Causality Inference for Attack Investigation

Ashish Gehani, SRI

Shonan 139

See:

Yonghwi Kwon, Fei Wang, Weihang Wang, Kyu Hyung Lee, Wen-Chuan Lee, Shiqing Ma, Xiangyu Zhang, Dongyan Xu, Somesh Jha, Gabriela Ciocarlie, Ashish Gehani, and Vinod Yegneswaran

MCI: Modeling-based Causality Inference in Audit Logging for Attack Investigation

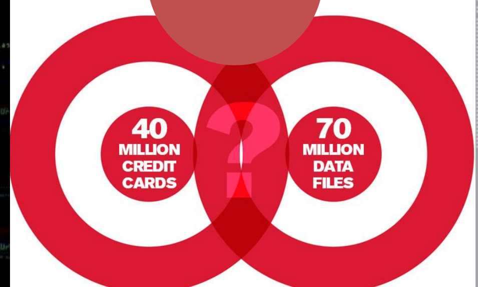
25th Annual Network and Distributed System Security Symposium (NDSS), 2018

Cyberattacks are becoming more sophisticated

- **Advanced Persistent Threat (APT)**

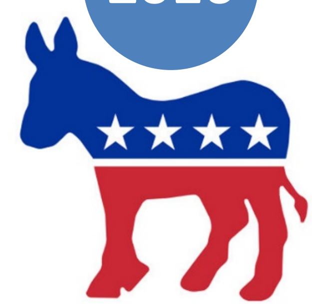
Targeted: Targets specific organizations to exfiltrate information or disrupt the systems.

Infrastructure (Nuclear plants)	Business (Target® Data Breach)	Government (OPM: Office of Personnel Management)	Politics (DNC email hack)
2010	2013	2015	2016



$18 million
40 million customers

18 million employees



Multiple stages of APTs

1. Reconnaissance*:* Learn the target organization

2. Infiltration*:* Enter into the victim via social-engineering (e.g., phishing) or vulnerabilities (e.g., zero-day)

3. Discovery and capture*:* Stay *low* and operate *slowly* to avoid detection while discovering critical machines and/or information

4. Exfiltration/Disruption*:* Send the captured secret information to attackers or destroy the systems

Combatting APTs is challenging

3. Discovery and capture: Stay *low* and operate *slowly* to avoid detection while discovering critical machines and/or information.

