access token.
2. The personal access token is used to clone more than 5 private GitHub repositories.
3. The source IP address for the activity is attributed to a VPN service by the vendor, Spur.

The detection relies on the data sources, GitHub Enterprise Audit Logs and Spur IP address

ChatGPT can make mistakes. Check important info.

# Documenting the Detection (2)

The screenshot shows the ADS Framework AI Assistant interface. At the top left is a dropdown menu labeled "ADS Framework". At the top right are two icons: an upward arrow and a pink circle containing a white "TH".

The main area contains a dark gray box with rounded corners. Inside, there is a text input field with placeholder text: "Write me an ADS document for this new detection please. The detection identifies the following behavior in a GitHub Enterprise Environment:". Below this is a list of three numbered items:

1. A user account grants permissions to a GitHub Personal Access token.
2. The personal access token is used to clone more than 5 private GitHub repositories.
3. The source IP address for the activity is attributed to a VPN service by the vendor, Spur.

At the bottom of the box, it says: "The detection relies on the data sources, GitHub Enterprise Audit Logs and Spur IP address feeds."

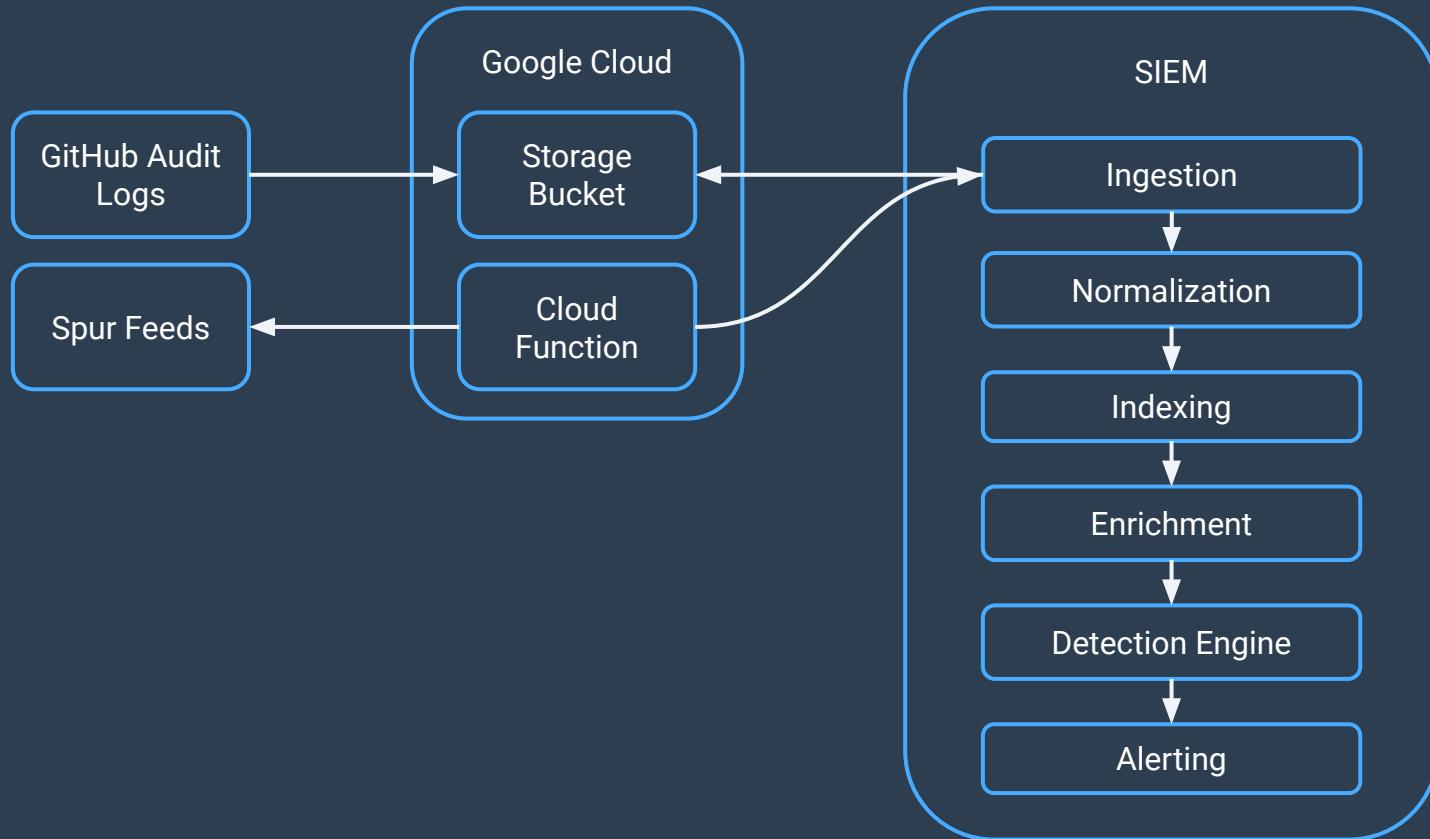
Below the main box, the word "Goal" is displayed next to a gear icon. To its right is a text box: "Detect unauthorized usage of GitHub Personal Access Tokens (PATs) for cloning multiple private repositories from a VPN-associated IP address." A cursor arrow points towards this text.

