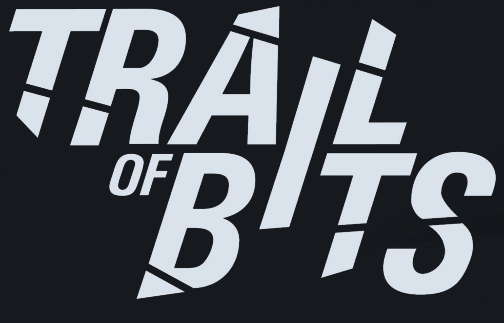


Systems security in practice: TRAIL threat modeling

Kelly Kaoudis, kelly.kaoudis@trailofbits.com

CleverHans Lab | April 19, 2024

Outline

- Academic threat modeling
- TRAIL: industry threat modeling at Trail of Bits
  - Security controls
  - Actors
  - Components
  - Trust boundaries and trust zones
  - Threat scenarios
  - Findings!

Main idea: threat modelling informs and enables making good system-level security decisions

A prototype's threat model...

- Manage scope, e.g. "side channel attacks are out of scope", "attackers with local hardware access are not considered"
- Actors (attackers, users) in the system
- Capabilities by actor, e.g., "a remote attacker can..."; "a user knows..."
- Constrain system and environment
  - What components, actors, data can be protected (by the prototype)
  - What components, actors, data will be affected (by the attack or weakness)
  - Even... what emergent properties

Your expectations?

Principle of falsifiability

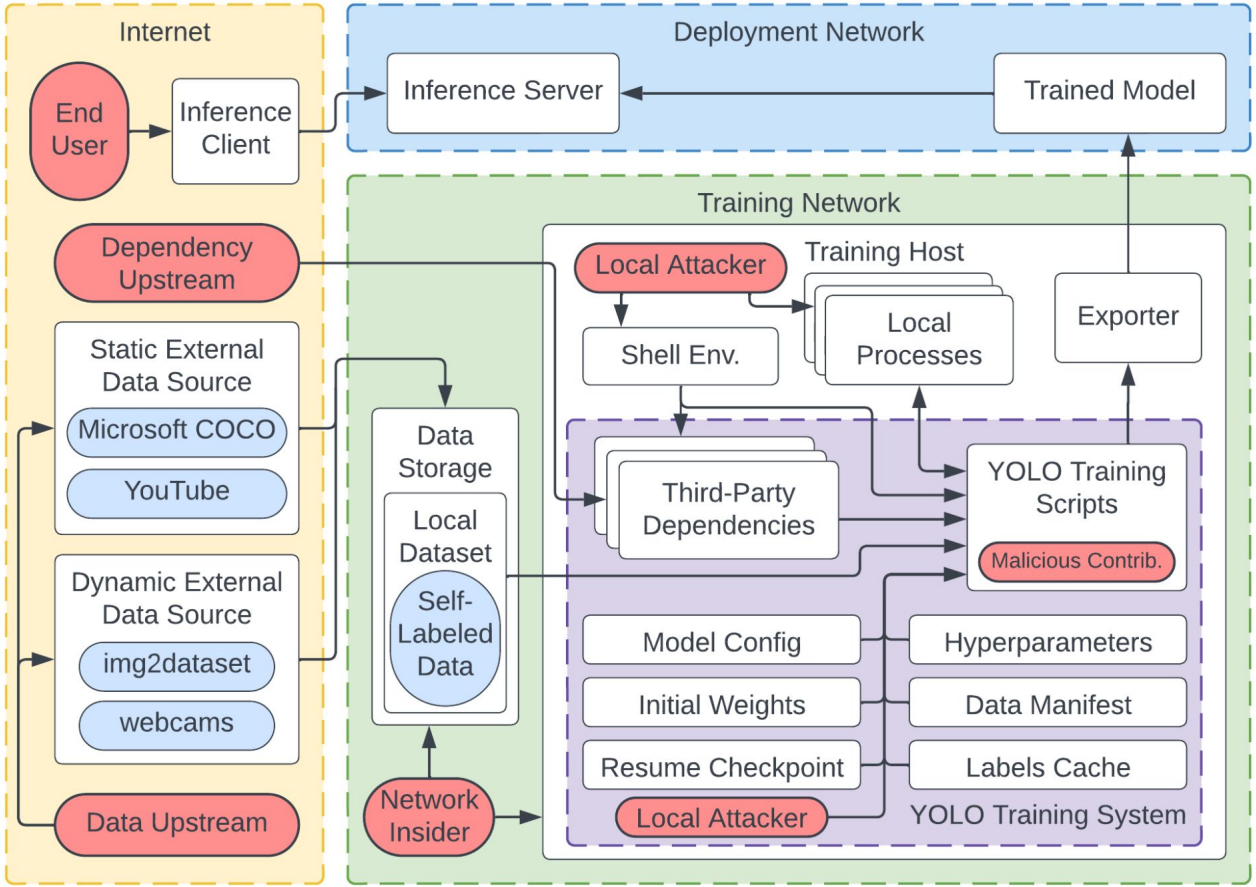
- Sets appropriate reader/reviewer/client expectations
- Enables objective evaluation
  - Papers: Keep reviewer 2's thought process on track!
  - Industry: If a client agrees to the conditions that enable a finding, they are more likely to agree the finding could happen
- Allows for "definition of done"
- Enables future work that refutes or builds on this work

