Intrusion Detection & Attack Scenario Reconstruction using Information Flow Analysis on Provenance Graphs

Venkat Venkatakrishnan

INRIA SUPSEC Workshop

01/25/2023

Advance Provenance Threat (APT)

Targeted cyber attacks on organizations are getting more sophisticated, more stealthy, and more serious.

Goal: to steal data, disrupt operations or destroy infrastructure.

- Generally carried out over a much longer time frame
  - Many APT campaigns remain undetected for months.
- Customized to the target, rather than using commodity tools/malware.
- Thousand of incidents every year affecting small to large organizations in different sectors.

Advance Provenance Threat (APT)

Targeted cyber attacks or ...icated, more stealthy, and more serious.

Goal: to steal data, disru...

- Generally carried out ...
  - Many APT campaigns ...
- Customized to the targ... malware.
- Thousand of incidents ...nizations in different sectors.

Equifax breaks down just how bad last year's data breach was

More than 99 percent of affected consumers had their Social Security numbers exposed.

Advance Provenance Threat (APT)

Targeted cyber attacks or ...icated, more stealthy, and more serious.

Goal: to steal data, disru...

- Ge...
  - M...
- Cu... ...malware.
- Th... ...nizations in different sec...

Equifax breaks down just how bad last year's data breach was

More than 99 percent of affected consumers had their Social Security numbers exposed.

The New York Times

Marriott Hacking Exposes Data of Up to 500 Million Guests

Advance Provenance Threat (APT)

Targeted cyber attacks or ...icated, more stealthy, and more serious.

Goal: to steal data, disru...

- Ge...
  - M...
- Cu... ...malware.
- Th... ...nizations in different sec...

Equifax breaks down just how bad last year's data breach was

More than 99 percent of affected consumers had their Social Security numbers exposed.

The New York Times

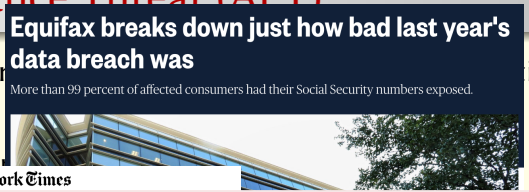
Marriott Hacking Exposes Data of Up to 500 Million Guests

Advance Provenance Threat (APT)

Targeted cyber attacks or ...icated, more stealthy, and more serious.

Goal: to steal data, disru...

- Ge...
  - M... o...
- Cu... ...malware.
- Th... ...nizations in different sec...



The New York Times

The annual cost of cyber-attacks to U.S. could exceed $100 billion.

[White House Council of Economic Advisers]

State-of-the-art Intrusion Detection

- APTs leverage different attack methods, and span multiple applications and hosts.
- Enterprises collect security audit information that can assist detection:
  - Security Information and Event Management (SIEM)
  - Intrusion Detection Systems (IDS)
  - Identity and access management tools
  - Application firewalls

Problem: Hard to correlate heterogeneous alarms coming from different devices.

- lacking the full picture (root cause, affected entities, etc.).

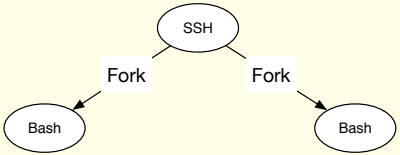
Significant manual effort and expertise are needed to piece together numerous alarms emitted by multiple security tools.

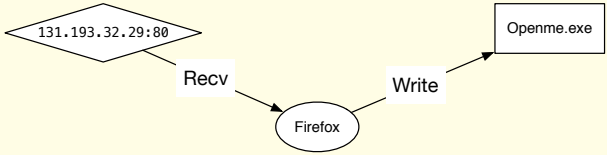
Goals

- Automating Detection, Forensics and Threat-hunting
  - Analysts should not have to manually search for interesting events
- Scalability and performance
  - Organize millions to billions of events from hosts
- High-level Scenario Reconstruction
  - not only assist with detection but also produce a compact summary of the causal chains that summarize an attack.
  - enable an analyst to quickly: ascertain whether there is an intrusion, understand root cause, and determine impact
- Cyber threat-hunting
  - When threat intelligence becomes available, analyze whether threat occurred

Provenance Graphs

- A directed Acyclic Graph (DAG) where:
  - vertices: system entities (socket, process, file, memory, etc.), and agents (user, groups, etc.)
  - edges: relationships (causal dependencies or information flow)





- Provenance graph enables defenders to:
  - leverage the full historical context of a system
  - reason about interrelationships between different events and objects

Provenance Graphs Limitations (thus far!)

Scalability: hundreds of millions to billions edges per day

Performance: forensic analysis might take hours or days to complete (not timely)

Dependency Explosion: might include many irrelevant events due to explosion of (false) dependencies.

As the Result: to date, provenance graphs have not met with widespread adoption.

- Loss of valuable source of information about system and attack activity

Challenge: To make effective and efficient use of provenance graphs for real-time intrusion detection & attack scenario reconstruction and threat-hunting.

Main results

Hossain MN, **Milajerdi SM**, Wang J, Eshete B, Gjomemo R, Sekar R, Stoller S, Venkatakrishnan VN. "***Sleuth***: Real-time Attack Scenario Reconstruction from COTS Audit Data". Proceedings of the 26th USENIX Security Symposium, 2017.

Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN. "***Holmes***: Real-time APT Detection through Correlation of Suspicious Information Flows". To appear in the Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE, May 2019.

Milajerdi SM, Eshete B, Gjomemo R, Venkatakrishnan VN. "***Poirot***: Aligning Attack Behavior with System Audit Records for Intrusion Detection and Forensics". ACM Conference on Computer and Communications Security (CCS), 2019.

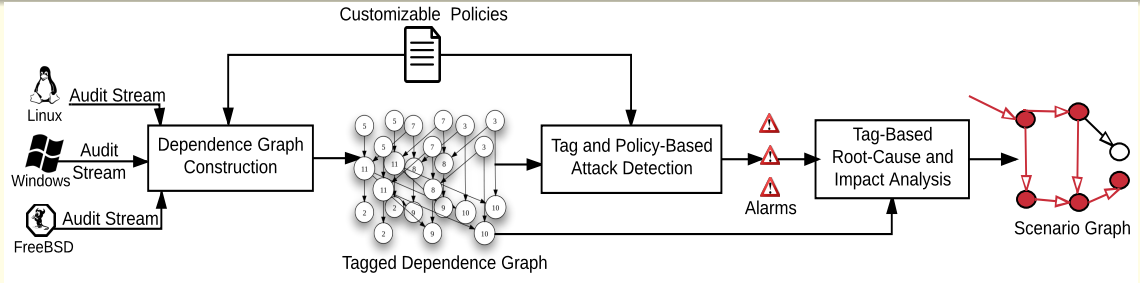
This talk = Primarily a summary of these results and our experience in participating in DARPA TC and CHASE Programs

"SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data".

Hossain MN, **Milajerdi SM**, Wang J, Eshete B, Gjomemo R, Sekar R, Stoller S, Venkatakrishnan VN.

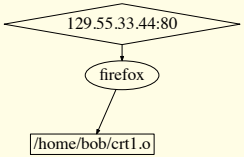
Proceedings of the 26th USENIX Security Symposium. USENIX Security, 2017.

Sleuth Architecture and Contributions



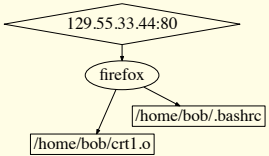
- Space-efficient in-memory dependence graph representation
- Effective attack detection based on trustworthiness and confidentiality tags
- Customizable policy framework for tag assignment and propagation
- Highly effective and efficient tag-based backward and impact analysis
- Experimental evaluation: fast, accurate and compact visual representation of APT campaigns

Illustrative Example



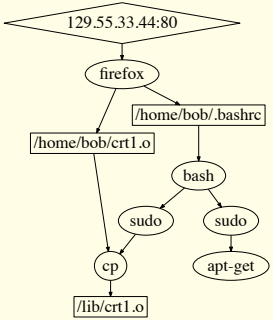
- **Attacker goal:** Insert backdoor into a vendor's software
- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`

Illustrative Example



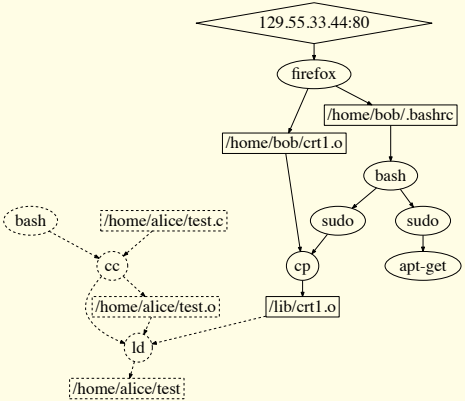
- **Attacker goal:** Insert backdoor into a vendor's software
- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob's `.bashrc` to redefine `sudo`