Below "Goal" is the word "Categorization" next to a downward arrow icon. Underneath are two circular buttons: one labeled "Initial Access / Val." with a tooltip "Improve false positives?" and another labeled "Add data sources?".

At the bottom of the interface is a footer bar with a message icon and the text "Message ADS Framework". On the far right of the footer is a circular icon with an upward arrow.

# Monitoring Your Data Pipeline

# Simple Data Pipeline Overview



# Reasons to Monitor Your Data Pipeline

- Environments drift over time
- Logging interruptions or spikes
- Parsing errors
  - Vendors change their logging schema
- Latency affecting log ingestion, normalization, indexing, or detection engine
- Highly recommend Josh Liburdi's [Building Better Hunt Data](#) presentation
  - Deep dive on monitoring & improving data quality



# Example Technique for Proactive Data Pipeline Monitoring

- We need to detect issues with our data pipeline
- “Silent asset” detections can generate false positives
- One option is to implement health checks for systems that are important to us
  - Carry out a basic read operation against monitored systems
  - Validate that the SIEM ingested, normalized, and indexed the event(s)
- Health check jobs can run in your SOAR, automation tool, or CI/CD pipeline



# Example: GitHub Audit Log Health Check (1)

- Health check consists of two GitHub Actions jobs
- First job is scheduled to run daily
- Simple API call to retrieve info for the GitHub Enterprise we're monitoring

## ▼ Ping monitored systems

```
1 ► Run python -m health_checks.monitored_systems
18 18-Jul-24 18:07:34 UTC | INFO | github_health_check | Performing GitHub API health check
19 18-Jul-24 18:07:34 UTC | INFO | github_health_check | Attempting to retrieve GitHub organization information for threatpunter1
20 18-Jul-24 18:07:34 UTC | DEBUG | _new_conn | Starting new HTTPS connection (1): api.github.com:443
21 18-Jul-24 18:07:34 UTC | DEBUG | _make_request | https://api.github.com:443 "GET /orgs/threatpunter1 HTTP/1.1" 200 None
22 18-Jul-24 18:07:34 UTC | INFO | github_health_check | GitHub API is reachable and authentication is successful.
```



Health Check - Monitored Systems passing

# Example: GitHub Audit Log Health Check (2)

Second job validates that the events from GitHub were ingested, normalized, indexed, and are searchable in the SIEM

