



Real-time APT Detection through Correlation of Suspicious Information Flows

Sadegh M. Milajerdi*, Rigel Gjomemo*, Birhanu Eshete[†], R. Sekar[‡], V.N. Venkatakrishnan*

*University of Illinois at Chicago
{smomen2,rgjome1,venkat}@uic.edu

[†]University of Michigan-Dearborn
birhanu@umich.edu

[‡]Stony Brook University
sekar@cs.stonybrook.edu

Advanced Persistent Threat (APT) and its challenges

Targeted cyber attacks on organizations getting more sophisticated and stealthy.

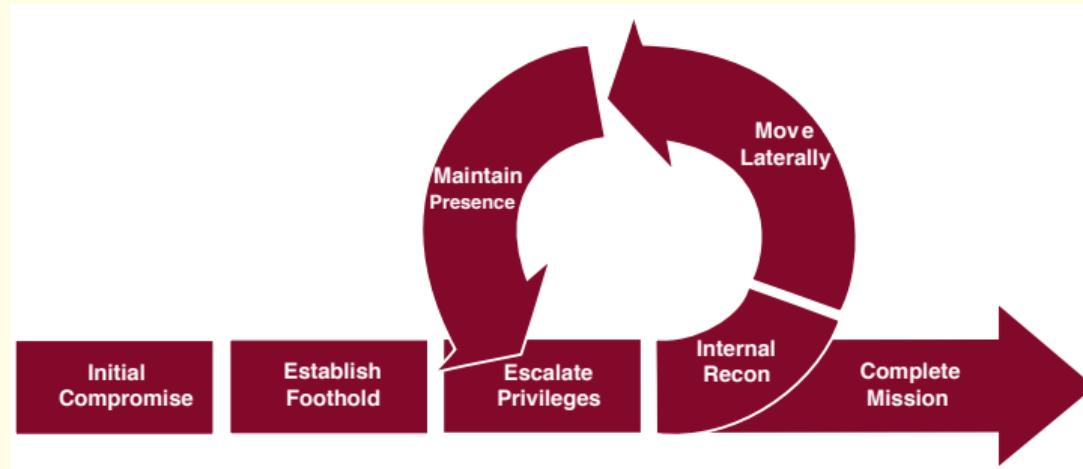
Goal: to steal data, disrupt operations or destroy infrastructure.

- APTs combine many different attack vectors
 - Each appearing in some log sources
 - Firewall, IDS/IPS, Netflow, DNS logs, Identity and access management tools
- Might occur over a long duration
 - Correlating heterogeneous alarms using heuristics like timestamp is not so effective
 - 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.

Intuition

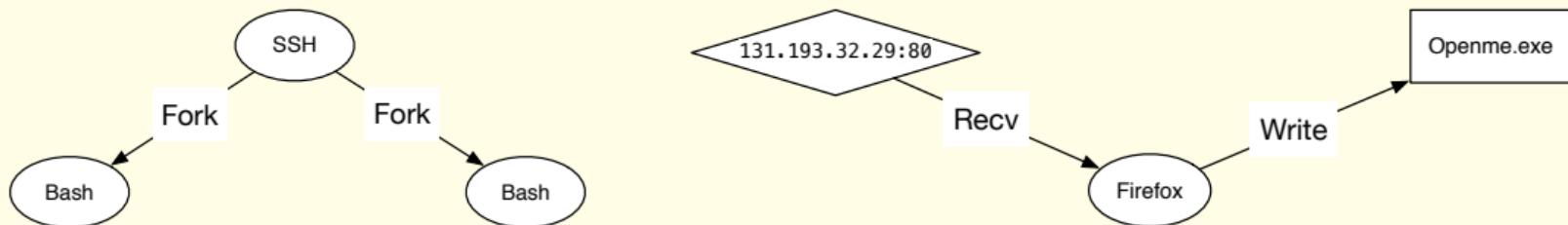
- APT behaviors often conform to the kill-chain [MANDIANT-APT1]



- 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

Approach

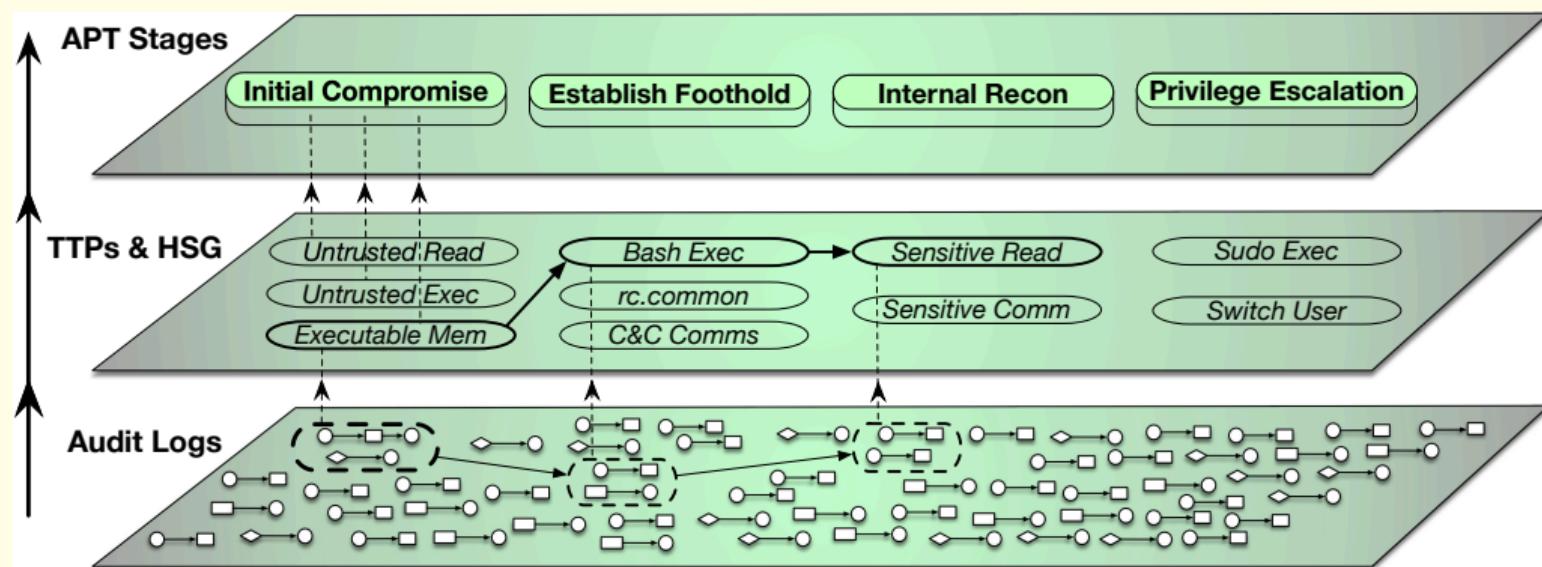
- Use Provenance Graph to enable alert correlation for attack campaign detection
 - vertices: system entities (socket, process, file, memory, etc.), and agents (user, groups, etc.)
 - edges: system calls (causal dependencies or information flow)



- Leverage the full historical context of a system
- Reason about interrelationships between different events and objects
- ***Key challenge:*** How to bridge semantic gap between low-level records and high-level activities in killchain?