Illustrative Example



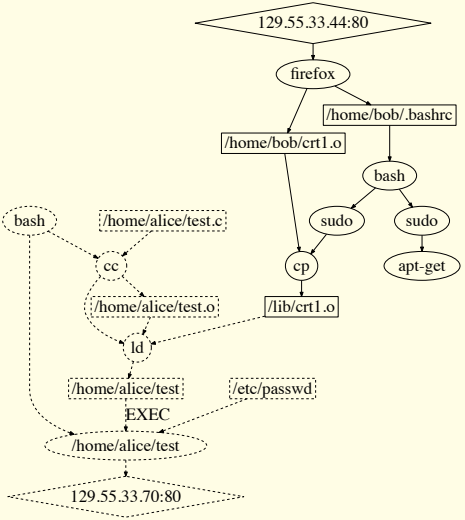
- **Attacker goal:** Insert backdoor into a vendor's software
- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob's `.bashrc` to redefine `sudo`
  3. Next time Bob uses sudo, it copies `/home/bob/crt1.o` to `/lib/crt1.o`

Illustrative Example



- **Attacker goal:** Insert backdoor into a vendor's software
- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob's `.bashrc` to redefine `sudo`
  3. Next time Bob uses sudo, it copies `/home/bob/crt1.o` to `/lib/crt1.o`
  4. When Alice builds her software, malicious `crt1.o` code is included in her executable.

Illustrative Example



- **Attacker goal:** Insert backdoor into a vendor's software
- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob's `.bashrc` to redefine `sudo`
  3. Next time Bob uses sudo, it copies `/home/bob/crt1.o` to `/lib/crt1.o`
  4. When Alice builds her software, malicious `crt1.o` code is included in her executable.
  5. When this software is run, it exfiltrates sensitive data (password file)

Provenance Tags

Trustworthiness (t-tag)

Benign authentic: Data from strongly *authenticated* sources believed to be *benign*.

Benign: Believed to be benign, but sources not well-authenticated.

Unknown: No good basis to trust this source.

Provenance Tags

Trustworthiness (t-tag)

Benign authentic: Data from strongly *authenticated* sources believed to be *benign*.

Benign: Believed to be benign, but sources not well-authenticated.

Unknown: No good basis to trust this source.

Code Vs Data Trustworthiness

- Processes have two t-tags: *code t-tag* and *data t-tag*
- Separation (a) aids detection and (b) speeds analysis by focusing on fewer root causes

Provenance Tags

Trustworthiness (t-tag)

Benign authentic: Data from strongly *authenticated* sources believed to be *benign*.

Benign: Believed to be benign, but sources not well-authenticated.

Unknown: No good basis to trust this source.

Code Vs Data Trustworthiness

- Processes have two t-tags: *code t-tag* and *data t-tag*
- Separation (a) aids detection and (b) speeds analysis by focusing on fewer root causes

Confidentiality (c-tag)

Secret: Highly sensitive, e.g., `/etc/shadow`

Sensitive: Disclosure has security impact, but less than disclosed secrets.

Private: Loss may not pose a direct security threat.

Public: Widely available, e.g., on public web sites

Flexible Policy Framework

- *Tag assignment* and *propagation* can be customized using policies.
  - Policies invoked at trigger points:
    - object creation, removal, read, write, load, execute, chmod, and chown
  - Can refer to subject, object and event attributes
- *Tag initialization* example:

  $init(o)$: $match(o.name, \texttt{"\^{}(file|IP:(10\textbackslash.0|127))"}) \rightarrow o.ttag = \text{BENIGN\_AUTH}$

  $init(o) : match(o.name, \texttt{"\^{}IP:"}) \rightarrow o.ttag = \text{UNKNOWN}$
- *Tag propagation*:
  - Default is to propagate tags from input to output
  - Custom policies created to capture exceptions, e.g., upgrade tag after a hash/signature verification.

Attack Detection Using Provenance Tags

Approach: Focus on *motive* and *means*

Motive: Does an act advance attacker's high-level objectives?

- Deploy and run attacker code
- Replace/modify important files, e.g., `/etc/passwd`, ssh keys, ...
- Steal and exfiltrate sensitive data

Means: Can the attacker control the action?

- Is the process performing the action trustworthy?

Attack Detection Using Provenance Tags

Approach: Focus on *motive* and *means*

Motive: Does an act advance attacker's high-level objectives?

- Deploy and run attacker code
- Replace/modify important files, e.g., `/etc/passwd`, ssh keys, ...
- Steal and exfiltrate sensitive data

Means: Can the attacker control the action?

- Is the process performing the action trustworthy?

Benefits

- Application-independent
- No need for training
- Resists attacker manipulation (assuming provenance isn't compromised)

Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.

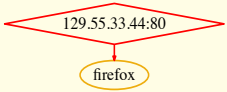
Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.



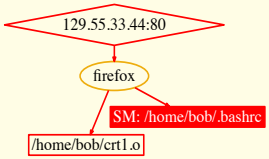
Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.



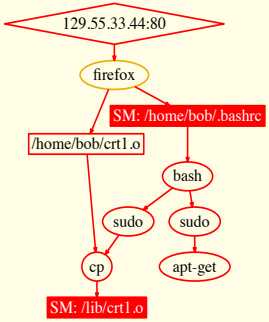
Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.



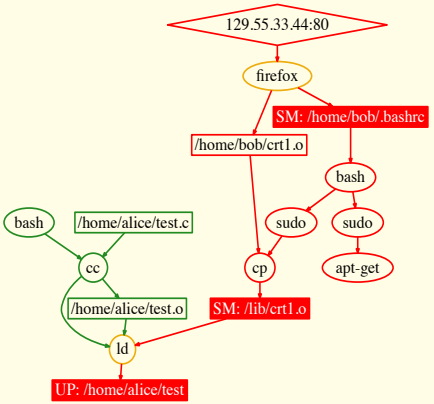
Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.



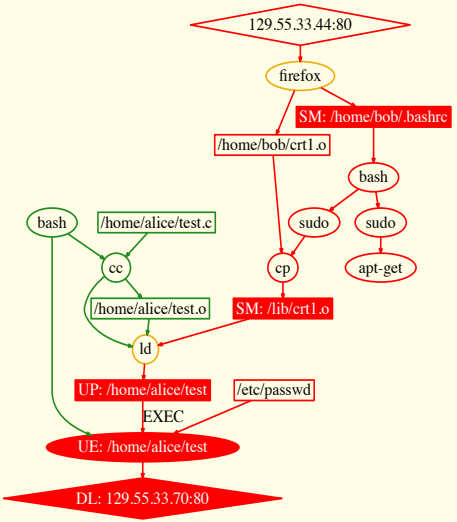
Attack Detection Policies

Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.



Adversarial Engagement Overview

Evaluation results rely on Red-Team/Blue-Team adverserial engagements organized by DARPA Transparent Computing program.

Data Set	length (hours)	# of Events	Untrusted Execution	Suspicious Modification	Execution Preparation	Data Leak
Win-1	06:22	0.1M	3	3	0	11
Win-2	19:43	0.4M	2	108	0	18
Lin-1	07:59	2.7M	2	5	1	6
Lin-2	79:06	39M	5	1	8	159
Lin-3	79:05	19M	5	2	0	5300

Backward Analysis

Goal: Identify entry point of an attack.

- Entry point is a *source*, i.e., vertex with in-degree zero.

Starting points: Suspect vertices marked by attack detectors.

Problem: Find source vertices from which a suspect vertex is reachable.

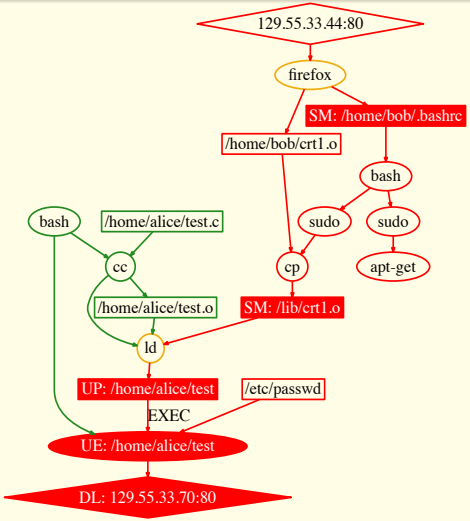
Complications:

Multiple sources: Suspect vertex is reachable from multiple sources.

Multiple suspect nodes: Typically, many detectors go off during attacks, and numerous vertices end up looking suspicious.

Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
  - Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
  - and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra's shortest path algorithm



Forward Analysis

Goal: Identify attack impact, in terms of all objects/subjects affected by the attack.