## ✓ Validate SIEM log ingestion from monitored systems

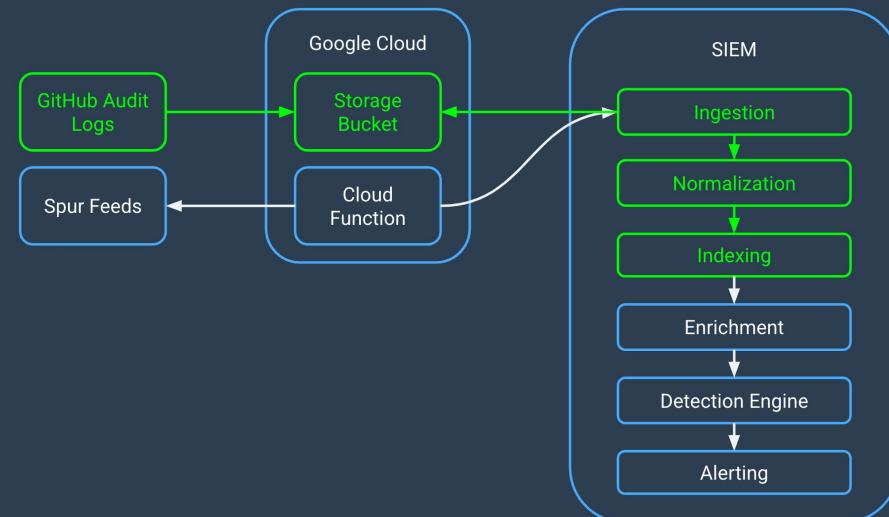
```
1 ►Run python -m health_checks.log_ingestion
17 18-Jul-24 18:24:53 UTC | INFO | validate_github_log_ingestion | Running UDM search to validate GitHub log ingestion:
18 {
19     "query": "metadata.event_type = \"USER_RESOURCE_UPDATE_CONTENT\" AND metadata.log_type = \"GITHUB\" AND metadata.product_name =
    \"GITHUB\" AND metadata.product_event_type = \"api.request\" AND extracted.fields[\"url_path\"] = \"/organizations/166444767\" AND
    network.http.method = \"GET\" AND extracted.fields[\"user_programmatic_access_name\"] = \"github-api-health-check\"",
20     "start_time": "2024-07-18T15:24:53Z",
21     "end_time": "2024-07-18T18:24:53Z"
22 }
23 18-Jul-24 18:24:57 UTC | INFO | validate_github_log_ingestion | 1 events returned from UDM search:
24 {
25     "query": "metadata.event_type = \"USER_RESOURCE_UPDATE_CONTENT\" AND metadata.log_type = \"GITHUB\" AND metadata.product_name =
```



Health Check - SIEM Log Ingestion passing

# Example: GitHub Audit Log Health Check (3)

- This isn't a comprehensive solution for monitoring *all* components of your data pipeline
- Intended to be a practical example to spark ideas
- With a bit of code, we have some confidence that security events from our GitHub environment are being ingested into our SIEM and are searchable



# Testing Detections

# Testing Your Detections: Don't Skip This Step!

- Validates that your detection & alerting capabilities are working
- Common issues impacting detections:
  - Logging interruptions or parsing errors
  - Misconfigured data sources
  - Vendors changing their logging schema
- Testing helps you identify & fix issues before malicious activity goes unnoticed



[State of SIEM Detection Risk \(2024\)](#) – CardinalOps

# Example: Testing the GitHub Detection (1)

- How can we automate the regular testing of the new GitHub detection?
- Perfect world: Create a test that simulates the attacker behavior end-to-end and validates that the detection generated an alert
- Real world:
  - Not possible to create & configure a new Personal Access Token via GitHub's API
    - Do we want to do this anyway? Probably not...
  - We don't want to carry out sensitive actions from a VPN service
  - Developing complex tests is expensive

# Example: Testing the GitHub Detection (2)

- Two GitHub Actions jobs scheduled to run daily
- First job replays historical GitHub audit log events to the SIEM for ingestion
  - Modify the timestamps in the logs prior to ingestion
  - Label the events so it's clear they're related to testing
- Second job validates that the detection generated an alert

## Replay logs to SIEM for ingestion

```
1 ►Run python -m health_checks.log_replay
18 18-Jul-24 19:16:31 UTC | INFO | <module> | Replying logs for ingestion into Google SecOps
19 18-Jul-24 19:16:31 UTC | INFO | <module> | Loading raw log entries from file /home/runner/work/detection-engineering-demo-1/detection-
engineering-demo-1/health_checks/logs/github.json
20 18-Jul-24 19:16:31 UTC | INFO | <module> | Loaded 7 raw log entries from file /home/runner/work/detection-engineering-demo-1/detection-
engineering-demo-1/health_checks/logs/github.json
21 18-Jul-24 19:16:31 UTC | INFO | <module> | Attempting to ingest raw log entries from file /home/runner/work/detection-engineering-demo-
1/detection-engineering-demo-1/health_checks/logs/github.json with labels [{"key': 'log_replay', 'value': 'true'}, {'key':
'log_replay_time', 'value': '2024-07-18 19:16:31.643298+00:00'}]
```