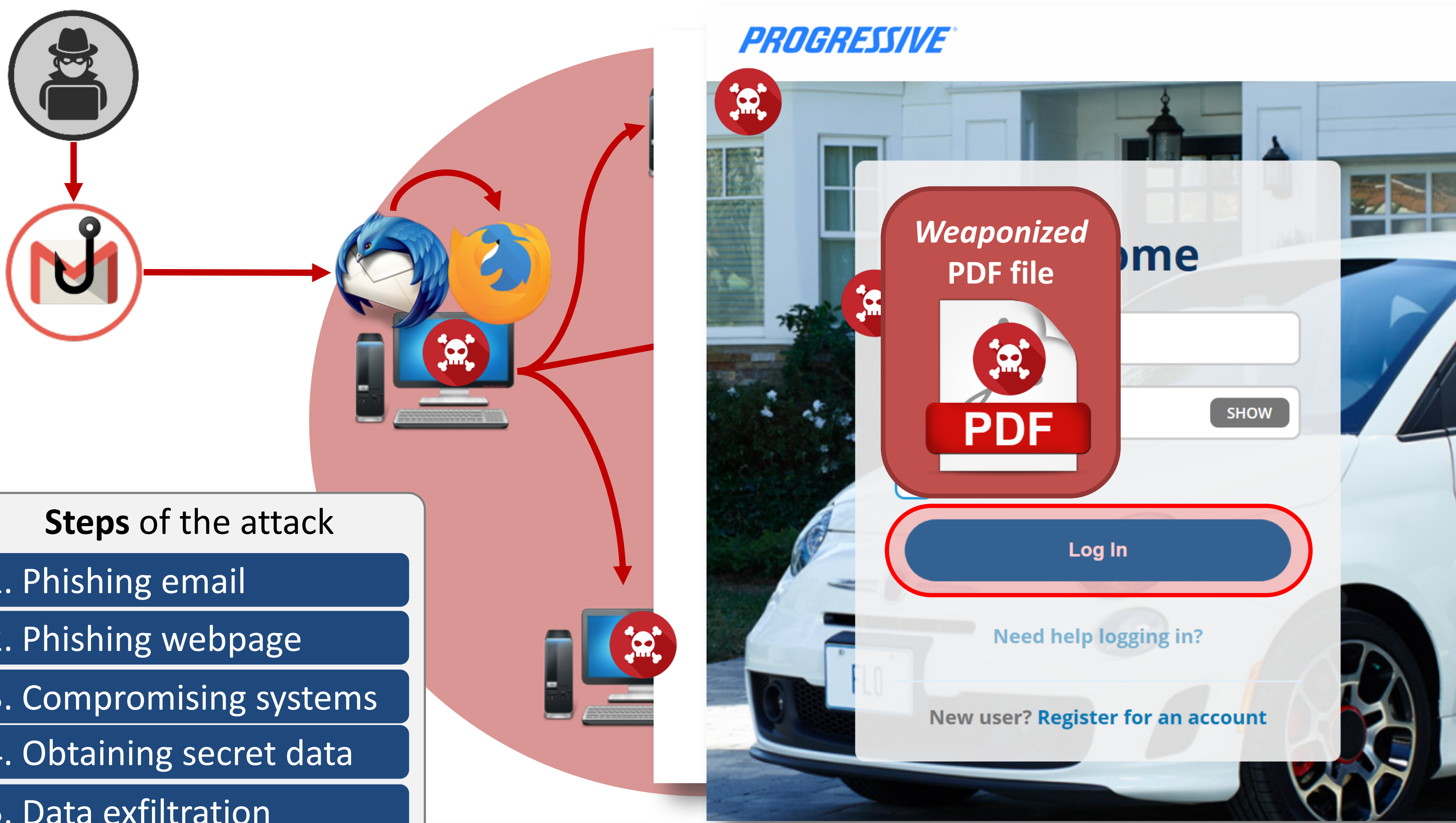
(Whitelisted) benign built-in software

APT attackers often leverage *benign built-in software (e.g., web-browsers and email clients* that are *already whitelisted)* to avoid detection.

Low and slow (Stealthy)

Incidents are often detected after a *few months*.

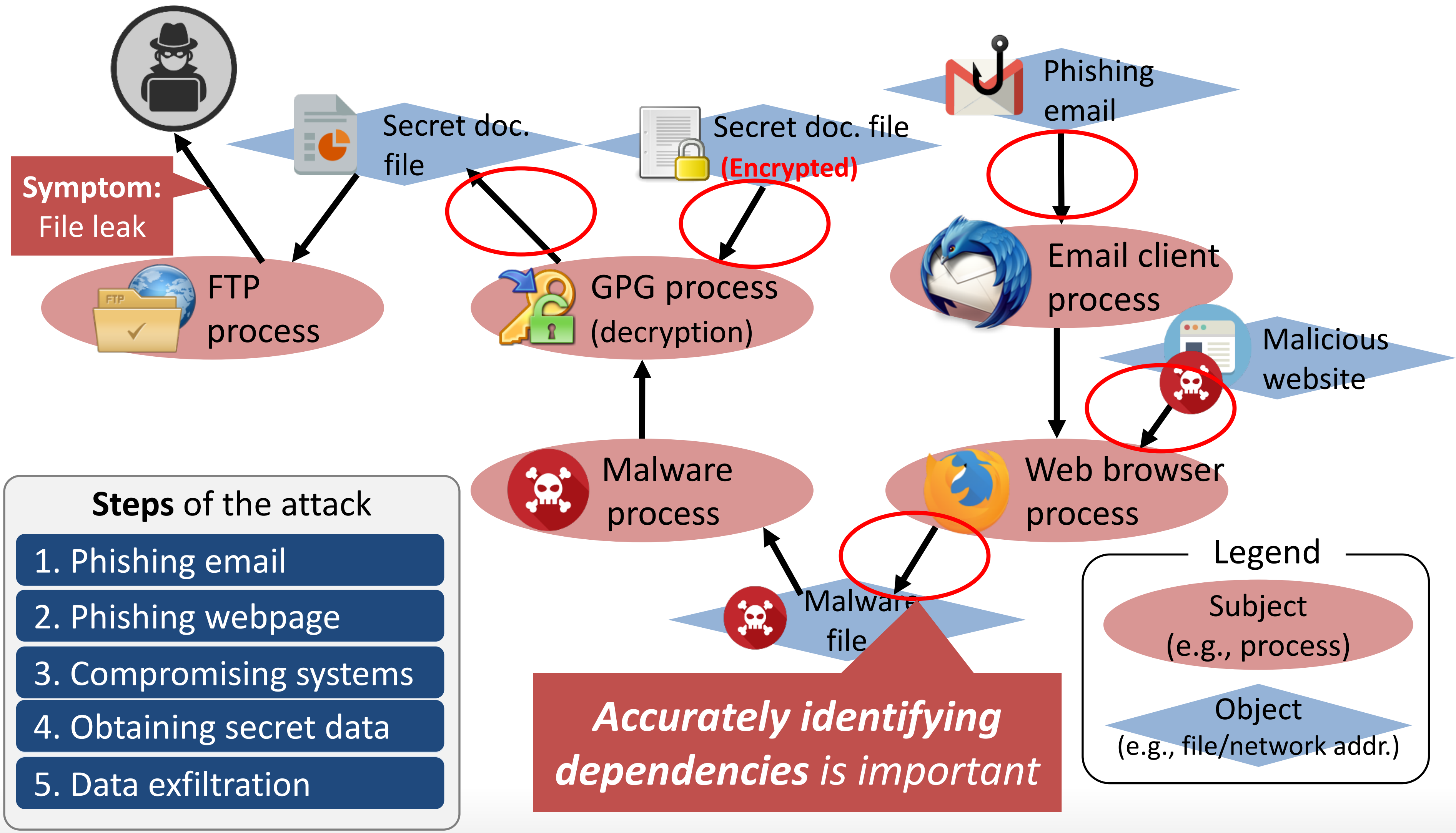
Example APT: Data exfiltration
(exerted from *real-world APTs*)