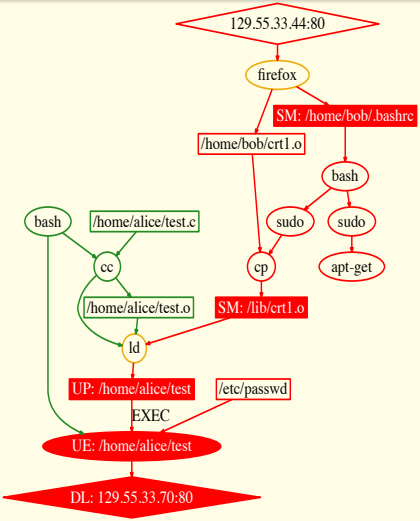
- Generate a subgraph of provenance graph that only includes objects and subjects affected by the attack.

Starting point: Sources identified by backward analysis

Challenge: Straight-forward dependence analysis may yield a graph with hundreds of thousands (if not millions) of edges.

Forward Analysis: Key Ideas

- Use cost metric to prune off distant nodes, i.e., nodes at a distance $\geq d_{th}$
- Cost metric favors
  - edges with untrusted *code* trustworthiness (cost=0);
  - and, to a lesser extent, edges with untrusted *data* trustworthiness (cost=1)



Graph Visual Simplification

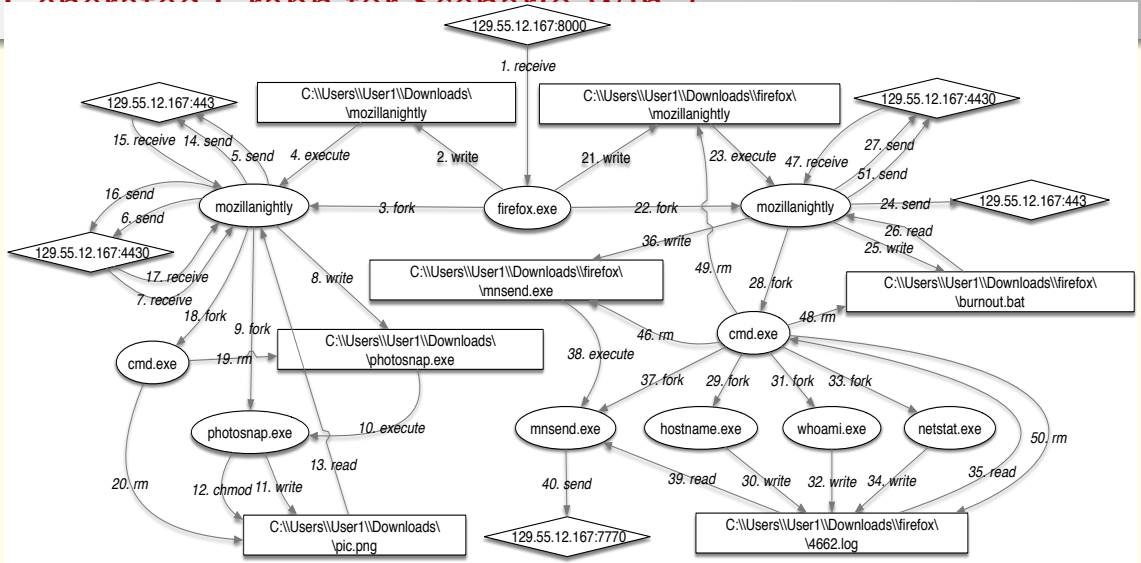
S1. Pruning uninteresting nodes: nodes that do not appear at least once in both forward and backward DFS.

S2. Removing local objects: objects isolated from the rest of the graph (e.g., pipes, temporary files).

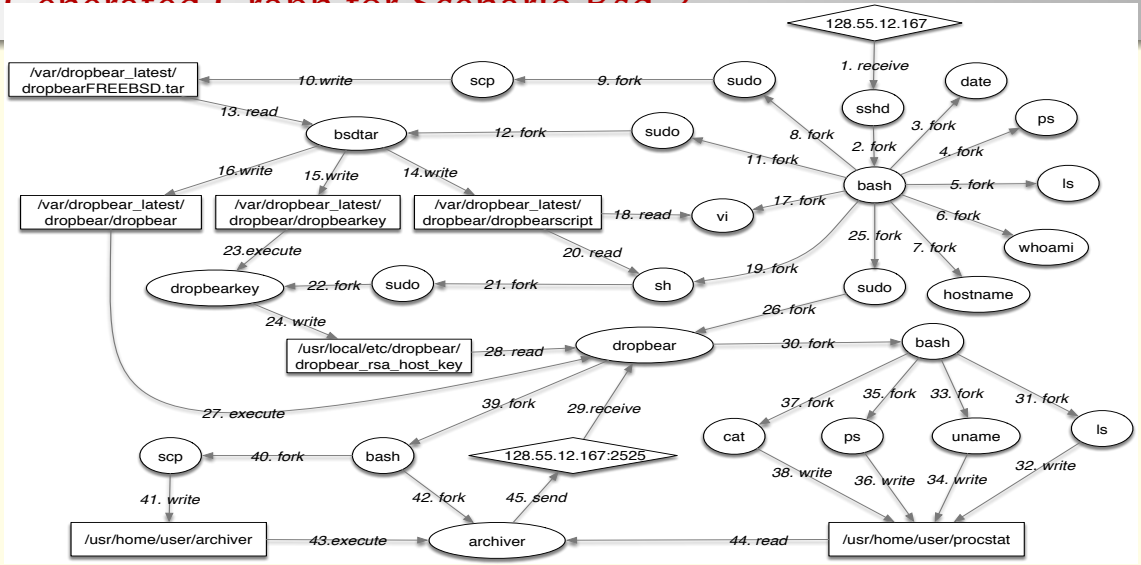
S3. Merging entities with same names: E.g., 100 instances of firefox with different pids and version numbers. Replace with one node while maintaining relevant incoming and outgoing edges.

S4. Repeated Event filtering: E.g., multiple writes to/reads from same entity (e.g., file, socket).

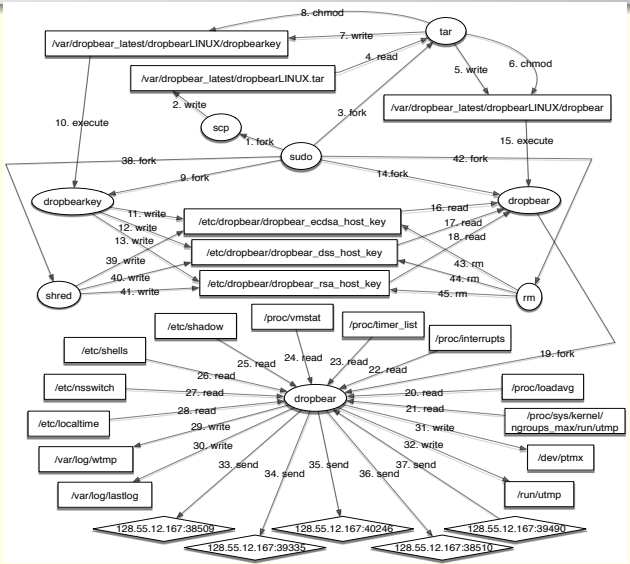
Generated Graph for Scenario \Vin 1



Generated Graph for Scenario Bad 2



Generated Graph for Scenario Lin-2



Campaign Reconstruction Summary

Campaign	Entry Points	Programs Executed	Key Files Involved	Exit Points	Correctly Identified	False Positives	Missed Entities
Win-1	2	8	7	3	20	0	0
Win-2	2	8	4	4	18	0	0
Lin-1	2	10	6	2	20	0	0
Lin-2	2	20	11	4	37	0	0
Lin-3	1	6	6	5	18	0	0
Bsd-1	4	13	9	2	13	15	1
Bsd-2	2	10	7	3	22	0	0
Bsd-3	4	14	7	1	26	0	0
Total	**19**	**89**	**57**	**24**	**174**	**15**	**1**

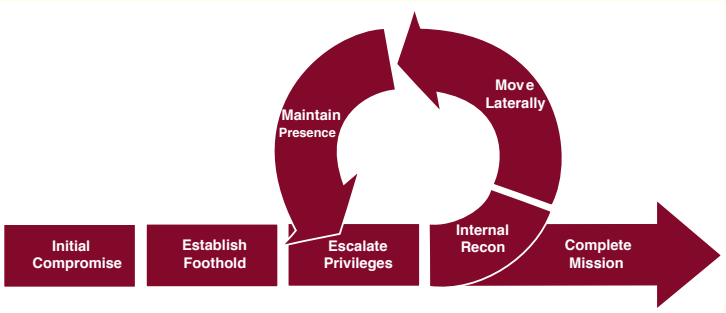
"*Holmes*: Real-time APT Detection through Correlation of Suspicious Information Flows".

Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN.

Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE, May 2019.