# Example: Testing the GitHub Detection (3)

Errors should be routed to the security team to investigate

```
✓ ✘ Validate detections were generated by tested rules 1s

1 ► Run python -m health_checks.rules
16 18-Jul-24 21:54:48 UTC | INFO | <module> | Searching for alerts for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2 with start time
2024-07-18T21:24:48Z and end time 2024-07-18T21:54:48Z
17 18-Jul-24 21:54:49 UTC | INFO | <module> | Retrieved a total of 0 alerts for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2
18 18-Jul-24 21:54:49 UTC | INFO | <module> | No alerts found for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2 that were generated by
testing. Check data pipeline and rule for issues.
19 Traceback (most recent call last):
20   File "/opt/hostedtoolcache/Python/3.10.14/x64/lib/python3.10/runpy.py", line 196, in _run_module_as_main
21     return _run_code(code, main_globals, None,
22   File "/opt/hostedtoolcache/Python/3.10.14/x64/lib/python3.10/runpy.py", line 86, in _run_code
23     exec(code, run_globals)
24   File "/home/runner/work/detection-engineering-demo-1/detection-engineering-demo-1/health_checks/rules.py", line 127, in <module>
25     raise Exception(
26 Exception: No alerts found for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2 that were generated by testing. Check data pipeline and
rule for issues.
27 Error: Process completed with exit code 1.
```



# Example: Testing the GitHub Detection (4)

- Check if any of the alerts from the detection were generated by testing
  - e.g. log\_replay: true
- Automatically close alerts generated by testing
- This approach can be applied to other detections where end-to-end, automated attack simulation isn't possible/needed

```
▼  Validate detections were generated by tested rules

1 ►Run python -m health_checks.rules
16
16 18-Jul-24 19:34:52 UTC | INFO | <module> | Searching for alerts for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2 with start time
2024-07-18T16:34:52Z and end time 2024-07-18T19:34:52Z
17 18-Jul-24 19:34:58 UTC | INFO | <module> | Retrieved a total of 1 alerts for rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2
18 18-Jul-24 19:34:58 UTC | INFO | <module> | Checking alert ID de_91a3791e-e98a-172f-99cd-7d210867facc created at 2024-07-
18T19:31:20.076663Z by rule ID ru_9bfedf55-d06b-49b2-ad42-63d0d3b43ce2 for indicators from rule testing
19 18-Jul-24 19:34:58 UTC | INFO | <module> | Alert ID de_91a3791e-e98a-172f-99cd-7d210867facc was created via testing activity
```

# Key Takeaways

# Key Takeaways

- Detection Engineering is a proactive approach for identifying attacker behavior
  - Goes beyond generic, out-of-the-box detections/signatures
  - Detect potential security incidents before they can cause significant damage
- Monitor your data pipeline to ensure that your data is available when you need it
- Test your detections to validate that you'll be alerted to malicious activity
- Teams require a diverse skill set
  - Combination of technical expertise, engineering mindset, curiosity, data analysis skills, etc

# Useful Resources

Monitoring for Suspicious GitHub Activity with Google Security Operations – David French

A Recipe for Improving SecOps Detections – John Stoner

ADS Framework AI Assistant – Wade Wells

Implementing a Modern Detection Engineering Workflow – Dan Lussier

SpecterOps Training Course – Adversary Tactics: Detection

Practical Threat Detection Engineering – Megan Roddie

# Thank you

**David French**

Staff Adoption Engineer, Google Cloud  
[@threatpunter](#)

# Q&A

# Appendix

# Tactics, Techniques, and Procedures (TTPs) Explained

# What is MITRE ATT&CK?

- Attack lifecycle framework – Helps us break down the phases of an attack and the techniques used in each phase
- A knowledge base of attacker behavior based on real-world observations
- Provides us with a common language to describe an attack and attacker TTPs