Steps of the attack

1. Phishing email
2. Phishing webpage
3. Compromising systems
4. Obtaining secret data
5. Data exfiltration

Obtaining ***the ideal causal graph***
from the symptom to the origin of attack (email)

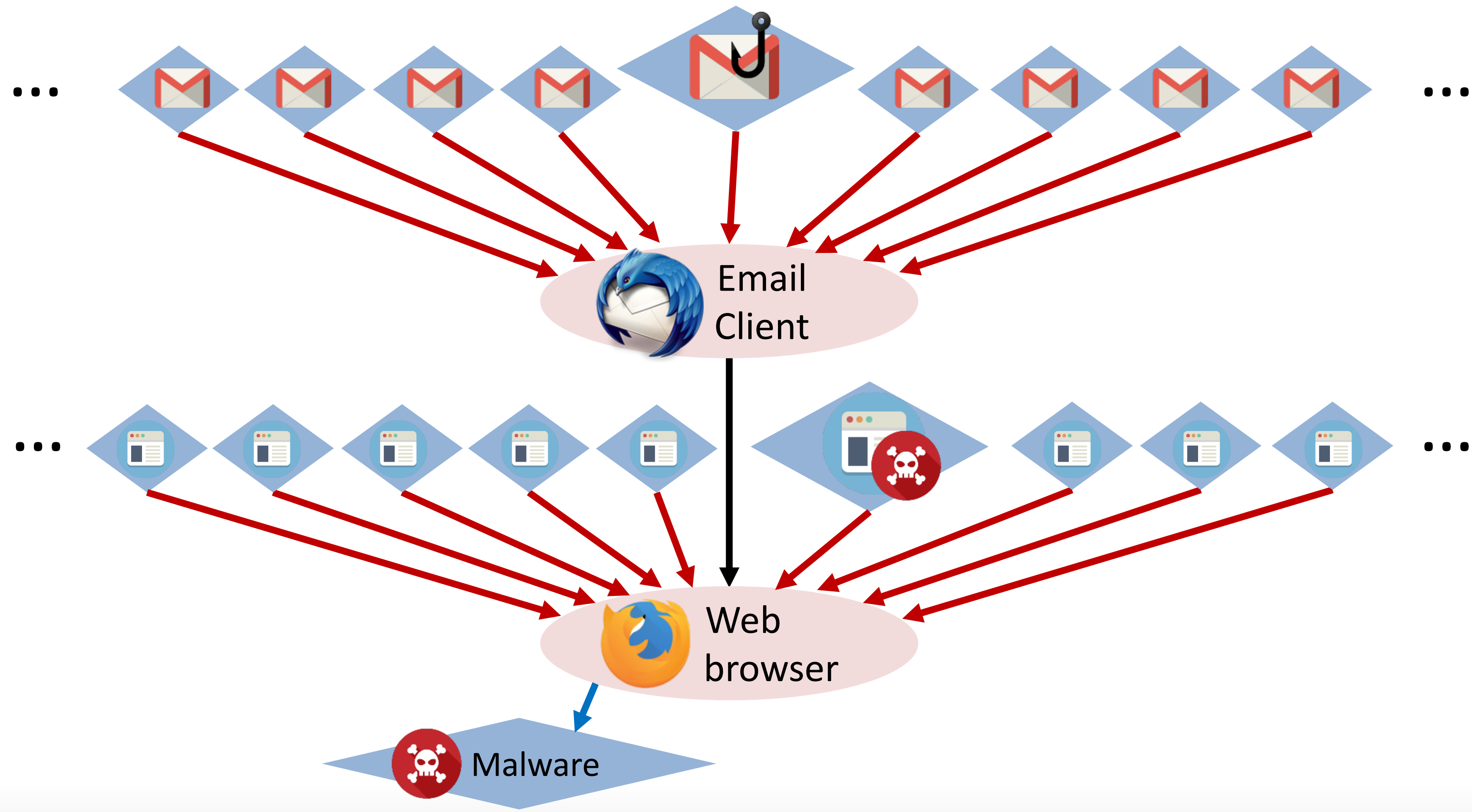


Existing attack investigation technique Type 1: ***Audit-logging***

- Record system calls (e.g., socket read and file write) and detect dependencies between them
  - Coarse-grained assumptions:
    1. System calls operate **on the *same file*** are related.
    2. Within ***the same process, output*** *system calls* are dependent on ***all preceding input*** *system calls*.