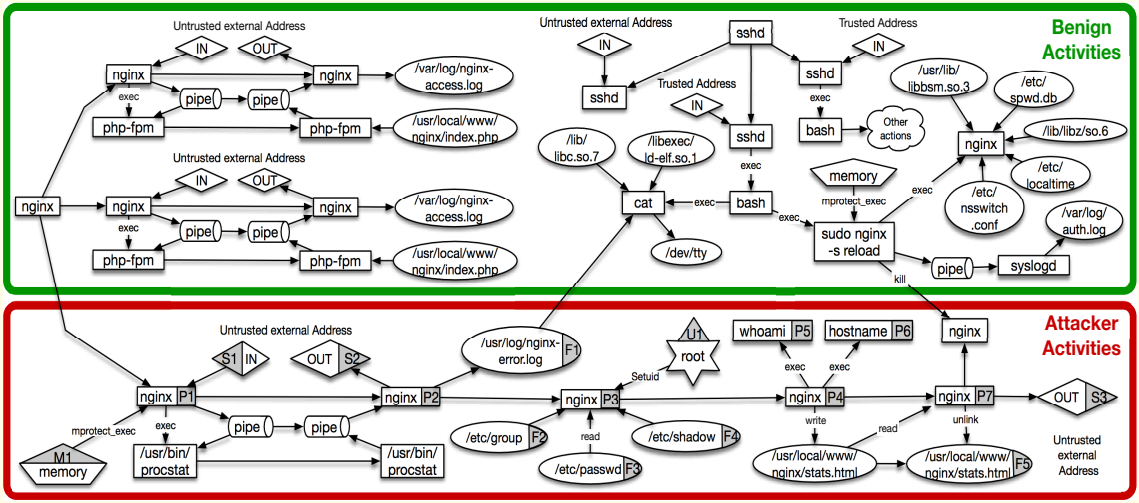
Approach Intuition

- APT behaviors often conform to the kill-chain

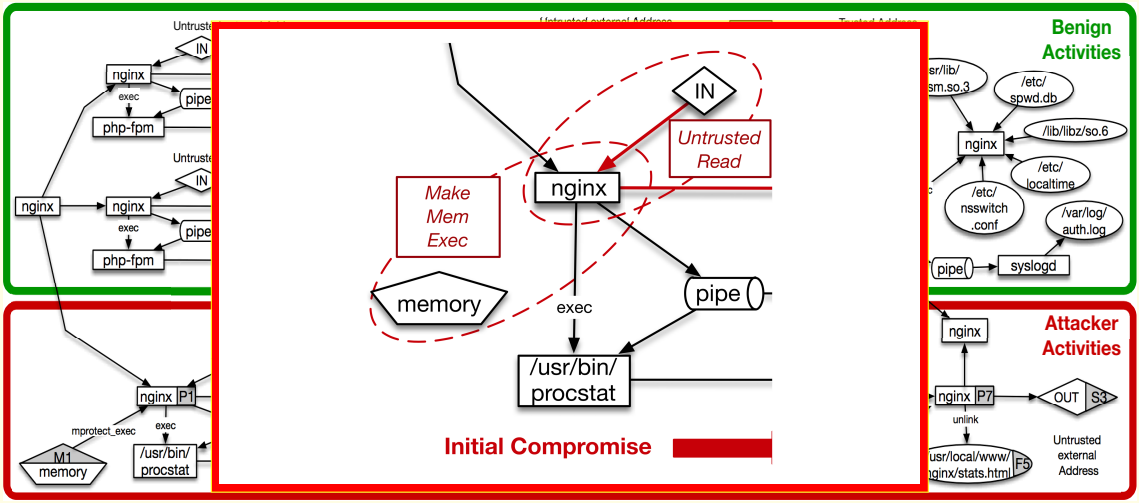


  - Our analysis of over 300 APT whitepapers confirms that most APTs follow this kill-chain
- In particular, high-level steps of APTs need to be causally connected
  - Use connectedness of high-level steps as a basis for campaign detection