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



Bridging the Semantic Gap

The diagram illustrates the semantic gap by mapping specific attack techniques from the MITRE ATT&CK matrix against various detection and exfiltration methods. Red arrows highlight several key connections:

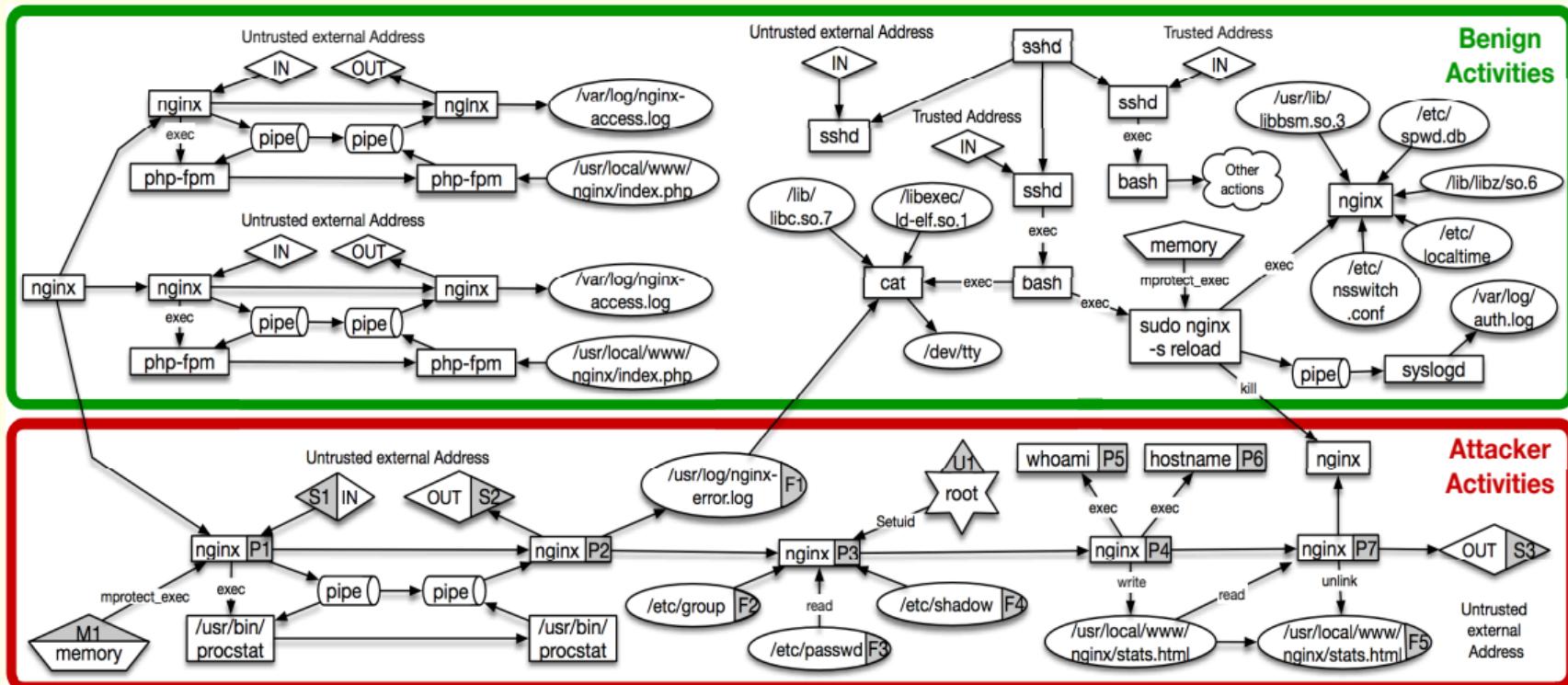
- Spearphishing Attachment** connects to **Image File Execution Options Injection**, **Process Injection**, **Remote File Copy**, and **Pass the Ticket**.
- User Execution** connects to **Valid Accounts**, **DLL Search Order Hijacking**, **Applet DLLs**, **Hooking**, **Proxy Execution**, **Signed Script**, **Keychain**, **Input Prompt**, **Process Discovery**, **System Network**, **Connections Discovery**, **Logon Scripts**, **Windows Remote Management**, and **System Owner/User Discovery**.
- Hooking** connects to **DCShadow**, **Bash History**, **Two-Factor Authentication**, **Indirect Command Execution**, **Port Knocking**, **Input Capture**, **Network Sniffing**, **Credential Dumping**, **Hidden Files and Directories**, **Kerberosasting**, **Securityd Memory**, **System Name Discovery**, and **Account Discovery**.
- Pass the Hash** connects to **Windows Admin Shares**.
- Standard Application Layer Protocol** connects to **Shared Webroot**.
- File and Directory Discovery** connects to **LLMNR/NBT-NS**.
- Replication Through Removable Media** connects to **File and Directory Discovery**.
- Clipboard Data** connects to **Replication Through Removable Media**.
- Automated Collection** connects to **Clipboard Data**.
- Exfiltration Over Network Medium** connects to **Clipboard Data**.
- Clipboard Data** connects to **File and Directory Discovery**.
- Automated Exfiltration** connects to **Clipboard Data**.
- Web Service** connects to **Clipboard Data**.
- Exfiltration Over Other Network Medium** connects to **Clipboard Data**.
- Standard Non-Application Layer Protocol** connects to **Clipboard Data**.
- Standard Application Layer Protocol** connects to **Clipboard Data**.
- Commonly Used Port** connects to **Clipboard Data**.
- Standard Cryptographic Protocol** connects to **Clipboard Data**.
- Custom Cryptographic Protocol** connects to **Clipboard Data**.
- Data Obfuscation** connects to **Clipboard Data**.
- Custom Command and Control Protocol** connects to **Clipboard Data**.