Coarse-grained assumptions cause
false dependencies

Dependency Explosion in Audit-logging



Dependency Explosion in Audit-logging

A causal graph consisting of

55 processes, ***41*** files, and ***415*** network addresses.

(only 5 processes, 5 files, 12 network addresses are relevant)

False dependencies cause

Dependency Explosion!

(Taking *from days to weeks* to examine)

Existing attack investigation technique Type 2: ***Taint analysis***

- Track dependency (e.g., data dependency) by monitoring the data propagation of individual operations (e.g., assignment and calculation)

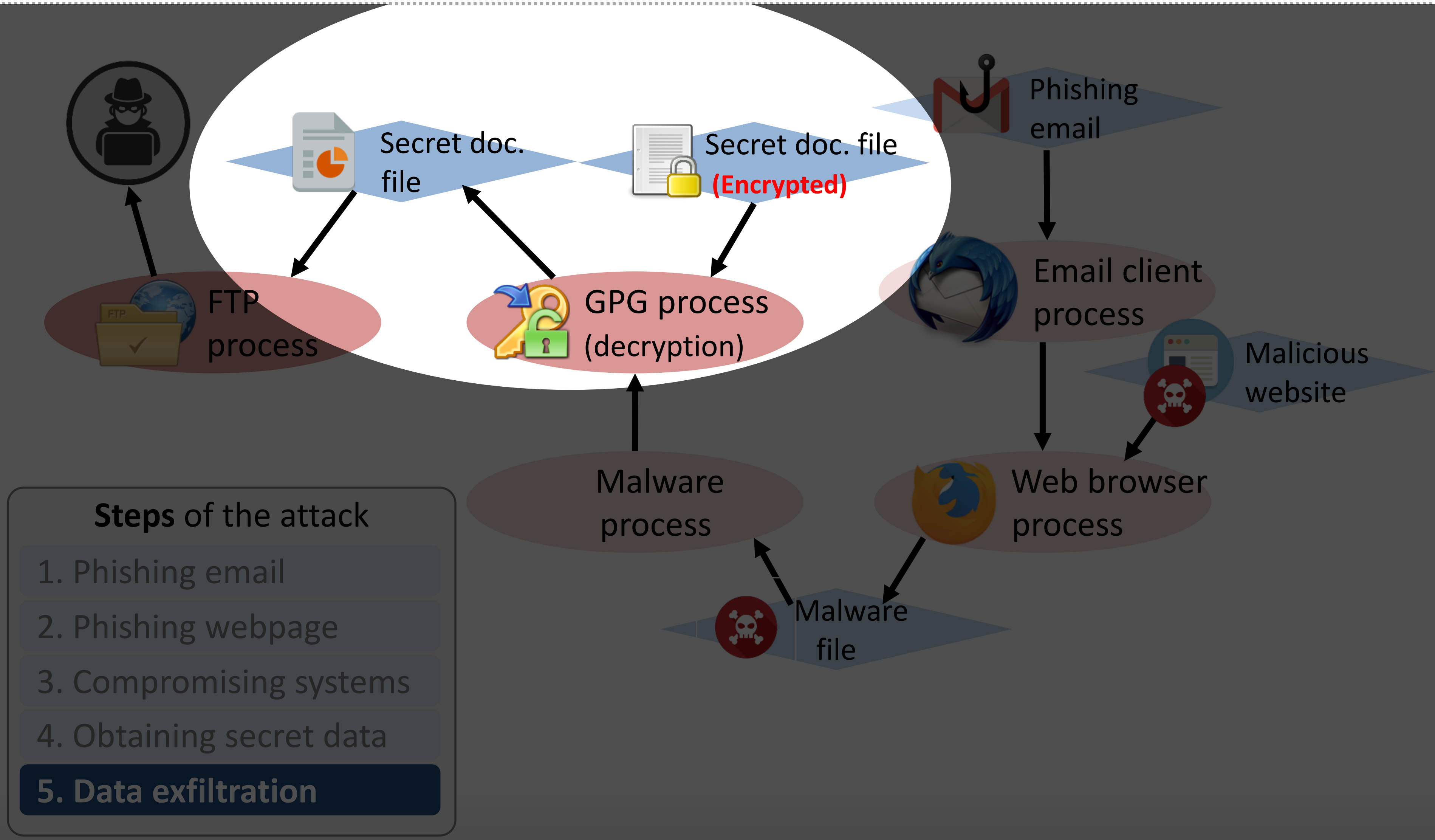
1. **x** = *input*();
2. **y** = **x** + 1;

Data-dependency
(**y** is data dependent on **x**)