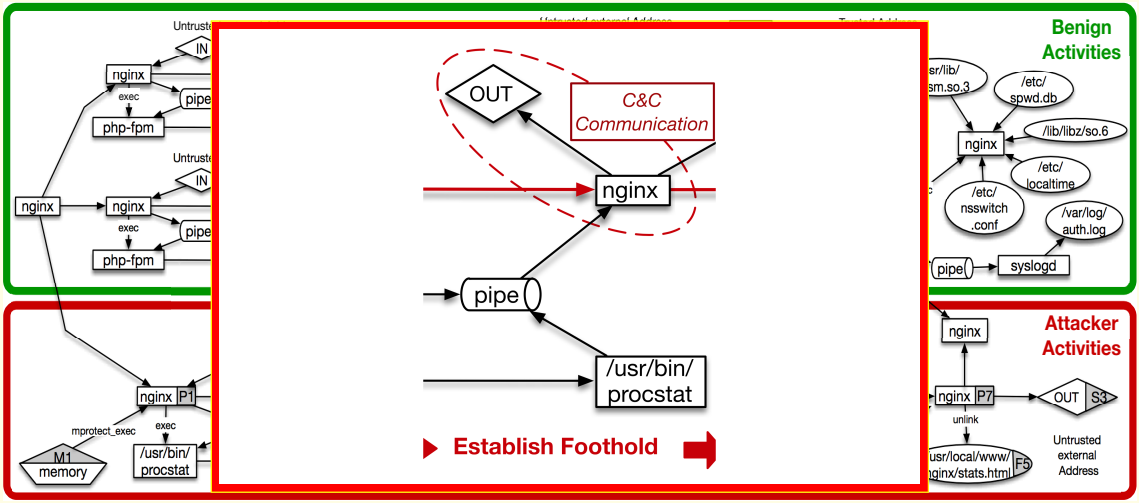
Illustrative Example



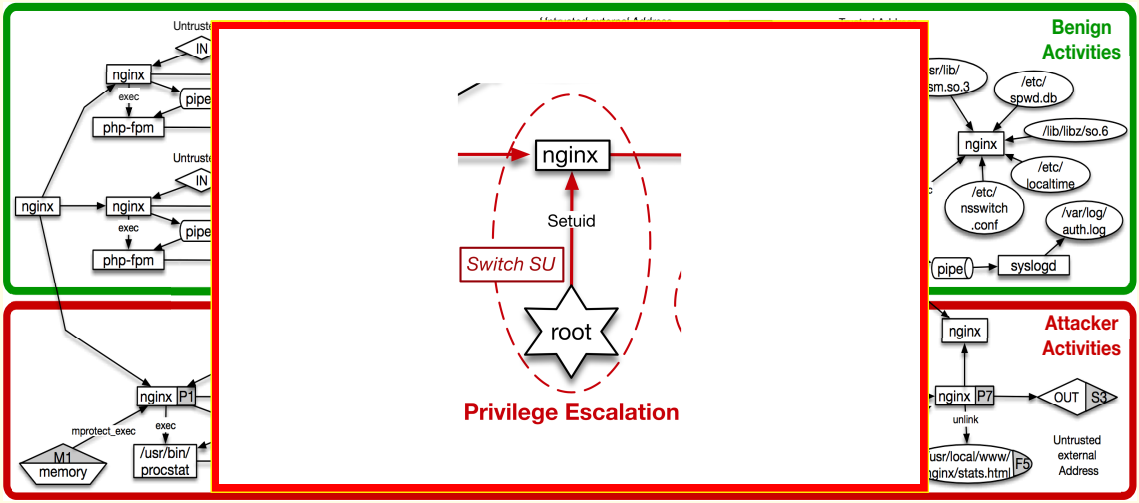
Illustrative Example



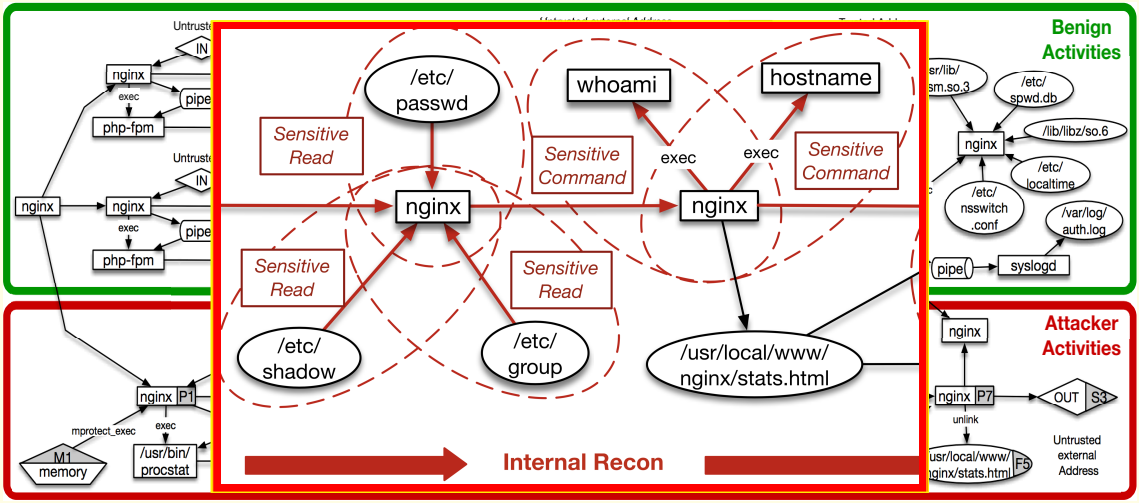
Illustrative Example



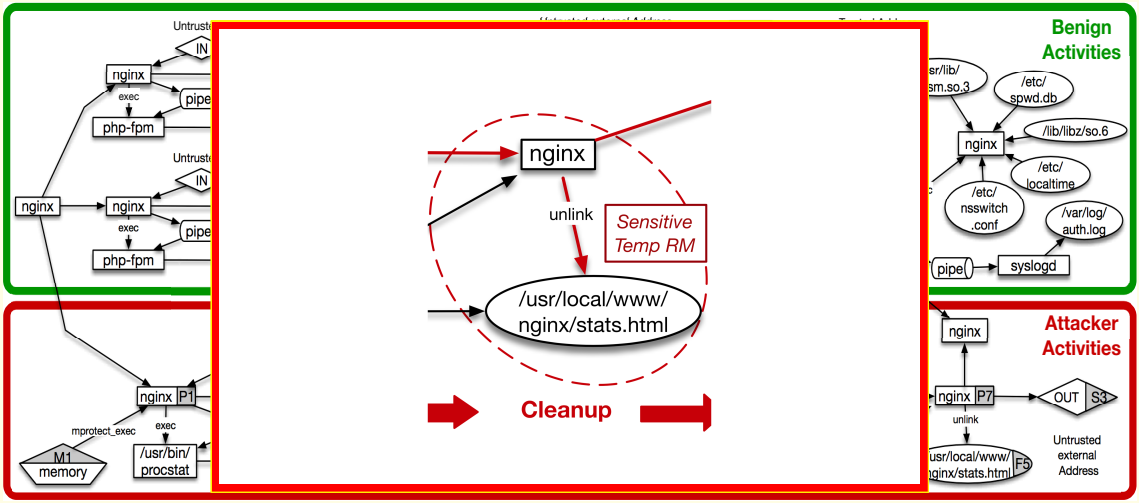
Illustrative Example



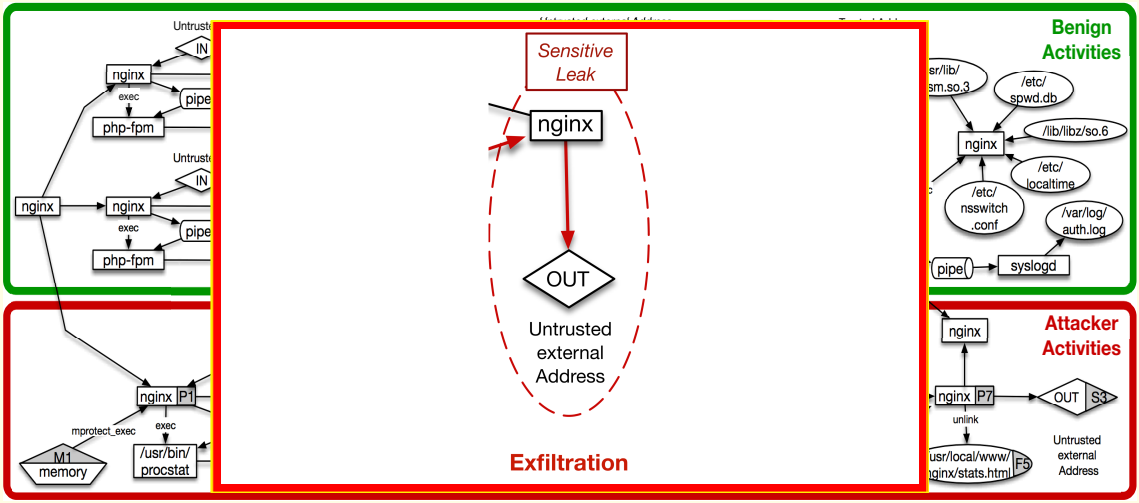
Illustrative Example



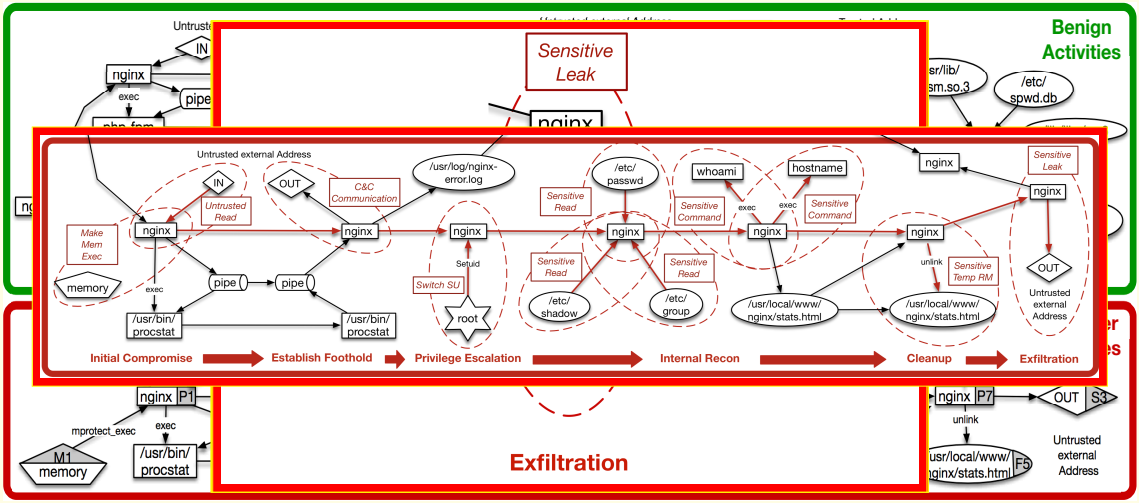
Illustrative Example



Illustrative Example



Illustrative Example



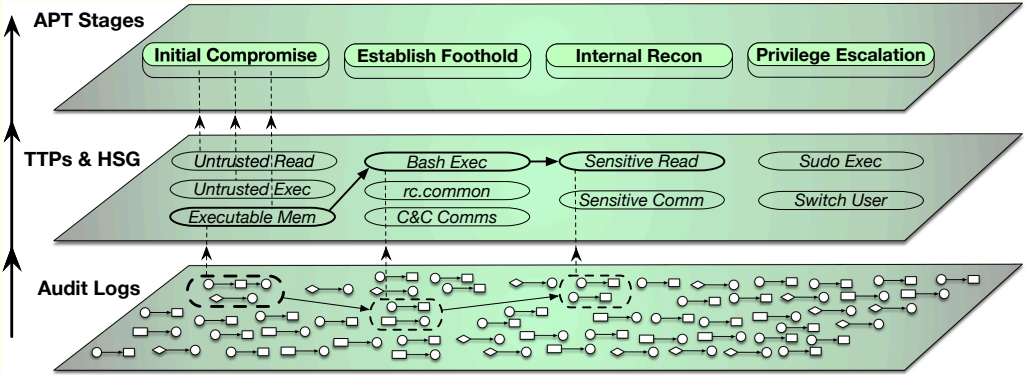
Goals

- Use alert correlation for attack campaign detection
  - Key challenge: How to fuse (unreliable) point-alerts into a strong detection signal
- *"Real-time High level APT behavior reconstruction"*
  - Current scenario graph at syscall level too low-level for quick attack comprehension
  - Key challenge: How to bridge semantic gap between low-level records and high-level activities in killchain?