Threat and Risk Analysis Informed Lifecycle

Kelly Kaoudis | CleverHans Lab | April 19, 2024

YOLOv7 Data Flow

Spencer Michaels, Maciej Domanski,
Alvin Crighton, Heidy Khlaaf



Systems thinking

- Everything is interconnected (dependencies)
- Emergent properties
- Protections and countermeasures will layer
- Boundaries: input, output, exchange
- ***Design-level*** weaknesses and vulns

Security Controls

- Applicable categories of security defences
- Example categories:
  - Contingency Planning
  - PII Processing and Transparency
  - Auditing and Accountability
  - Awareness and Training
- Are defences in the category implemented, or missing (a gap)?

“A vulnerability is any trust assumption involving people, processes, or technology that can be violated in order to exploit a system,”
NIST Special Publication 800-154

https://csrc.nist.gov/CSRC/media/Publications/sp/800-154/draft/documents/sp800_154_draft.pdf

Mozilla’s RRA (Rapid Risk Assessment)

- What does the system or application do?
- What data can it process or store?
- Confidentiality: What happens if all the data is disclosed to the world?
- Integrity: What if data is incorrect, misleading, website defaced, etc.?
- Availability: What if data or service is missing, deleted, unreachable?
- Impact (reputation, finances, productivity, system usability…)

NIST SP 800-154: data-centric threat modelling

- Step 1: identify and characterise the system and data of interest
- Step 2: identify and select the attack vectors to be included in the model
- Step 3: characterise the security controls for mitigating the attack vectors
- Step 4: analyse the resulting threat model

Threat modeling the TRAIL of Bits way

1. Agree with client on initial scope
2. Discovery*: learn the system, actors, organization, process
3. Agree with client on system model, assumptions
4. Show the gaps: threat scenarios, findings
5. Security controls maturity analysis
6. Report delivery and discussion

**refine scope; discuss each step of discovery with the client so they know (and expect) what we are doing*

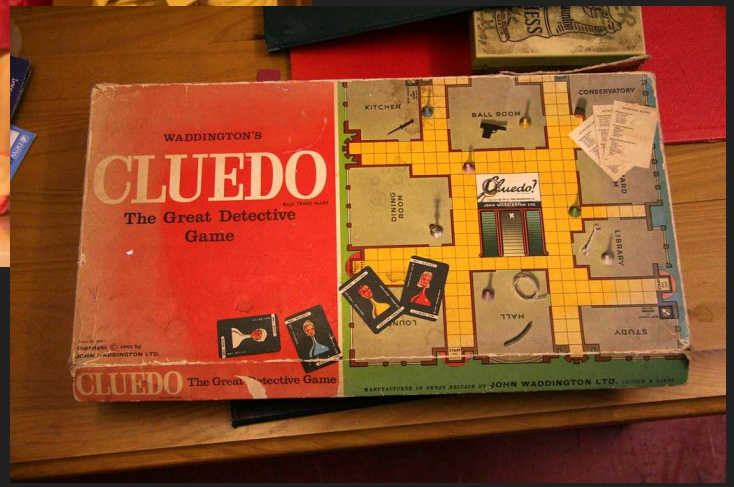
Threat: motive, method

-  : any type of fault or error that results in unexpected output or behaviour
- Vulnerability: an incorrect assumption of security where there is actually weakness (one bug type; )
- Threat = motive (data or access within the system) + method (way of exploiting the vuln)

Who’s in the system?