Reconnaissance 10 techniques	Resource Development 8 techniques	Initial Access 10 techniques	Execution 14 techniques	Persistence 20 techniques	Privilege Escalation 14 techniques	Defense Evasion 43 techniques	Credential Access 17 techniques	Discovery 32 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 18 techniques	Exfiltration 9 techniques	Impact 14 techniques
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (6)	Abuse Elevation Control Mechanism (6)	Adversary-in-the-Middle (3)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (3)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal	
Gather Victim Host Information (4)	Acquire Infrastructure (8)	Drive-by Compromise	Command and Scripting Interpreter (10)	BITS Jobs	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction	
Gather Victim Identity Information (3)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	BITS Jobs	Build Image on Host	Browser Information Discovery	Lateral Tool Transfer	Audio Capture	Content Injection	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact	
Gather Victim Network Information (6)	Compromise Infrastructure (8)	External Remote Services	Develop Capabilities (4)	Container Initialization Scripts (5)	Boot or Logon Autostart Execution (14)	Debugger Evasion	Cloud Infrastructure Discovery	Cloud Service Dashboard	Automated Collection	Data Encoding (2)	Exfiltration Over C2 Channel	Data Manipulation (3)	
Gather Victim Org Information (4)	Establish Accounts (3)	Hardware Additions	Deploy Container	Exploitation for Client Execution	Browser Extensions	Deobfuscate/Decode Files or Information	Forced Authentication	Cloud Service Hijacking (2)	Browser Session Hijacking	Data Obfuscation (3)	Defacement (2)	Disk Wipe (2)	
Phishing for Information (4)	Phishing (4)	Obtain Capabilities (7)	Obtain Capabilities (7)	Inter-Process Communication (3)	Compromise Host Software Binary	Deploy Container	Cloud Service Discovery	Cloud Storage Object Discovery	Clipboard Data	Data from Cloud Storage	Exfiltration Over Other Network Medium (1)	Endpoint Denial of Service (4)	
Search Closed Sources	Replication Through			Create or Modify System	Domain or Tenant	Forge Web Credentials (2)	Input	Dynamic Resolution	Dynamic Resolution			Financial Theft	

# TTPs Explained: Tactics

- **Tactics** represent the **why** behind an attacker's actions
  - The goal the attacker is trying to achieve during a particular stage of an attack

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	43 techniques	17 techniques
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (6)	Abuse Elevation Control Mechanism (6)	Abuse Elevation Control Mechanism (6)	Adversary-in-the-Middle (3)
Gather Victim Host Information (4)	Acquire Infrastructure (8)	Drive-by Compromise	Command and Scripting Interpreter (10)	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)
Gather Victim Identity Information (3)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	BITS Jobs	Build Image on Host	Credentials from Password Stores (6)
Gather Victim Network Information (6)	Compromise Infrastructure (8)	External Remote Services	Deploy Container	Account Manipulation (6)	Debugger Evasion	Deobfuscate/Decode Files or Information	Exploitation for Credential Access
Gather Victim Org Information (4)	Develop Capabilities (4)	Establish Accounts (3)	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Deploy Container	Direct Volume Access	Forced Authentication
Phishing for Information (4)	Obtain Capabilities (7)	Hardware Additions	Inter-Process Communication (3)	Browser Extensions	Domain or Tenant	Forge Web Credentials (2)	
Search Closed Sources (2)	Phishing (4)	Replication	Inter-Process Communication (3)	Compromise Host Software Binary	Create or		

# TTPs Explained: Techniques

**Techniques** represent **how** an attacker achieves a tactical objective by performing an action

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	43 techniques	17 techniques
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (6)	Abuse Elevation Control Mechanism (6)	Abuse Elevation Control Mechanism (6)	Adversary-in-the-Middle (3)
Gather Victim Host Information (4)	Acquire Infrastructure (8)	Drive-by Compromise	Command and Scripting Interpreter (10)	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)
Gather Victim Identity Information (3)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	BITS Jobs	Build Image on Host	Credentials from Password Stores (6)
Gather Victim Network Information (6)	Compromise Infrastructure (8)	External Remote Services	Deploy Container	Account Manipulation (6)	Debugger Evasion	Deobfuscate/Decode Files or Information	Exploitation for Credential Access
Gather Victim Org Information (4)	Develop Capabilities (4)	Establish Accounts (3)	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Autostart Execution (14)	Deploy Container	Forced Authentication
Phishing for Information (4)	Obtain Capabilities (7)	Hardware Additions	Inter-Process Communication (3)	Browser Extensions	Deobfuscate/Decode Files or Information	Direct Volume Access	Forge Web Credentials (2)
Search Closed Sources (2)	Phishing (4)	Replication	Inter-Process Communication (3)	Compromise Host Software Binary	Create or	Domain or Tenant	