ATT&CK™ is prominently displayed at the bottom left of the matrix.

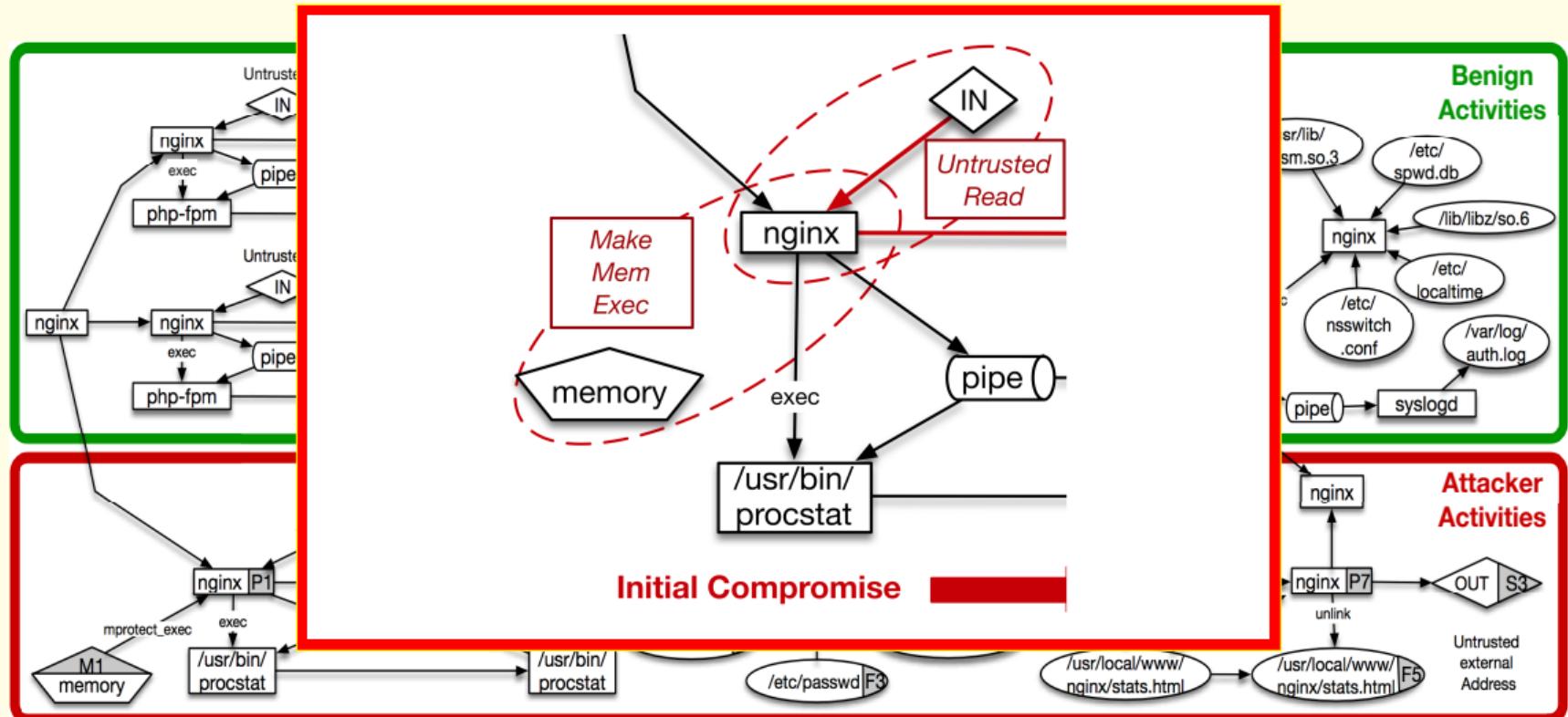
Hardware Additions	Scheduled Task		Binary Padding	Credentials in Registry	Browser Bookmark Discovery	Exploitation of Remote Services	Data from Information Repositories	Exfiltration Over Physical Medium	Remote Access Tools	
Trusted Relationship	LSASS Driver		Extra Window Memory Injection	Exploitation for Credential Access	Network Share Discovery	Distributed Component Discovery	Video Capture	Exfiltration Over Multi-hop Proxy	Port Knocking	
Supply Chain Compromise	Local Job Scheduling		Access Token Manipulation	Forced Authentication	Object Model Discovery	Audio Capture	Command and Control Channel	Domain Fronting	Multi-hop Proxy	
Spearphishing Attachment	Trap Launch		Bypass User Account Control	Hooking	Peripheral Device Discovery	Remote File Copy	Automated Collection	Data Encrypted	Data Encoding	
Exploit Public-Facing Application	Staled Binary	Proxy Execution	Image File Execution Options Injection	Password Filter DLL	File and Directory Discovery	Pass the Ticket	Clipboard Data	Remote File Copy	Multi-Stage Channels	
Replication Through Removable Media	Exploitation for Client Execution		Plist Modification	LLMNR/NBT-NS	Replication Through Removable Media	Email Collection	Automated Exfiltration	Exfiltration Over Other Network Medium	Web Service	
Spearphishing via Service	CMSTP		Valid Accounts	Poisoning	Permission Groups Discovery	Screen Capture	Exfiltration Over Network Medium	Standard Non-Application Layer Protocol	Standard Application Layer Protocol	
Spearphishing Link	Dynamic Data Exchange		DLL Search Order Hijacking	Private Keys	Windows Admin Shares	Data Staged	Alternative Protocol	Standard Application Layer Protocol	Commonly Used Port	
Drive-by Compromise	Mshta		Applet DLLs	Keychain	Pass the Hash	Input Capture	Data Transfer	Connection Proxy	Standard Cryptographic Protocol	
Valid Accounts	AppleScript		Startup Items	Input Prompt	Process Discovery	Third-party Software	Data from Network	Multi-layer Encryption	Custom Cryptographic Protocol	
	Source		DCShadow	Bash History	System Network	Shared Webroot	Shared Drive	Man in the Browser	Data Compressed	
	Space after filename		Signed Script	Two-Factor Authentication	Connections Discovery	Logon Scripts	Data from Local System	Data from Removable Media	Scheduled Transfer	Commonly Used Port
	Execution through Module Load		Keychain	Indirect Command Execution	System Owner/User Discovery	Windows Remote Management	Media			Standard Cryptographic Protocol
	Regsvcs/Regasm	New Service	AppInit DLLs	Port Knocking	Application Window	SSH Hijacking				Custom Cryptographic Protocol
	InstallUtil	File System Persistence	Web Shell	BITS Jobs	Configuration Discovery	AppleScript				Data Obfuscation
	Regsvr32		Service Registry Permissions Weakness	Control Panel Items	Input Capture	Discovery				Custom Command and Control Protocol
	Execution through API		Process Doppelgänging	CMSTP	Application Window	Protocol				
	PowerShell		Mshta	Network Sniffing	Discovery					
				Credential Dumping	Discovery					
				Hidden Files and Directories	Discovery					
				Kerberasting	Discovery					
				Securityd Memory	System Name Discovery					
				Brute Force	Account Discovery					
					Remote Services					