- Users, system admins or operators, attackers
- What’s in the system that they might want (sensitive data, privileged access, persistence)?
- What *should* each actor type be able to do?
- What *can* they do?
- What do they know?

Clue at the Citadel Theatre. Author: Nanc Price, Source: Alberta Prime Times.



Eclipse Jetty System Actors

Kelly Kaoudis, Spencer Michaels

External Attacker	An external attacker is an attacker on the public network (internet) from which at least one Jetty instance is accessible. This attacker can observe and analyze Jetty source commits as they land in the public repository for exploitable features.
Internal Attacker	An attacker on a private or application network from which at least one Jetty instance is accessible.
Client	“Client” refers to either a client of a Jetty server instance that can integrate the Jetty client libraries or a wholly distinct networked application.
Local Attacker	A local attacker is an attacker who controls a process or user account on the same host as the Jetty instance and can affect the system environment, including the filesystem.
Jetty Contributor	A non-maintainer Jetty contributor.
Jetty Maintainer	A core Jetty contributor. Maintainers must review and approve pull requests prior to merging them.
Application Developer	An application developer creates, maintains, and updates applications deployed via Jetty.
Server Administrator	A server administrator administers a networked application that is either built with Jetty components, served via a Jetty instance embedded as a servlet container in another framework, or served via a standalone Jetty instance.
Server Deployer	A server deployer releases an application served via Jetty or built with Jetty components into the running environment. The deployer may not be a separate individual from the server administrator and application developer.

Components

<Component name>	<Component name> is a <noun> that <verb>s <specific types of data>. The <component name> can connect to <other component A> via <connection type specifics> and receives <specific data> from <another component B> over <connection type specifics>. *(Optionally, link to specific actor types that interact with the component)* <Actor> can <verb> the <component name>.

Eclipse Jetty Components

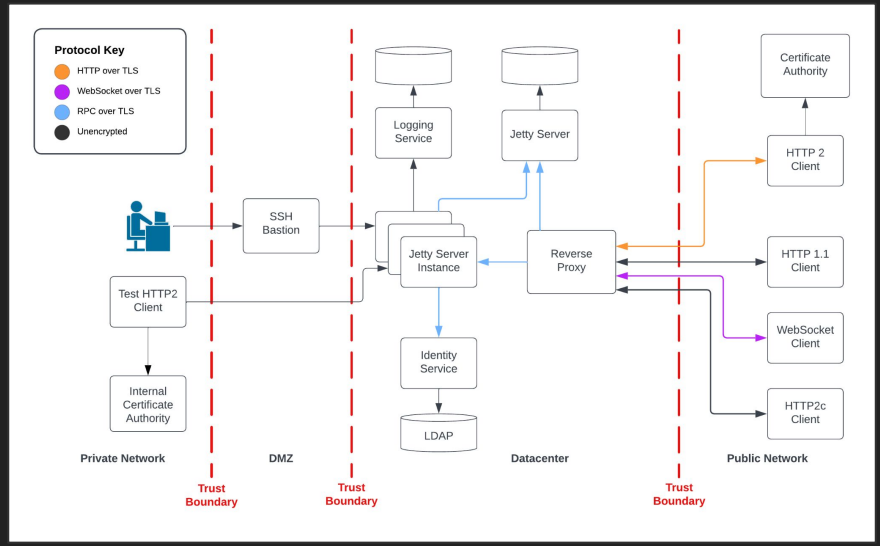
Kelly Kaoudis, Spencer Michaels

Source control	Source control includes the infrastructure that provides version control, hosts the Jetty codebase, facilitates pull requests and issues, and allows maintainers to release Jetty JARs and security advisories.	Application Specific Logic (*)	Developer-provided business logic connects with Jetty (and clients) via the application logic base APIs. This component is out of scope.
Jetty Client (*)	A client requests data from a Jetty server or from a server built with Jetty libraries. Client-side Jetty libraries may optionally be used to handle client network connections and parsing. This component is out of scope.	Server Side Component Libraries	Server-side component libraries are used to build Jetty-based web servers. These component libraries provide server-side connection and request handling and parsing support for protocols such as HTTP/1.1, HTTP/2, HTTP/3, WebSocket, and FastCGI.
Client Side Component Libraries	The deployer or administrator can add client-side component libraries to the Jetty server to form a microservice that can both receive and initiate connections and requests.	Reverse Proxy (*)	The reverse proxy fronts the Jetty-served application so that no Jetty instance needs be exposed to a public network directly. The reverse proxy can also handle TLS termination on behalf of a Jetty-served application. This component is out of scope.