# TTPs Explained: Sub-Techniques

**Sub-Techniques** provide greater detail and granularity on how a Technique might be implemented

Reconnaissance	Resource Development	Initial Access	Execution
10 techniques	8 techniques	10 techniques	14 techniques
<p>Active Scanning (3)</p> <p>Gather Victim Host Information (4)</p> <p>Gather Victim Identity Information (3)</p> <p>Gather Victim Network Information (6)</p> <p>Gather Victim Org Information (4)</p> <p>Phishing for Information (4)</p> <p>Search Closed Sources (2)</p> <p>Search Open Technical Databases (5)</p> <p>Search Open Websites/Domains (3)</p>	<p>Acquire Access</p> <p>Acquire Infrastructure (8)</p> <p>Compromise Accounts (3)</p> <p>Compromise Infrastructure (8)</p> <p>Develop Capabilities (4)</p> <p>Establish Accounts (3)</p> <p>Obtain Capabilities (7)</p> <p>Stage Capabilities (6)</p>	<p>Content Injection</p> <p>Drive-by Compromise</p> <p>Exploit Public-Facing Application</p> <p>External Remote Services</p> <p>Hardware Additions</p> <p>Phishing (4)</p> <p>Replication Through</p>	<p>Cloud Administration Command</p> <p>Command and Scripting Interpreter (10)</p> <p>Container Administration Command</p> <p>Deploy Container</p> <p>Exploitation for Client Execution</p> <p>Inter-Process Communication (3)</p> <p>Native API</p> <p>Scheduled Task/Job (5)</p> <p>Serverless Execution</p>

# Reviewing a MITRE ATT&CK Technique

Home > Techniques > Enterprise > Phishing > Spearphishing Attachment

## Phishing: Spearphishing Attachment

Other sub-techniques of Phishing (4)

Adversaries may send spearphishing emails with a malicious attachment in an attempt to gain access to victim systems. Spearphishing attachment is a specific variant of spearphishing. Spearphishing attachment is different from other forms of spearphishing in that it employs the use of malware attached to an email. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon User Execution to gain execution.<sup>[1]</sup> Spearphishing may also involve social engineering techniques, such as posing as a trusted source.

There are many options for the attachment such as Microsoft Office documents, executables, PDFs, or archived files. Upon opening the attachment (and potentially clicking past protections), the adversary's payload exploits a vulnerability or directly executes on the user's system. The text of the spearphishing email usually tries to give a plausible reason why the file should be opened, and may explain how to bypass system protections in order to do so. The email may also contain instructions on how to decrypt an attachment, such as a zip file password, in order to evade email boundary defenses. Adversaries frequently manipulate file extensions and icons in order to make attached executables appear to be document files, or files exploiting one application appear to be a file for a different one.

Procedure Examples

ID	Name	Description
C0028	2015 Ukraine Electric Power Attack	During the 2015 Ukraine Electric Power Attack, Sandworm Team obtained their initial foothold into many IT systems using Microsoft Office attachments delivered through phishing emails. <sup>[2]</sup>
G0018	admin@338	admin@338 has sent emails with malicious Microsoft Office documents attached. <sup>[3]</sup>

Technique

Tactic

Procedures

ID: T1566.001  
Sub-technique of: T1566  
① Tactic: Initial Access  
① Platforms: Linux, Windows, macOS  
Contributors: Philip Winther  
Version: 2.2  
Created: 02 March 2020  
Last Modified: 31 January 2024

Version Permalink

# Mapping Threat Intelligence to TTPs

# Mapping Threat Intel to TTPs: Why bother?

- Helps break down the phases of an attack and understand the techniques used in each phase
- Provides us with a common language to categorize observed behaviors
- Focus detection efforts on specific & relevant behaviors
- Makes it easier to understand current coverage against attacker behavior