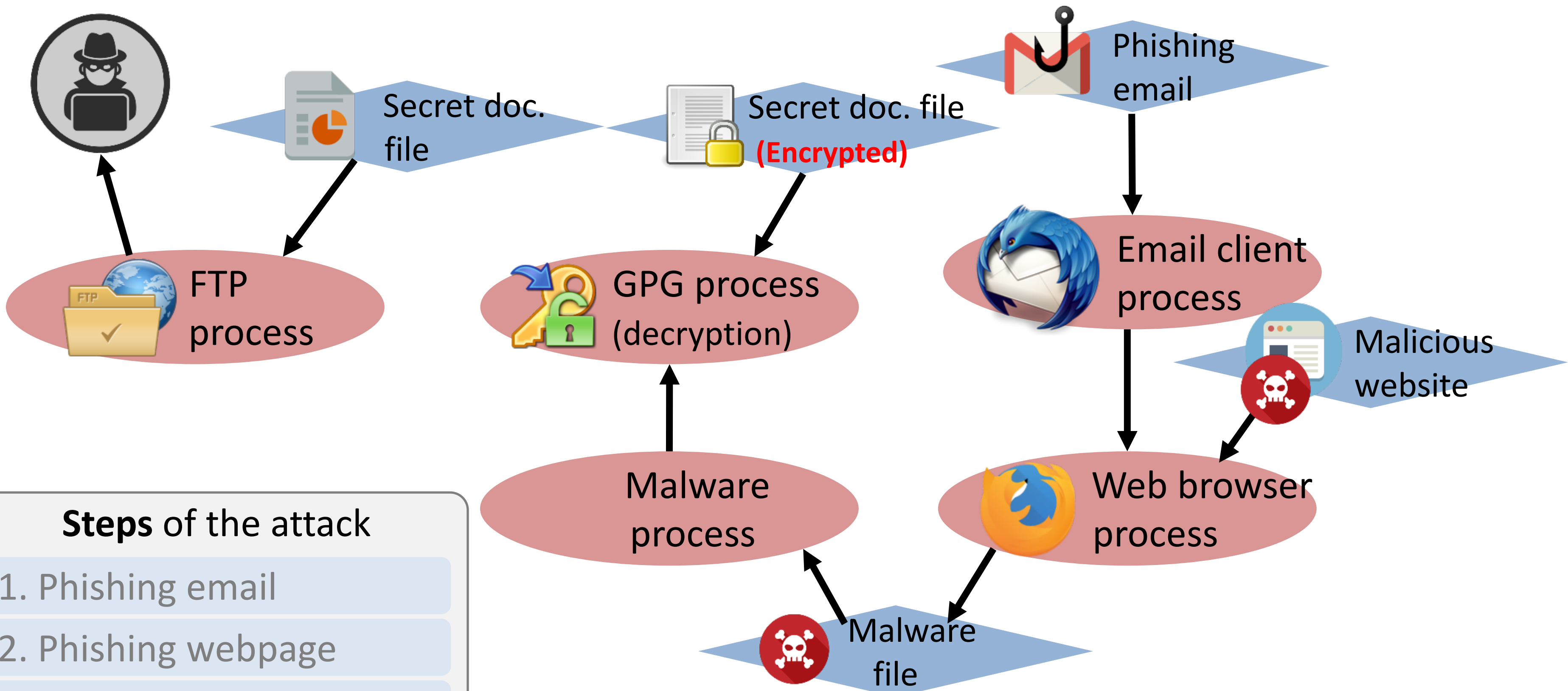
Significant overhead caused by monitoring every instruction