Eclipse Jetty Components

Kelly Kaoudis, Spencer Michaels

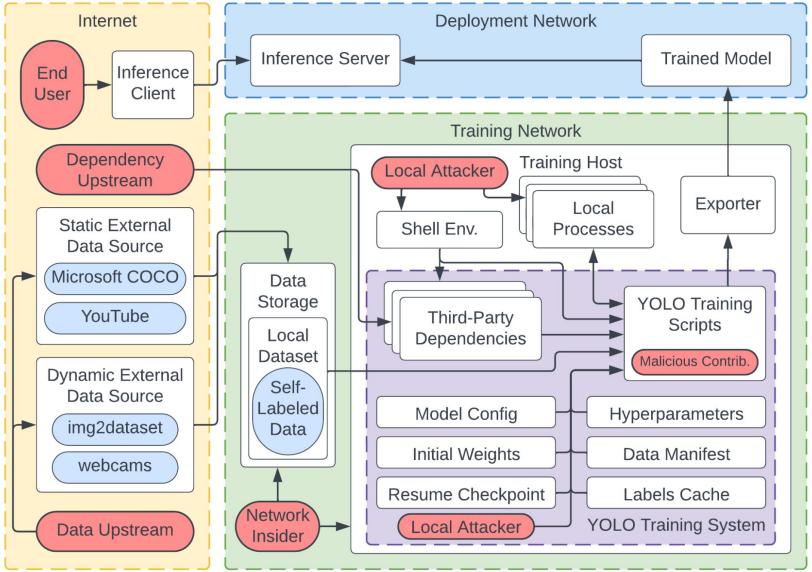
Source control	Source control includes the infrastructure that provides version control, hosts the Jetty codebase, facilitates pull requests and issues, and allows maintainers to release Jetty JARs and security advisories.	Application Specific Logic (*)	Developer-provided business logic connects with Jetty (and clients) via the application logic base APIs. This component is out of scope.
Jetty Client (*)	A client requests data from a Jetty server or from a server built with Jetty libraries. Client-side Jetty libraries may optionally be used to handle client network connections and parsing. This component is out of scope.	Server Side Component Libraries	Server-side component libraries are used to build Jetty-based web servers. These component libraries provide server-side connection and request handling and parsing support for protocols such as HTTP/1.1, HTTP/2, HTTP/3, WebSocket, and FastCGI.
Client Side Component Libraries	The deployer or administrator can add client-side component libraries to the Jetty server to form a microservice that can both receive and initiate connections and requests.	Reverse Proxy (*)	The reverse proxy fronts the Jetty-served application so that no Jetty instance needs be exposed to a public network directly. The reverse proxy can also handle TLS termination on behalf of a Jetty-served application. This component is out of scope.



YOLOv7 Trust Zone Connections

Spencer Michaels, Maciej Domanski, Alvin Crighton, Heidy Khlaaf

Origin Zone	Dest. Zone	Data Description	Connection Type	Authentication / Authorization
Internet	Training Network, YOLO System	Third-party dependencies of the YOLO system are retrieved and installed onto the model host.	HTTP	
Internet	YOLO System	Configuration files are downloaded from the internet and loaded by the YOLO scripts.	HTTP	
Training Host	YOLO System	Environment variables on the training host are loaded into YOLO's PyTorch execution environment.	POSIX APIs	Local user scope
Training Network	YOLO System	Configuration files located on the training host's local filesystem, or other hosts on the training network, are loaded by the YOLO scripts.	Local filesystem	Filesystem permissions
		...		