MITRE

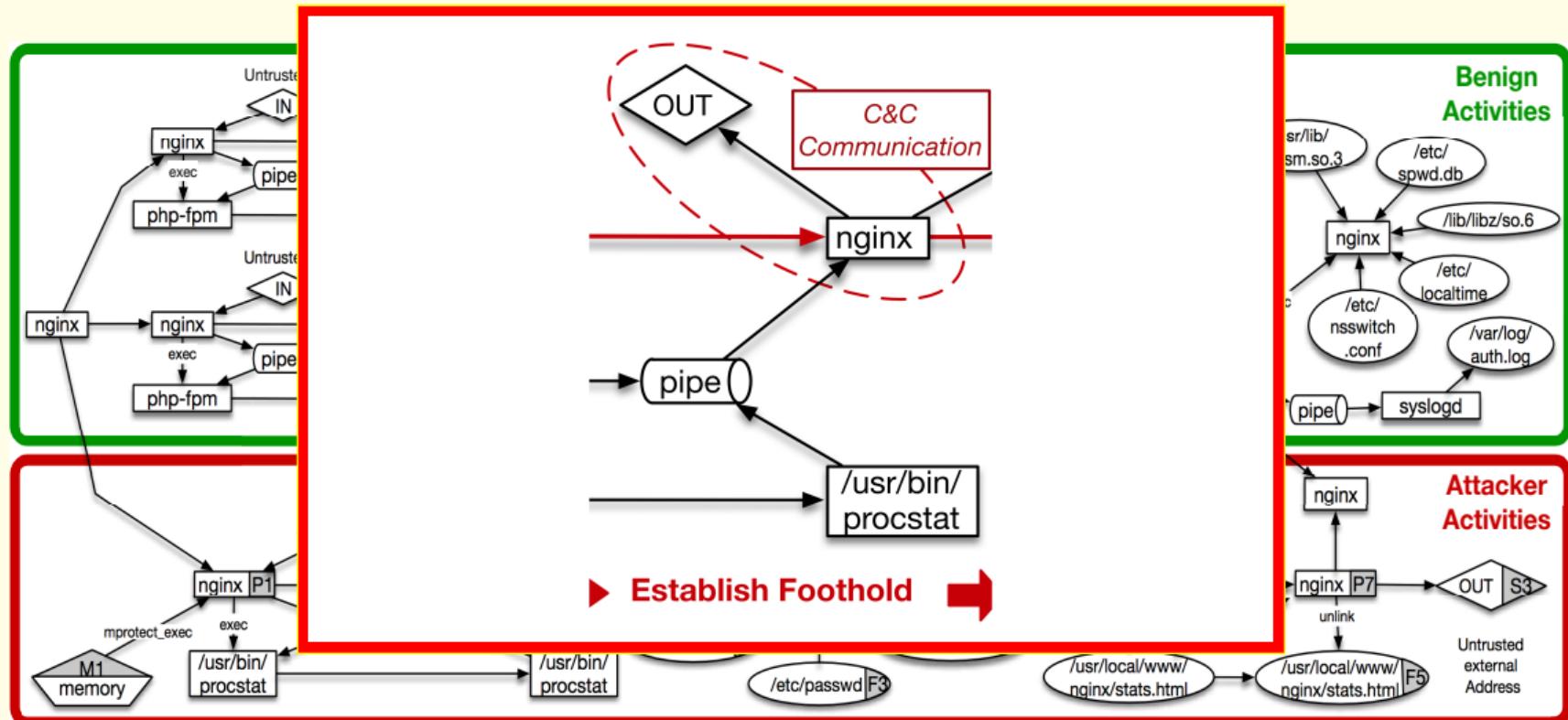
Illustrative Example



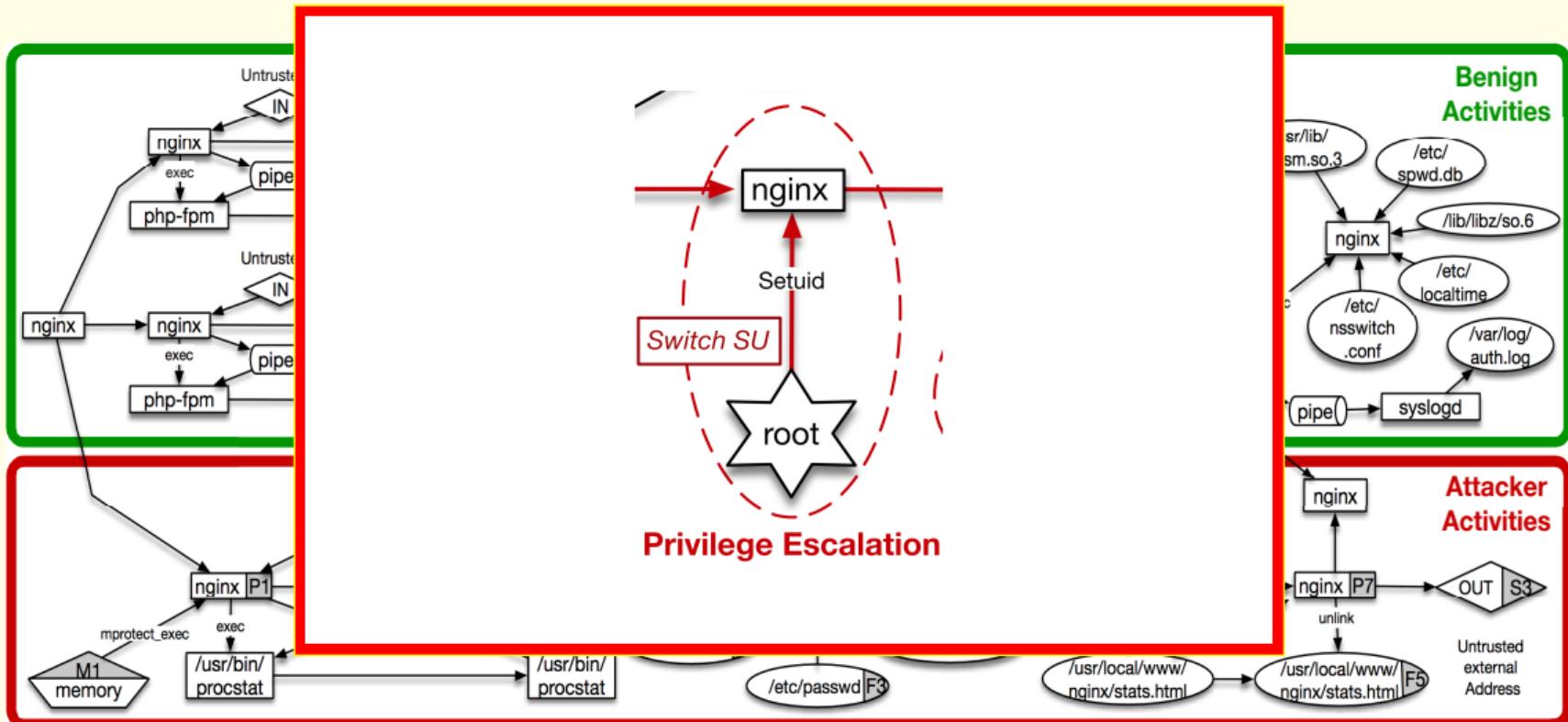
Illustrative Example



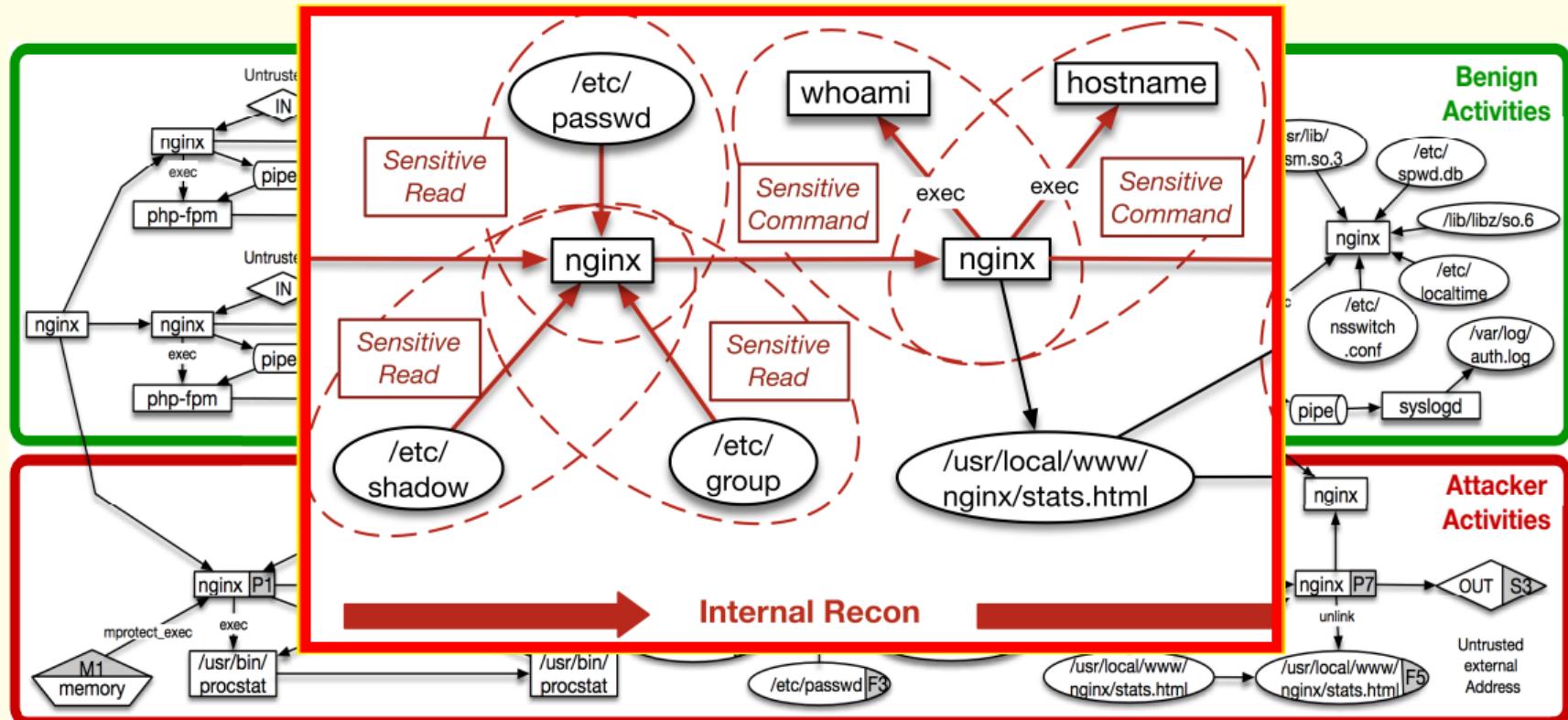
Illustrative Example



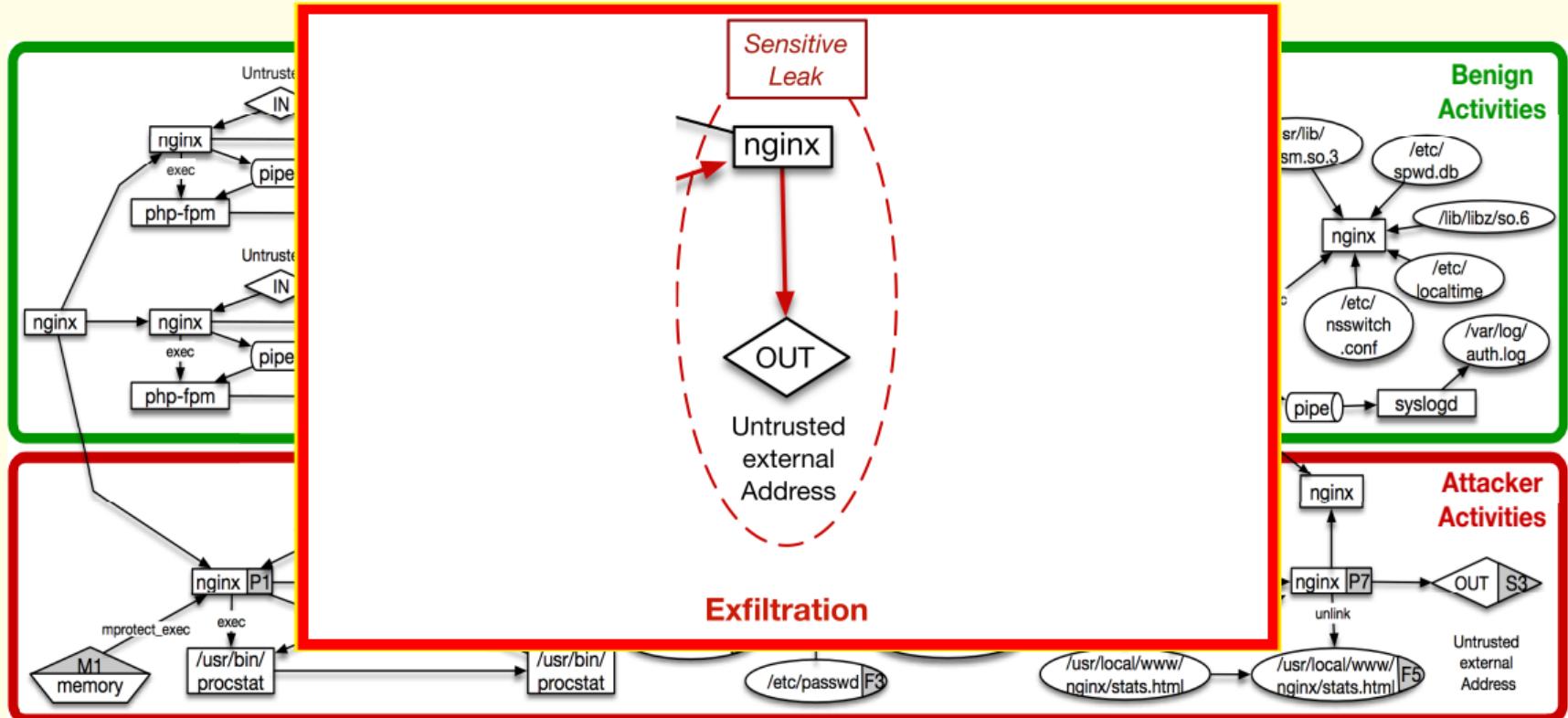
Illustrative Example



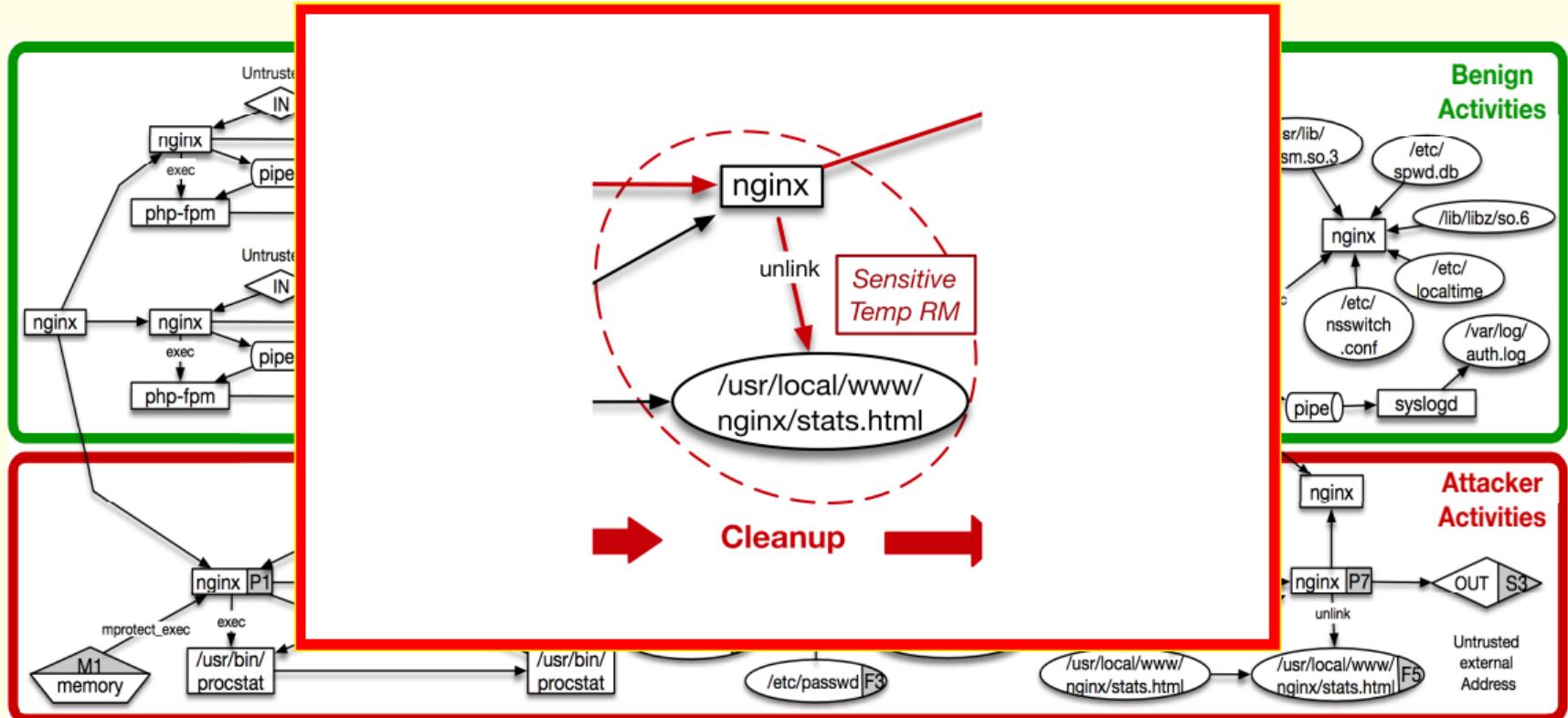
Illustrative Example



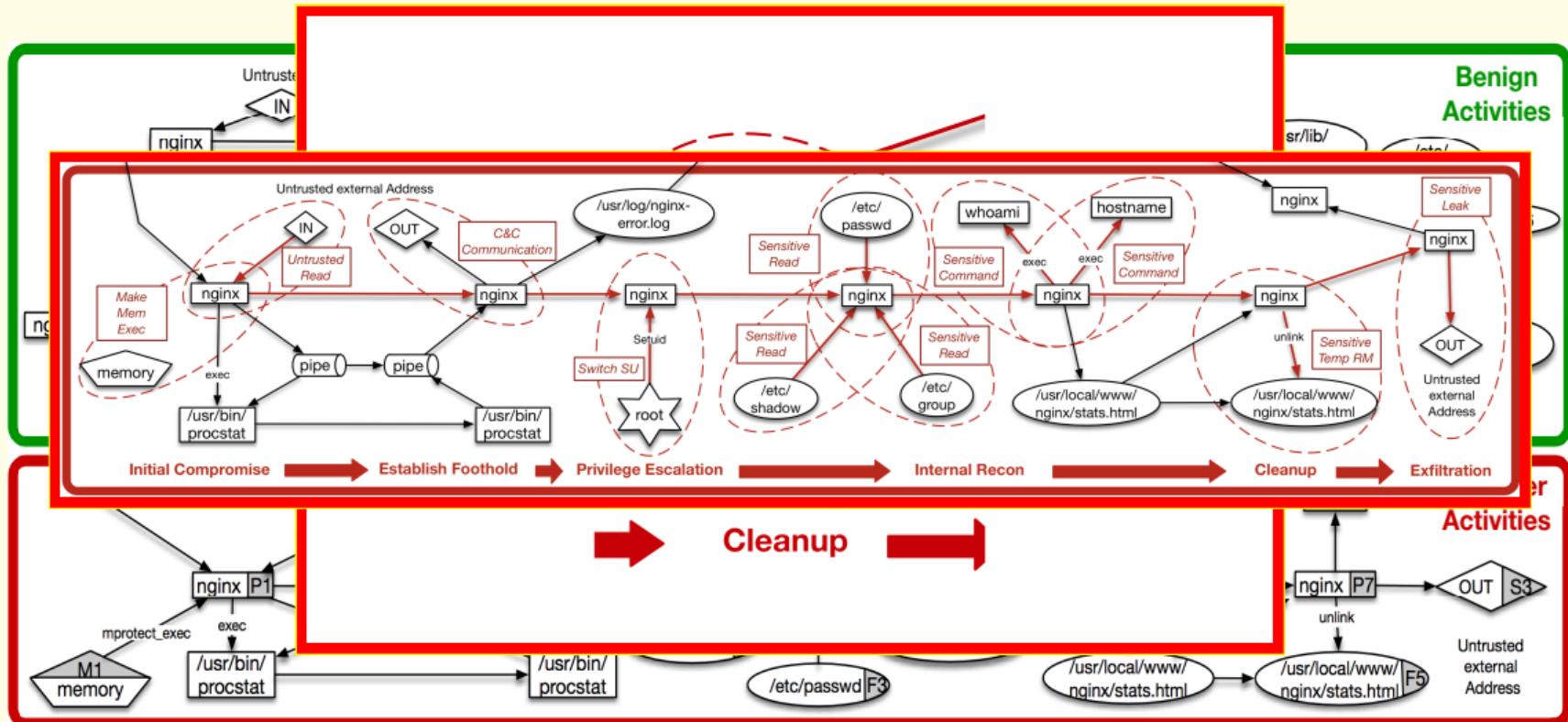
Illustrative Example



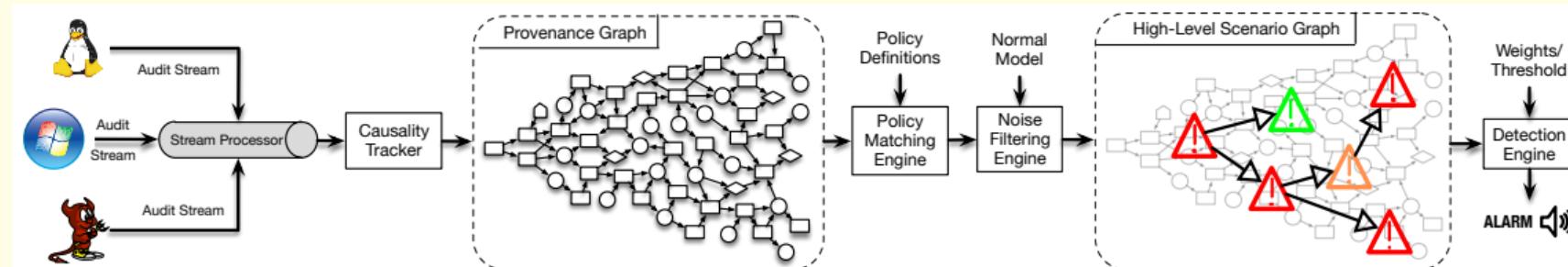