Taint analysis techniques have difficulty handling ***Control Dependency***

Taint-analysis ***fails to track*** dependencies



Taint-analysis ***fails to track*** dependencies



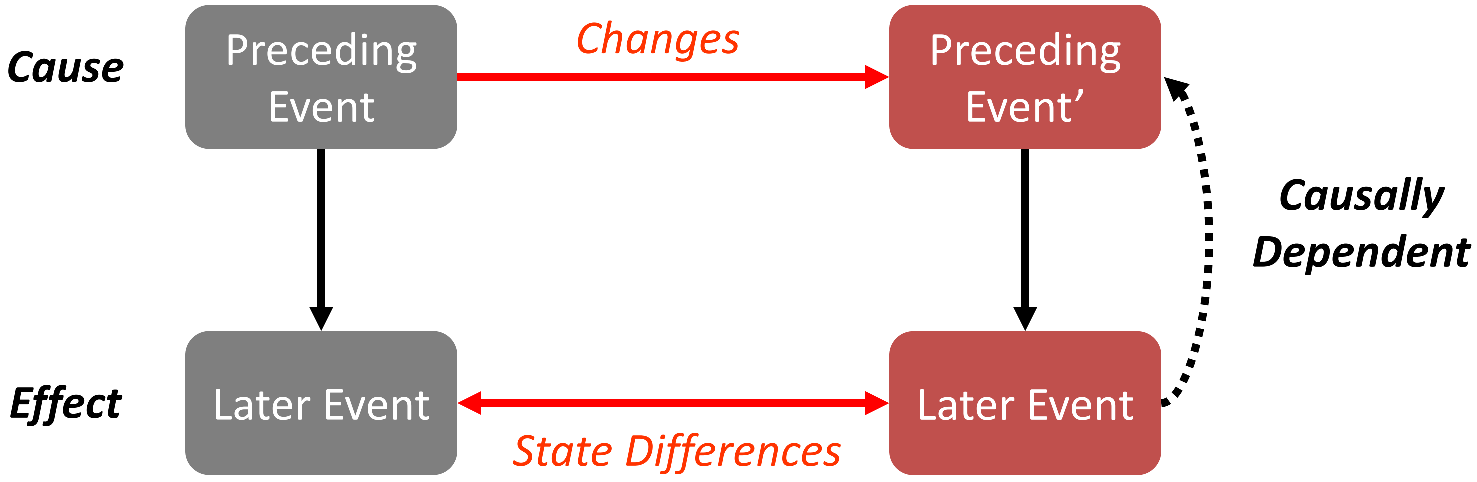
Steps of the attack

1. Phishing email
2. Phishing webpage
3. Compromising systems
4. Obtaining secret data
5. **Data exfiltration**

LDX: Lightweight dual execution for causality inference [ASPLOS'16]

- The original concept of ***counter-factual causality***

  Given two events (e.g., system calls),
  a later event is causally dependent on a preceding event,
  if ***changes* at the *preceding* event** lead to **state *differences* in the *later* event.**



LDX is significantly faster and more accurate *than state-of-the-art taint-analysis techniques*

6.08% average **runtime overhead** on 12 SPEC CPU2006 and 12 real-world applications

3 **times more accurate** than the state-of-art taint analysis techniques (i.e., Taintgrind and Libdft)

Requires instrumentation *of target programs*

Toward practical causality inference in the *enterprise environment*

- *Changing end-user systems* is *not allowed*
  - Modifications to commercial programs are not allowed.
  - Organizations do not allow modified programs and/or kernel to be used.

Instrumentation free
causality inference technique is required

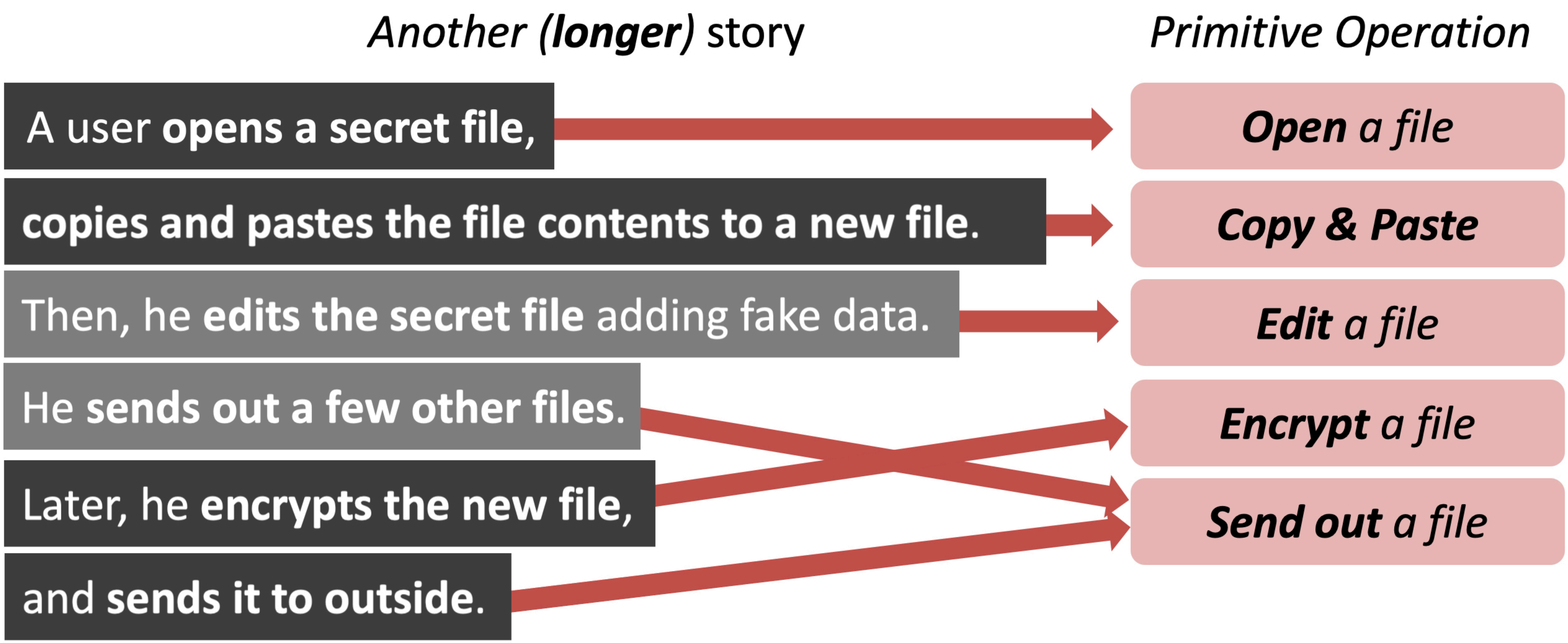
Intuition behind *instrumentation free* causality inference: ***Behavior decomposition***