Systematically thinking about threats: STRIDE

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Expansion of authority

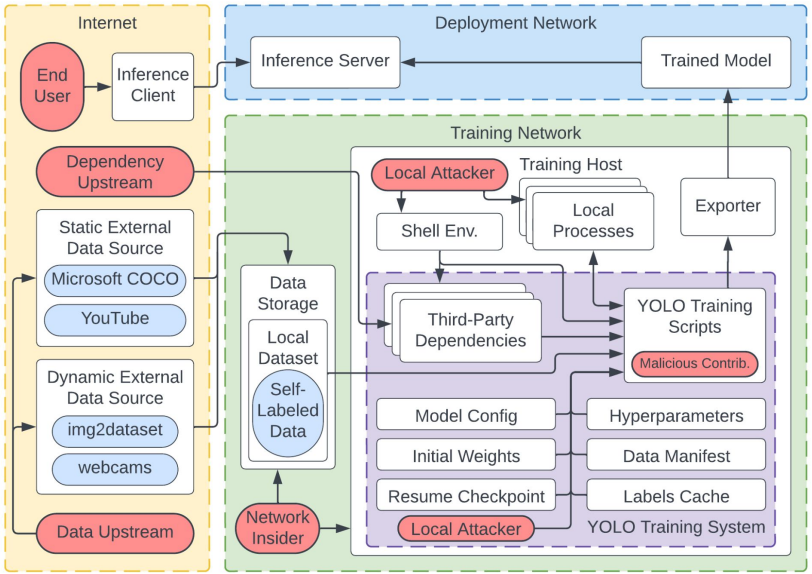
STRIDE: things that break user trust

- Spoofing violates authenticity
- Tampering violates integrity
- Repudiation violates non-repudiation
- Information disclosure violates confidentiality
- Denial of service violates availability
- Expansion of authority violates authorization (privilege enforcement)

YOLOv7 Threat Scenarios

Spencer Michaels, Maciej Domanski, Alvin Crighton, Heidy Khlaaf

Threat	Scenario	Actor(s)	Component(s)
Dataset compromise	A domain scraped by img2dataset expires and falls under an attacker's control, allowing the attacker to poison a small part of the dataset.	• External Attacker	• Dynamic External Data Source
Dataset compromise	An attacker poisons the dataset upstream, leading to mislabeled data in the local dataset.	• External Attacker	• Static & Dynamic External Data
Host compromise	An attacker sneaks malicious code into the codebase of the YOLO system or one of its dependencies, gaining control of the host machine.	• Contributor • External Attacker	• YOLO Scripts • Third Party Dependencies
YOLO process compromise	An attacker compromises a local process such as WandB or Tensorboard and writes to the YOLO training scripts' intermediate state files, corrupting the model weights.	Local Attacker	• Resume Checkpoint • Labels Cache
	...		



a finding = what if...

a gap + an actor + a vector

Severity	Difficulty
Informational	Undetermined
Undetermined	Low
Low	Medium
Medium	High
High	

risk = likelihood * impact

Low difficulty == high likelihood

Severity == impact

5. cURL treats localhost as secure by default

Severity: **Informational**	Difficulty: **High**
Type: Configuration Management	Finding ID: TOB-CURLTM-5
Target: cURL, libcurl	

Description
By default, cURL assumes that connection requests to `localhost`, `127.0.0.1`, and `[::1]` are secure and disables relevant security features, such as accepting the use of the `secure` cookie flag for insecure connections to `localhost` and cURL skipping name resolution checks. This may mislead cURL users into believing that their connections to `localhost` are secure.

Threat Scenario
A web developer uses cURL to make requests against a site they are developing and running on `http://localhost:8080`. Since cURL accepts and honors secure cookies from an insecure `localhost`, the developer assumes the application's behavior in `localhost` will match when it is deployed to production and makes assumptions about how the cookie flags will be treated when deploying to production.

Recommendations
Short term, explicitly document how cURL treats requests to `localhost` differently than requests to upstream servers.

Long term, update cURL so that it treats `localhosts` securely by default, and introduce a flag that users can use when calling cURL to turn off insecure behavior, such as disallowing cookies with the secure flag to be sent to `localhost` endpoints. This flag can work similarly to `-k`, which users can use when leveraging self-signed certificates to bypass validation.

6. Insufficient input validation strategy

Severity: **High**	Difficulty: **Medium**
Type: Configuration Management	Finding ID: TOB-CURLTM-6
Target: cURL, libcurl	

Description

cURL performs input sanitization using a denylist of characters rather than strongly validating characters against an allowlist, regex, or similar. For instance, cURL allows potentially unsafe characters into cookie jar files, which could lead to broken functionality. This behavior deviates from relevant RFC specifications such as RFC 1738, which defines a set of permitted characters for URIs and disallows all others.