Illustrative Example



Illustrative Example



HOLMES Architecture



- Develop TTP specifications over audit logs
- Use specifications to detect TTPs
- Filter noise based on data quantities of benign information flows, measured in bytes transferred
- Construct high-level graph (HSG) that correlates individual alerts/TTPs
- Derive campaign detection signal from graph

Example TTP specifications

APT Stage	TTP	Event Family	Events	Severity	Prerequisites
<i>Initial_Compromise(P)</i>	<i>Untrusted_Read(S, P)</i>	READ	FileIoRead (Windows), read/pread/readv/preadv (Linux,BSD)	L	$S.ip \notin \{\text{Trusted_IP_Addresses}\}$
	<i>Make_Mem_Exec(P, M)</i>	MPROT	VirtualAlloc (Windows), mprotect (Linux,BSD)	M	$\$PROT_EXEC\$ \in M.flags$ $\wedge \exists \text{Untrusted_Read}(?, P') : path_factor(P', P) \leq path_thres$
<i>Establish_Foothold(P)</i>	<i>Shell_Exec(F, P)</i>	EXEC	ProcessStart (Windows), execve/fexecve (Linux,BSD)	M	$F.path \in \{\text{Command_Line_Utilities}\}$ $\wedge \exists \text{Initial_Compromise}(P') : path_factor(P', P) \leq path_thres$

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

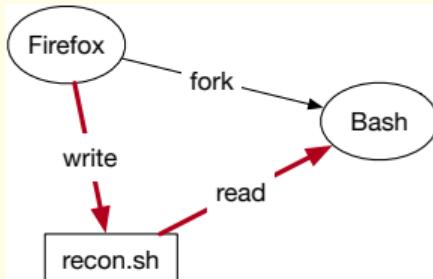
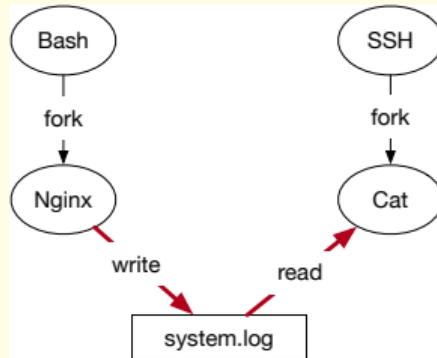
Entity types: P=Process, F=File, S=Socket, M=Memory, U=User.