- *A complex system-wide behavior* can be decomposed into *primitive operations*

	Primitive Operation
A user **opens a secret file,**	***Open** a file*
copies and pastes the file contents to a new file.	***Copy & Paste***
Then, he **edits the new file containing the secret.**	***Edit** a file*
Later, he **encrypts the new file,**	***Encrypt** a file*
and **sends it outside.**	***Send out** a file*

Intuition behind *instrumentation free* causality inference: ***Behavior decomposition***

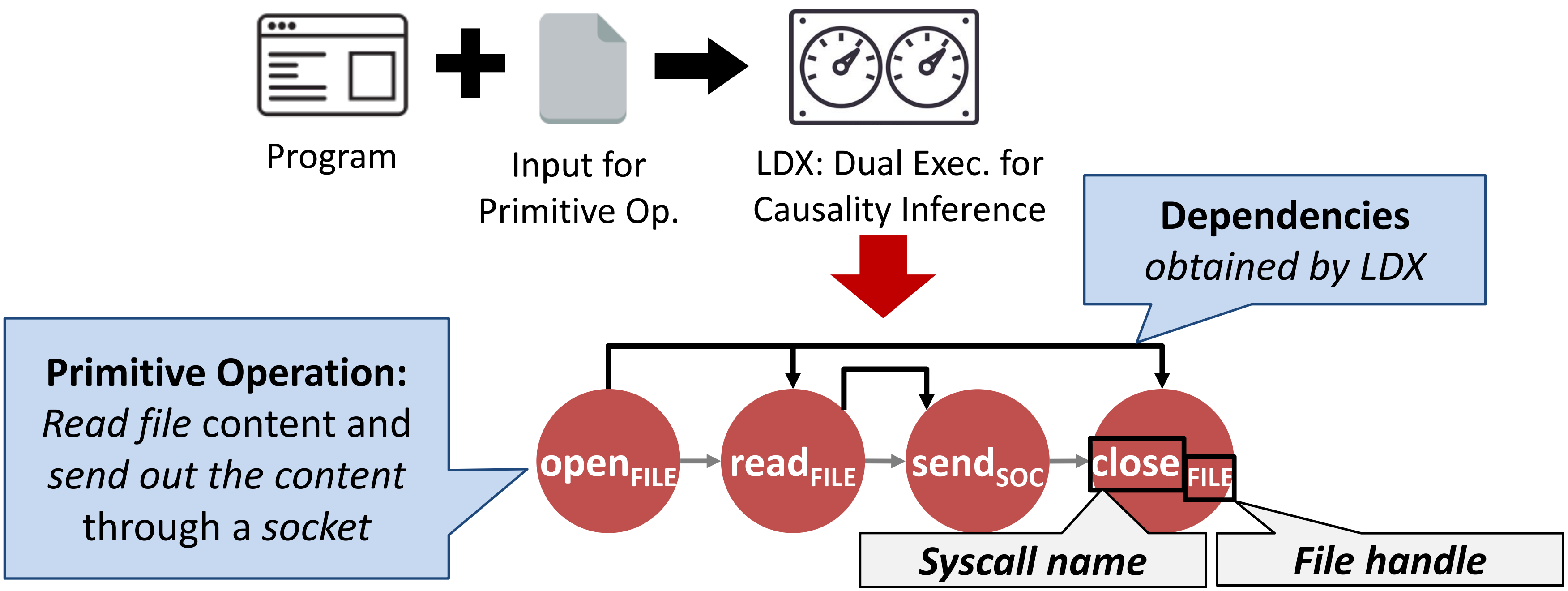
- Primitive operations can be used to compose other *combinational complex behaviors*



MCI: *Model-based* Causality Inference

1. Acquire causal models *(Offline)*

For each program, it uses **LDX *(in offline)*** to acquire *causal models* for *primitive operations* (e.g., *opening a file, copy and paste, and edit a file*).