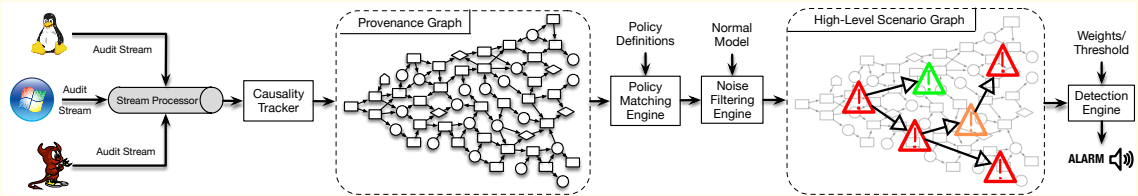
> *Main Approach:* Information-flow as a basis for alert correlation and campaign detection

Bridging the Semantic Gap

Use Tactics, Techniques, and Procedures (TTPs) from MITRE's **ATT&CK** framework as an intermediate layer to bridge low-level audit records to high-level steps



Holmes Architecture



- A Comprehensive APT Framework captures ways in which high level steps are actually executed as TTPs
- Develop TTP specifications over audit logs
- Construct high-level graph that correlates individual alerts/TTPs
- Derive campaign detection signal from graph

Example TTP specifications

APT Stage	TTP	Event Family	Events	Severity	Prerequisites
Initial_ Compromise(P)	*Untrusted_ Read*(S, P)	READ	FileIoRead (Windows), read/pread/readv/preadv (Linux,BSD)	L	$S.ip \notin$ {Trusted_IP_Addresses}
	Make_Mem_ Exec(P, M)	MPROT	VirtualAlloc (Windows), mprotect (Linux,BSD)	M	\$PROT_EXEC\$ $\in M.flags$ $\wedge \exists$ *Untrusted_Read*$(?, P')$: *path_factor*$(P', P) <=$ *path_thres*
Establish_ Foothold(P)	*Shell_ Exec*(F, P)	EXEC	ProcessStart (Windows), execve/fexecve (Linux,BSD)	M	$F.path \in$ {Command_Line_Utilities} $\wedge \exists$ *Initial_Compromise*(P') : *path_factor*$(P', P) <=$ *path_thres*

Severities: L=Low, M=Moderate, H=High, C=Critical

Entity types: P=Process, F=File, S=Socket, M=Memory, U=User.

- Domain-specific language and compiler for TTP specifications

Prerequisite-Consequence patterns

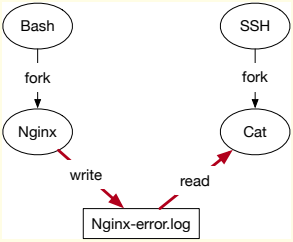
- Conditions that must be met for any TTP to be considered a match
- May include information that is derivable from event arguments (file names, permissions).
- Correlation modeling
  - Prerequisites may contain conditions related to previously matched TTPs
  - E.g. we demand an initial compromise step before an exfiltration step
- Causally connect two different activities of a campaign
- Also, help reduce false alerts as part of campaign

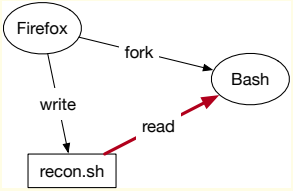
Avoiding spurious dependencies

- Spurious dependencies can result in dependence explosion
- Addressed by asking a key question: *what is the influence that attacker had in creating a dependency?*
- Key notion *Ancestor cover for f*: set of all processes that influence a dependency f.

$$\forall p \in f \; \exists a \in AC(f) \quad a = p \text{ or } a \text{ is an ancestor of } p$$

- *Minimal Ancestor cover for f - corresponds to the minimum number of processes attacker should exploit to influence a dependency f.*





Avoiding spurious dependencies (Cont.)

$$path\_factor(N_1, N_2) = \min_{\forall f : f.src = N_1, f.dst = N_2} AC_{min}(f)$$

- *path_factor* value computed incrementally in real-time

APT Stage	TTP	Event Family	Severity	Prerequisites
Complete_Mission(P)	*Sensitive_Leak*(P, S)	SEND	H	$S.ip \notin$ {Trusted_IP_Addresses} $\wedge \exists$ *Internal_Reconnaissance*(P') : *path_factor*$(P', P) <=$ *path_thres* $\wedge \exists$ *Initial_Compromise*(P'') : *path_factor*$(P'', P) <=$ *path_thres*

Value of *path_thres* could be set based on the threat an organization is preventing from

- we assume attacker is not willing or capable to compromise more than 3 exploits.

Noise reduction techniques

- Long-living processes (browsers, web servers, SSH daemons) trigger TTP matches from time to time

Idea: to exclude benign TTP from HSG to reduce noise

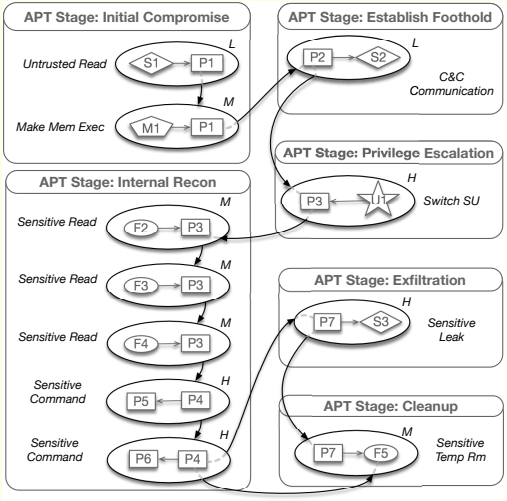
1. Ignore TTP when its prerequisites match the prerequisites encountered during training.
2. Consider the data quantities of benign information flow, measured in bytes transferred

Example: the amount of information that can flow from the file /etc/passwd to nginx is equal to the size of that file, since nginx reads that file only once.

- if significantly more bytes are observed flowing from /etc/passwd to nginx, then this flow may be part of an attack.

Signal Correlation, HSG, and Threat Tuples

- A TTP is matched and added to the HSG if all its prerequisites are satisfied.
- HSG $\rightarrow$ Threat Tuple: represents various stages of an APT campaign.
- Each element in tuple takes on severity levels $\langle M, L, H, H, -, H, M \rangle$
- HSG provides a compact, visual summary of the campaign at any moment.
  - cyber-analyst can quickly infer the big picture of the attack (scope and magnitude)



HSG Ranking and Prioritization

Severity level transformed to a number based on NIST severity score mappings

Qualitative level	Quantitative Range	Rounded up Average Value
Low	0.1 - 3.9	2.0
Medium	4.0 - 6.9	6.0
High	7.0 - 8.9	8.0
Critical	9.0 - 10.0	10.0

Tuple transformed into numeric value as weighted product

$$\prod_{i=1}^{n}(S_i)^{w_i} \geq \tau \qquad w_i = \frac{(10+i)}{10}$$

Alert raised based on threshold learned from benign activity data

$$\langle C, M, -, H, -, H, M \rangle \rightarrow \langle 10, 6, 1, 8, 1, 8, 6 \rangle \rightarrow 1163881$$

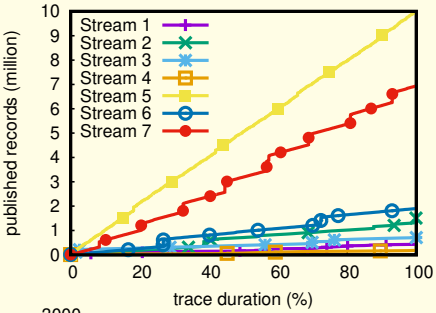
Evaluation Datasets

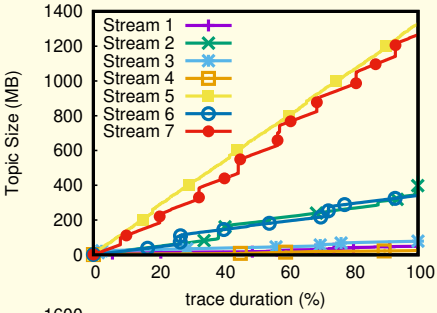
Dataset 1: Using this dataset, we measure the optimal threshold value

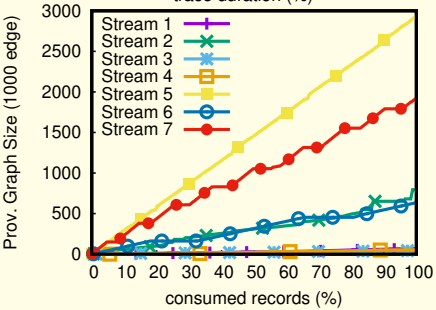
Stream No.	Duration	Platform	Scenario No.	Scenario Name	Attack Surface
1	0d1h17m	Ubuntu 14.04 (64bit)	1	Drive-by Download	Firefox 42.0
2	2d5h8m	Ubuntu 12.04 (64bit)	2	Trojan	Firefox 20.0
3	1d7h25m	Ubuntu 12.04 (64bit)	3	Trojan	Firefox 20.0
4	0d1h39m	Windows 7 Pro (64bit)	4	Spyware	Firefox 44.0
5	5d5h17m	Windows 7 Pro (64bit)	5.1	Eternal Blue	Vulnerable SMB
			5.2	RAT	Firefox 44.0
6	2d5h17m	FreeBSD 11.0 (64bit)	6	Web-Shell	Backdoored Nginx
7	8d7h15m	FreeBSD 11.0 (64bit)	7.1	RAT	Backdoored Nginx
			7.2	Password Hijacking	Backdoored Nginx

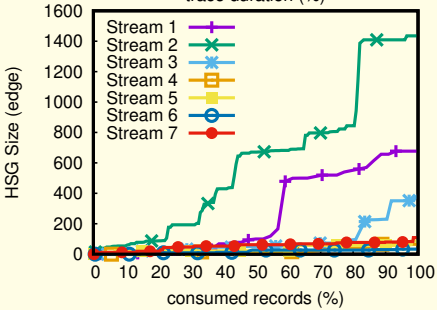
Dataset 2: live detection in a setting that we have no prior knowledge of when or how red-team is conducting the attacks.