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.)

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

- *path_factor* value computed incrementally in real-time

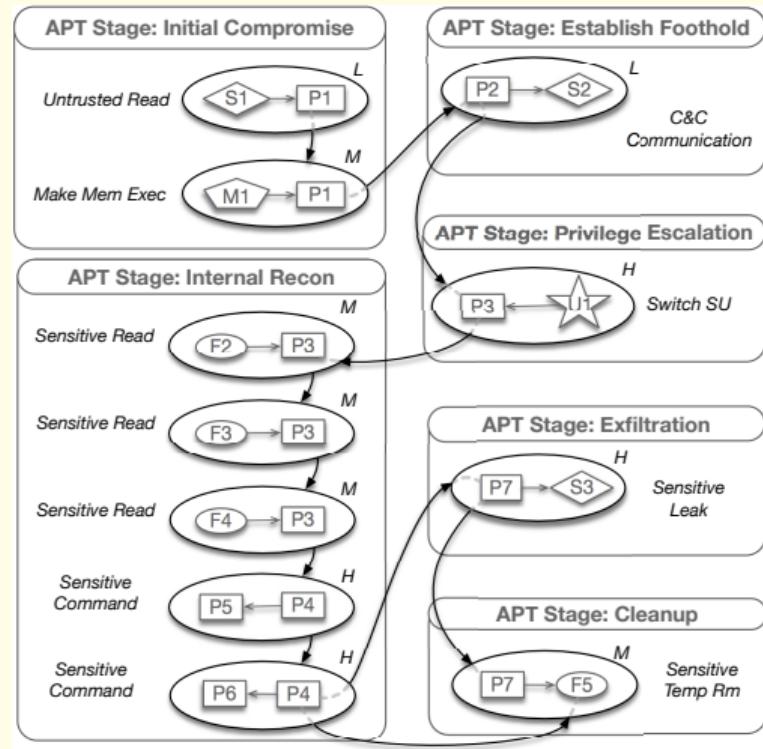
APT Stage	TTP	Event Family	Severity	Prerequisites
<i>Complete_Mission(P)</i>	<i>Sensitive_Leak(P, S)</i>	SEND	H	$S.ip \notin \{\text{Trusted_IP_Addresses}\}$ $\wedge \exists \text{Internal_Reconnaissance}(P') : \text{path_factor}(P', P) \leq \text{path_thres}$ $\wedge \exists \text{Initial_Compromise}(P'')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

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.

Signal Correlation, HSG, and Threat Tuples

- A TTP is matched and added to the HSG if all its prerequisites are satisfied.
- HSG → 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

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

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

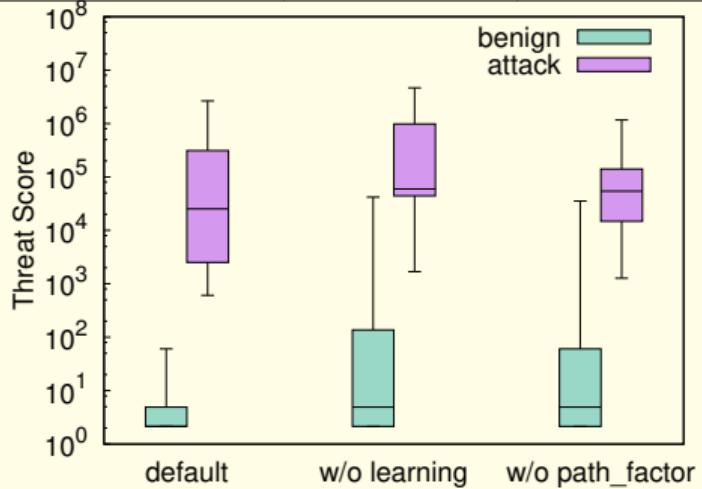
Dataset 1: Using this dataset, we measure the optimal threshold value

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

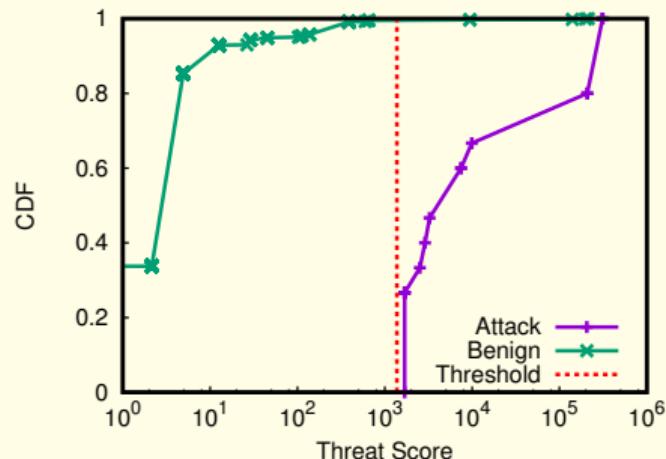
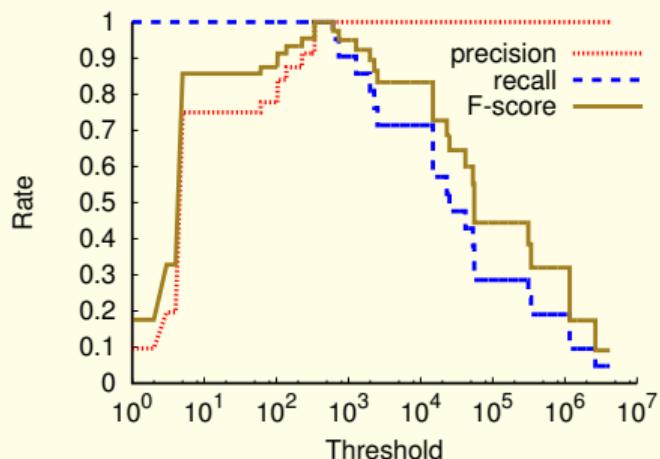
Dataset 2: live detection in a setting that we have no prior knowledge of when or how red-team is conducting the attacks.

- After this experiment, dataset has been released publicly.

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	



Optimal threshold Value and Live Experiment Results



- F-score maximum at [338.25, 608.26] for 6 APT stages
 - Average severity of each APT step = 2.09
- Threshold set for Live experiment (7 APT stages): $2.09^{\sum_{i=1}^7 w_i} = 2.09^{9.8} = 1378$
- 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.

- Dependence explosion mitigation by using the concept of minimum ancestral cover
- Benign system activities pruning 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.