Threat Scenario

A zero-day exploit that takes advantage of weak URI validation is used against applications that rely on libcurl. Attackers leverage the exploit to compromise the confidentiality, integrity, or availability of user data and services that rely on such applications.

6. Insufficient input validation strategy

Severity: **High**	Difficulty: **Medi**
Type: Configuration Management	Finding ID: TOB
Target: cURL, libcurl	

Description

cURL performs input sanitization using a denylist of characters validating characters against an allowlist, regex, or similar. For instance, cURL allows potentially unsafe characters into cookie jar files, which could lead to broken functionality. This behavior deviates from relevant RFC specifications such as RFC 1738, which defines a set of permitted characters for URIs and disallows all others.

Threat Scenario

A zero-day exploit that takes advantage of weak URI validation is used against applications that rely on libcurl. Attackers leverage the exploit to compromise the confidentiality, integrity, or availability of user data and services that rely on such applications.

Justification

The severity is high. Because validation relies in many cases on denylists, it is difficult to account for future attacks that could make cURL vulnerable to attacks allowing malicious actors to compromise users, perform privilege escalation, or use cURL to run custom code remotely.

The difficulty is medium. There are no immediate concerns regarding allowed characters which cURL may not account for in their deny lists. However, deny lists are difficult to maintain and provide little protection against potential zero-day attacks, as new exploits may rely on the use of characters such as '`\t`', which cURL may not verify against.

Recommendations

Short term, default to using allow lists for sanitization and validation strategies for the various parsing tasks that cURL performs, such as cookie and URI parsing routines.

Long term, review RFCs for the various protocols and strings that cURL works with and parses and assure that the code conforms to the expectations outlined in such documents. Additionally, follow recommendations for TOB-CURLTM-6.

TM5. Incident Response is not automated and is under-documented

Severity: High
Type: IR,RA
Component(s): All

Difficulty: High
Finding ID: TOB-VOATZ-TM05

Description
The implementation team noted that Incident Response and Threat Hunting processes were neither automated nor directly documented. Most IR or Hunt activities involved systems administrators sifting through logs manually via tools such as `grep`. Manual tooling increases the chances that an incident will be missed, both in terms of how long an incident occurs and what is the actual impact of the incident.

Justification
The severity is High for the following reasons:

- Missing or incomplete documentation does not in and of itself impact the normal operation of the system.
- However, missing documentation may hinder the correct implementation, remediation, or related activities such as incident response by the implementation team.
- Additionally, not alerting on incidents in an automated fashion increases the chance that incidents may be missed, or that more serious incidents will be missed by disaggregate data.

The difficulty is High for the following reasons:

- The implementation team must perform an inventory of all assets and data throughout the system.
- The team must also have previously completed TOB-VOAT-TM02: Missing and Incomplete Data Classification.

TM5. Incident Response is not automated and is under-documented

Severity: High
Type: IR,RA
Component(s): All

Difficulty: High
Finding ID: TOB-VOATZ-TM05

Description
The implementation team noted that Incident Response and Threat Hunting processes were neither automated nor directly documented. Most IR or Hunt activities involved systems administrators sifting through logs manually via tools such as `grep`. Manual tooling increases the chances that an incident will be missed, both in terms of how long an incident occurs and what is the actual impact of the incident.

Justification
The severity is High for the follo
- Missing or incomplete d
operation of the system.
- However, missing docum
remediation, or related a
team.
- Additionally, not alerting
that incidents may be mi
disaggregate data.

The difficulty is High for the foll
- The implementation team must perform an inventory of all assets and data throughout the system.
- The team must also have previously completed TOB-VOAT-TM02: Missing and Incomplete Data Classification.

Recommendation
Implement a robust incident response process that is both well documented and largely automated. This will require having a known-good host baseline, as per TOB-VOAT-TM16: Missing Host Verification Process, as well as a defined data classification, as per TOB-VOAT-TM02: Missing Data Classification. Furthermore, leave manual processes that rely upon tools such as `grep` or a list of regexes to search in Centralized Logging Solution for threat hunting and other exploratory exercises.

References
- NIST 800-53:IR Family
- NIST 800-61: Computer Security Incident Handling Guide

Thank you!

Check out more of our work: github.com/trailofbits/publications

Kelly Kaoudis
Senior Security Engineer
kelly.kaoudis@trailofbits.com