- After this experiment, dataset has been released publicly.


published records (million)
trace duration (%)
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7


Topic Size (MB)
trace duration (%)
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7

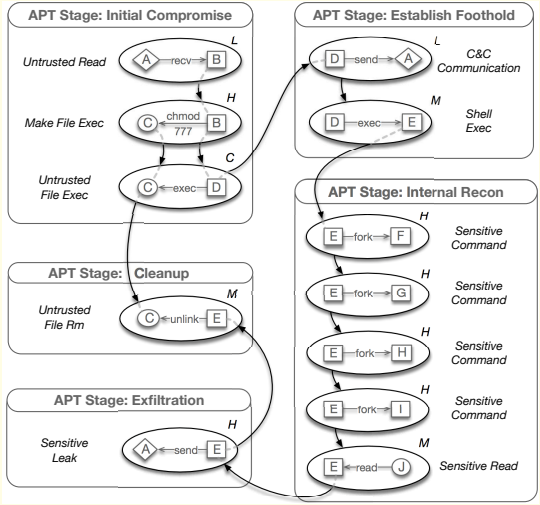

Prov. Graph Size (1000 edge)
consumed records (%)
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7


HSG Size (edge)
consumed records (%)
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7

HSG of Scenario-1 (Drive-by Download)

$$\langle C, M, -, H, -, H, M \rangle \rightarrow 1163881$$

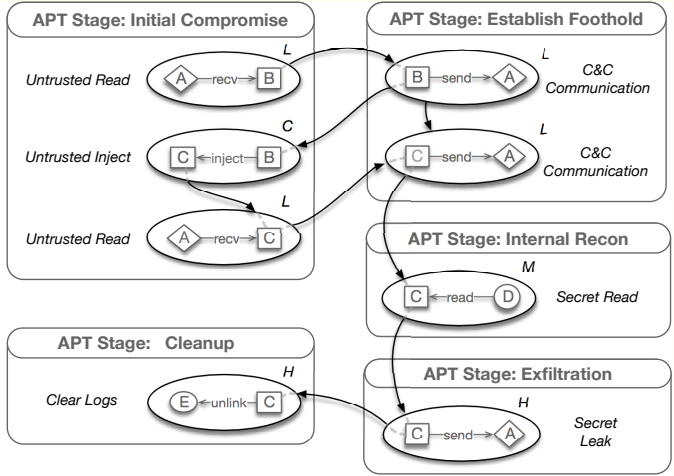
- A= Untrusted External Address
- B= Firefox
- C= Malicious dropped file (net)
- D= RAT process
- E= bash
- F= whoami
- G= uname
- I= netstat
- J= company_secret.txt;



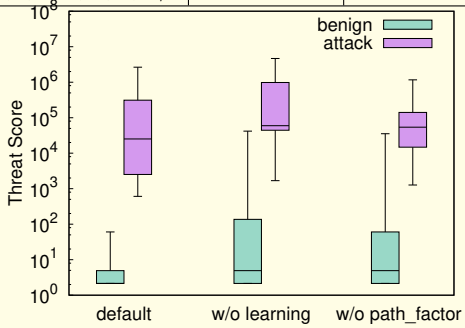
HSG of Scenario-5.1 (Eternal Blue)

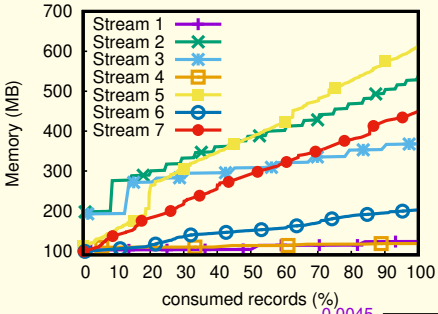
$\langle C, L, -, M, -, H, H \rangle \rightarrow 339504$

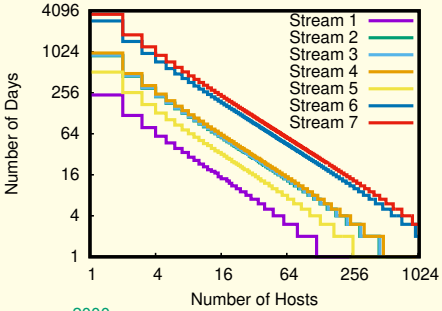
- A= Untrusted External Address
- B= lsass.exe
- C= rundll32.exe
- D= password.txt
- E= Winevt logs;

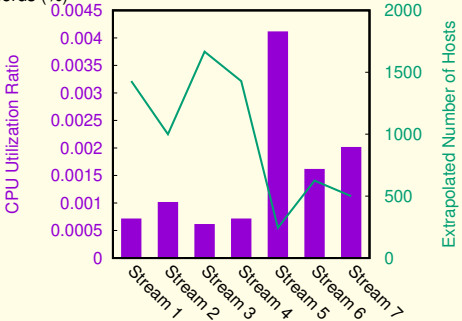


Scenario No.	Threat Tuple	Threat Score	Highest Benign Score in Dataset
1	$\langle C, M, -, H, -, H, M \rangle$	1163881	61
2	$\langle C, M, -, H, -, H, - \rangle$	55342	226
3	$\langle C, M, -, H, -, H, M \rangle$	1163881	338
4	$\langle C, M, -, H, -, -, M \rangle$	41780	5
5.1	$\langle C, L, -, M, -, H, H \rangle$	339504	104
5.2	$\langle C, L, -, -, -, -, M \rangle$	608	
6	$\langle L, L, H, M, -, H, - \rangle$	25162	137
7.1	$\langle C, L, H, H, -, H, M \rangle$	4649220	133
7.2	$\langle M, L, H, H, -, H, M \rangle$	2650614	

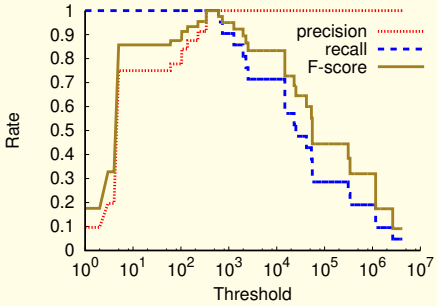


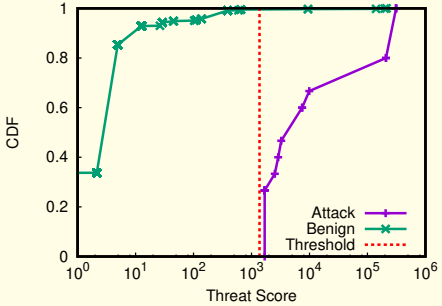

Memory (MB)
consumed records (%)
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7


Number of Days
Number of Hosts
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7


CPU Utilization Ratio
Extrapolated Number of Hosts
Stream 1
Stream 2
Stream 3
Stream 4
Stream 5
Stream 6
Stream 7

Optimal threshold Value and Live Experiment Results





- F-score maximum at [338.25, 608.26]: Average severity of each APT step = 2.09
- Threshold set for Live experiment: $2.09^{\sum_{i=1}^{7} w_i} = 2.09^{9.8} = 1378$
- A few false positive: system administrator connecting via SSH

Summary

- Presented a real-time APT detection system that correlates TTPs that might be used to carry out each APT stage.

visualize high-level APT behavior in real time.

- Mitigating the dependence explosion problem by using the concept of minimum ancestral cover
- Pruning benign system activities based on data quantities in the flow of information
- Experiments show high accuracy and performance for **Holmes**
- Effectiveness evaluated using a live experiment w/o having prior knowledge of attacks.

"*Poirot*: Aligning Attack Behavior with System Audit Records for Intrusion Detection and Forensics".

Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN. To appear in the Proceedings of the ACM Computer and Communications Security (CCS), Nov 2019.

Cyber Threat Hunting

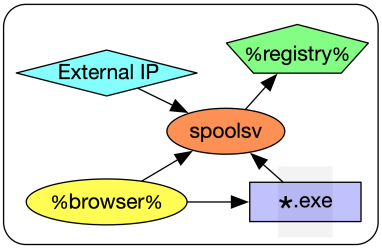
- Cybersecurity labs discover new threats
- The details released as Threat Intelligence Reports
  - Structured: OpenIOC, STIX, MISP
  - Unstructured: Natural Language Description

Common question: Has my enterprise been target of this threat?