Reconnaissance 10 techniques	Resource Development 8 techniques	Initial Access 10 techniques	Execution 14 techniques	Persistence 20 techniques	Privilege Escalation 14 techniques	Defense Evasion 43 techniques	Credential Access 17 techniques	Discovery 32 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 18 techniques	Exfiltration 9 techniques	Impact 14 techniques
Active Scanning (3)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (6)	Abuse Elevation Control Mechanism (6)	Adversary-in-the-Middle (3)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (3)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal	
Gather Victim Host Information (4)	Acquire Infrastructure (8)	Drive-by Compromise	Command and Scripting Interpreter (10)	BITS Jobs	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Transfer Over Alternative Protocol (3)	Data Destruction	
Gather Victim Identity Information (3)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	Boot or Logon Autostart Execution (14)	Account Manipulation (6)	BITS Jobs	Browser Information Discovery	Internal Spearphishing	Audio Capture	Content Manipulation (3)	Defacement (2)	Data Encrypted for Impact	
Gather Victim Network Information (6)	Compromise Infrastructure (8)	External Remote Services	Deploy Container	Boot or Logon Initialization Scripts (5)	Build Image on Host	Debugger Evasion	Cloud Infrastructure Discovery	Lateral Tool Transfer	Automated Collection	Exfiltration Over C2 Channel	Data Manipulation (3)	Data Manu	
Gather Victim Org Information (4)	Develop Capabilities (4)	Hardware Additions	Deploy Container	Browser Extensions	Boot or Logon Autostart Execution (14)	Deobfuscate/Decode Files or Information	Cloud Service Dashboard	Remote Service Session Hijacking (2)	Browser Session Hijacking	Exfiltration Over Other Network Medium (4)	Disk Wipe (2)	Endpoint Denial of Service (4)	
Phishing for Information (4)	Establish Accounts (3)	Exploitation for Client Execution	Exploit Process Communication (3)	Compromise Host Software Binary	Boot or Logon Initialization Scripts (5)	Deploy Container	Cloud Service Discovery	Clipboard Data	Clipboard Data	Dynamic Resolution (1)	Financial Theft	Financial Theft	
Search Closed Sources (3)	Obtain Capabilities (7)	Replication Through	Replication Through	Create or Modify System	Direct Volume Access	Forge Web Credentials (2)	Cloud Storage Object Discovery	Domain or Tenant	Data from Cloud Storage	Dynamic Resolution (1)	Fileless Persistence (1)	Fileless Persistence (1)	

# Mapping Threat Intel to TTPs (1)

- The threat group compromises a software developer's Okta user account via a Smishing campaign
- They're using VPN services like Mullvad VPN, NordVPN, and ExpressVPN to mask their IP address and geolocation
- They create a GitHub Personal Access Token under the victim's account and proceed to clone all of the GitHub repositories that the account has access to
- They're using an open-source tool to steal the contents of code repositories in an automated fashion

# Mapping Threat Intel to TTPs (2)

- Tactic: Initial Access ([TA0001](#))
- Technique: Phishing ([T1566](#))
- Procedures:
  - Register a domain that looks similar to the target organization's Okta SSO subdomain
    - e.g. company-sso[.]com or company-okta[.]net
  - Host a fake Okta SSO login portal using the new domain
  - Send an SMS message to the target user asking them to sign in to the fake Okta portal
  - Harvest the user's username, password, and one-time passcode (OTP)
  - Using a VPN/anonymization service, login to the target organization's Okta portal as the target user
  - Access the target web application(s) as the authenticated user

# Mapping Threat Intel to TTPs (3)

- The threat group compromises a software developer's Okta user account via a Smishing campaign
- They're using VPN services like Mullvad VPN, NordVPN, and ExpressVPN to mask their IP address and geolocation
- They create a GitHub Personal Access Token under the victim's account and proceed to clone all of the GitHub repositories that the account has access to
- They're using an open-source tool to steal the contents of code repositories in an automated fashion

# Mapping Threat Intel to TTPs (4)

- Tactic: Persistence ([TA0003](#))
- Technique: Create Account ([T1136](#)) – Not always a perfect fit, but that's ok
- Procedures:
  - Create a new GitHub Personal Access Token (PAT) under the compromised user's account
  - Grant the PAT access to all of the code repositories that the user has access to

---
- Tactic: Collection ([TA0009](#))
- Technique: Data from Information Repositories: Code Repositories ([T1213.003](#))
- Procedures:
  - Use the open source tool, "ghorg" to clone all of the GitHub code repositories that the PAT has access to

# Preventative Security Controls

# Preventative Security Controls

- Configure a Personal Access Token (PAT) policy for your GitHub organization(s)
  - Enforce an approval policy
- Encourage users to create short-lived tokens for specific tasks
- Limit permissions granted to tokens based on principle of least privilege
- Configure an IP allow list for your GitHub organization (if possible)
- Require security keys for GitHub two-factor-authentication
- Monitor for secrets in code, CI/CD scripts, wikis, etc
  - Attackers will find them and use them against you