- State of the art: Looking for fragmented Indicators of Compromise (IOC)
  - hash values, file/process names, IP addresses, domain names
- Limited against:
  - Use of legitimate-looking names (like svchost in Windows)
  - Updated or re-purposed attacks
  - Malware polymorphism

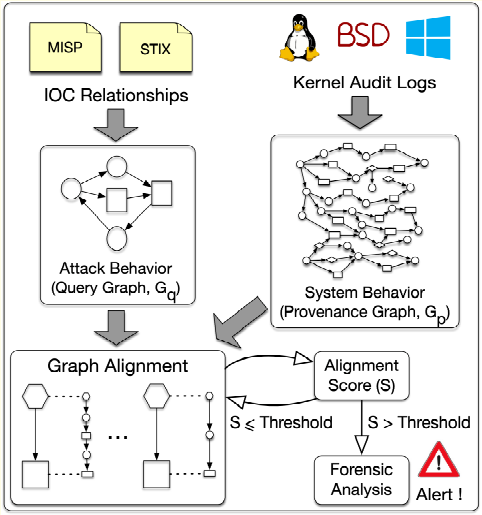
Intuition and Challenges

- Relationships between artifacts and single IOCs contain essential clues on the behavior of the attacks
  - Tied to attacker goals (More difficult to change)
- Challenges:
  - Search at scale
    - Link delayed actions over weeks/months
    - Cannot rely on timestamp for correlation
  - Efficient Matching (timely)
    - Security Analysts can't react to many false positives



Approach

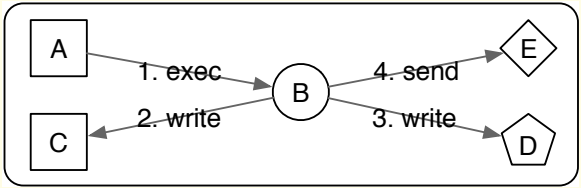
- Modeling Threat Hunting as finding an embedding of a query graph in a much larger provenance graph
  - G_p: kernel audit logs modeled as a graph
  - G_q: attack behavior modeled as a graph
  - Both labeled, typed, and directed
  - Inexact matching



Query Graphs

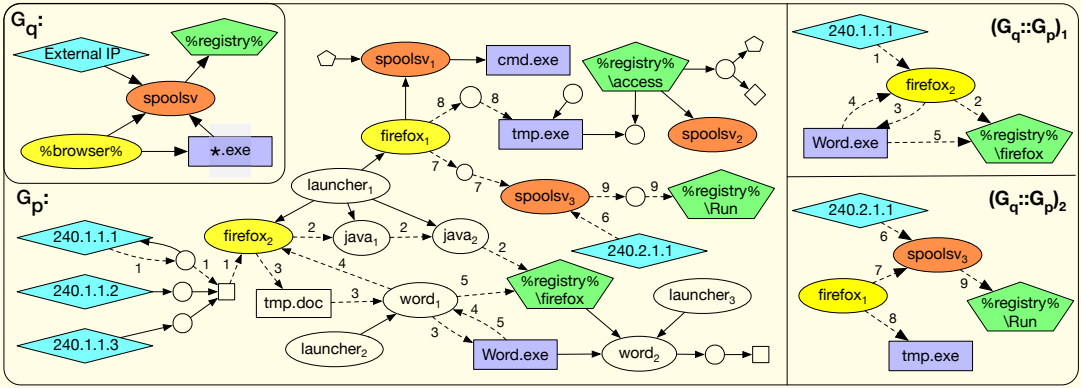
- Built by automated tools or human operators
- Using more general terms to increase coverage (∗)
- Not intended to be a precise subgraph
  - Summary of the actual attack graph

Upon execution, 8aba4b5184072f2a50cbc5ecfe326701 writes "28542CC0.dll" to this location: "C:\Documents and Settings\All Users\Application Data\28542CC0.dll". In order to maintain persistence, the original malware adds this registry key: "%HKCU%\Software\Microsoft\Windows\CurrentVersion\Run\ 28542CC0". The malware then connects to a host in South Korea (180.150.228.102).



A=∗.%exe%, B=∗, C=%APPDATA%\∗,
D=%HKCU%\Software\Microsoft\Windows\CurrentVersion\
Run\∗, E=%External IP address%

Sample Alignments



Influence Score between two nodes

- *Influence score*: likelihood that an attacker can produce a flow
- Robust against evasion method

$$\Gamma_{i,j} = \begin{cases} \max\limits_{i \dashrightarrow j} \dfrac{1}{C_{min}(i \dashrightarrow j)} & \exists i \dashrightarrow j \mid C_{min}(i \dashrightarrow j) \leq C_{thr} \\ 0 & \text{otherwise} \end{cases}$$

- $C_{min}(i \dashrightarrow j)$ represents the minimum number of distinct, independent compromises an attacker has to conduct to be able to generate the flow $i \dashrightarrow j$.

Goodness Score of a graph alignment

A metric that specifies the goodness of a graph alignment $G_q :: G_p$.

$$S(G_q :: G_p) = \frac{1}{|F(G_q)|} \sum_{(i \dashrightarrow j) \in F(G_q)} \Gamma_{k,l} \mid i : k \;\&\; j : l$$

- Nodes i and j are members of $V(G_q)$, and nodes k and l are members of $V(G_p)$
- Flow $i \dashrightarrow j$ is a flow defined over G_q
- Normalized by dividing it with the maximal value possible for that sum
- Raise an alarm when the goodness score $S(G_q :: G_p)$ bypasses a threshold value

Evaluation

- Three experiments:
  - DARPA Transparent Computing (TC) program red-team vs. blue-team adversarial engagement scenarios
    - Windows, BSD, and Linux
    - Benign background activities in parallel
  - Real-world incidents with publicly available natural language descriptions
  - Attack-free activities generated by ordinary users
    - Robustness against false signals
- Set the value of C_{thr} to 3 (=threshold of $\frac{1}{3}$)
  - Configurable (higher C_{thr} will increase the number of false positives while may detect sophisticated attacks, with multiple initial entry points)

Graph Alignment Scores for TC dataset

Scenario	Earliest iteration bypassing threshold	Earliest score bypassing threshold	Max score in 20 iterations	Earliest iteration resulting Max score
BSD-1	1	0.45	0.64	5
BSD-2	1	0.81	0.81	1
BSD-3	1	0.89	0.89	1
BSD-4	1	1	1	1
Win-1	1	0.63	0.63	1
Win-2	1	0.47	0.63	4
Linux-1	1	0.58	0.58	1
Linux-2	2	0.55	0.71	5
Linux-3	1	0.54	0.54	1
Linux-4	1	0.87	0.87	1

Evaluation on the malware reports

Malware Name	Report Source	Year	Reported Samples	Sample Relation	Isolated IOCs	Detection Results			
						RedLine	Loki	Splunk	POIROT
njRAT	Fidelis	2013	30	different	153	F+H	F+H	P	B (score=0.86)
DeputyDog	FireEye	2013	8	subset	21	F×2+H+R	F×2+H	P+R	B (score=0.71)
Uroburos	Gdata	2014	4	subset	26	F+H	F+H	R	B (score=0.76)
Carbanak	Kaspersky	2015	109	different	230	-	PE	S	B (score=0.68)
DustySky	Clearsky	2016	79	different	250	-	-	-	B (score=1.00)
OceanLotus	Eset	2018	9	subset	117	F+R	F+PE	P+R	B (score=0.65)
Hawkeye	Fortinet	2019	3	different	3	-	PE	-	B (score=0.62)

B=Behavior, PE=PE-sieve, F=File Name, H=Hash, P=Process Name, R=Registry, S=Windows Security Event.

“AI challenges in Threat Detection”.

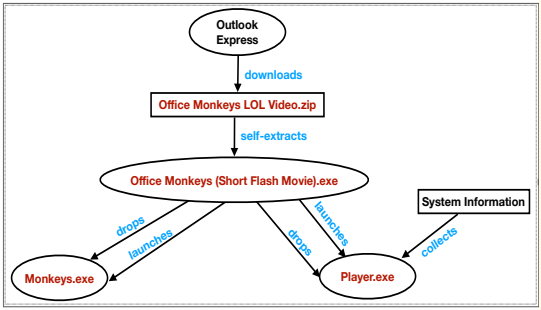
From Threat Intelligence Reports to Graph Representations

- Robust identification and linking of threat-relevant entities in NL reports

Challenge for NLP: graph-based representation of IOCs including intra/inter-source semantic links between fragmented IOC signals in addition to their connections with other system entities.

See our paper *Extractor:Extracting Attack Behavior from Threat Reports* (EURO S&P 2021)

> Email Client downloads Office Monkeys LOL Video.zip. Office Monkeys LOL Video.zip self-extracts to Office Monkeys (Short Flash Movie).exe. Monkeys (Short Flash Movie).exe drops Monkeys.exe and Player.exe under %temp%. It first launches Monkeys.exe. It then launches Player.exe. Player.exe then collects system information.



Acknowledgements

- DARPA under the TC and CHASE programs
- National Science Foundation
- Collaborators and co-authors *Md. Nahid Hossain, Sadegh M. Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R. Sekar and Scott Stoller*