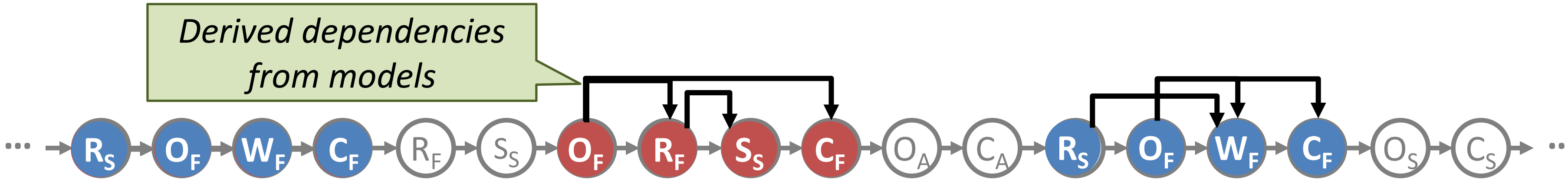


Causal Model:
A sequence of system calls with inter-dependencies

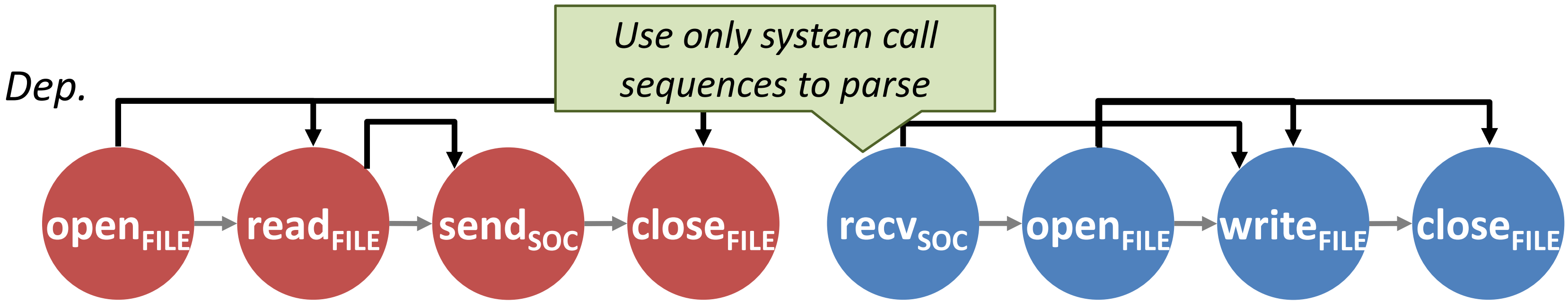
MCI: *Model-based* Causality Inference

2. Parse audit-logs with the causal models

MCI parses audit-logs into concrete model instances



Production audit-log (system call trace): Circles represent system calls and arrows mean the orders. *No dependency information between system calls.*



Causal models: Causal model 1 (Red) and Causal model 2 (Blue)

Challenges in model-based causality inference

1. Language complexity to describe syscall sequences

– Complex system call subsequences of causal models requires *expressive language*

- Context-free: **R**$r^n w^n$ (e.g., Rrw, Rrrww, Rrrrwww, …)

```
attach[...] = parse( recv(...) );        // recv
for ( i = 0; i < n; i++ )                // (read)^n
  read( attach[i], buf, ... );
for ( i = 0; i < n; i++ )                // (write)^n
  write( fout[i], buf, ... );
```

- Context-sensitive: **R**$r^n w^m c^n c^m$ (e.g., **R**rrwccc, **R**rrwwcccc, …)

```
attach[...] = parse( recv(...) );     // recv
for ( i = 0; i < n; i++ )             // (read)^n
  read( attach[i], buf, ... );
for ( j = 0; j < m; j++ )             // (write)^m
  write( fout[j],  compress(buf) );
for ( i = 0; i < n; i++ )             // (close)^n
  close( attach[i] );
for ( j = 0; j < m; j++ )             // (close)^m
  close( fout[j] );
```

Challenges in model-based causality inference

1. ***Language complexity*** to describe syscall sequences

– Complex system call subsequences of causal models requires *expressive language*

- Context-free: $\mathbf{R}r^nw^n$ (e.g., Rrw, Rrrww, Rrrrwww, …)

```
attach[...] = parse( recv(...) );         // recv
for ( i = 0; i < n; i++ )                  // (read)^n
  read( attach[i], buf, ... );
for ( i = 0; i < n; i++ )                  // (write)^n
  write( fout[i], buf, ... );
```

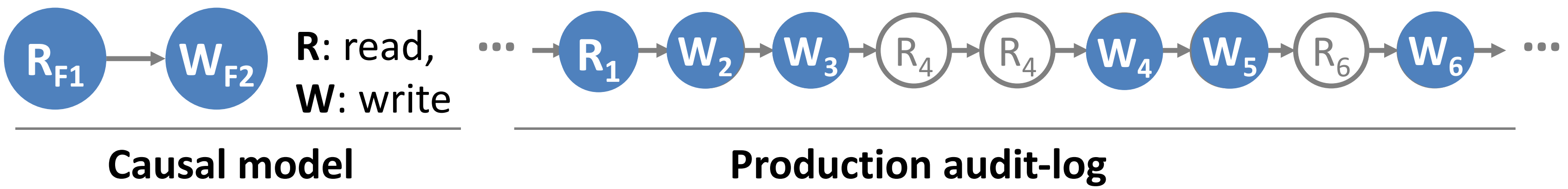
- Context-sensitive: $\mathbf{R}r^nw^mc^nc^m$ (e.g., **R**rrwccc, **R**rrwwcccc, …)

More expressive languages lead to ***higher costs*** in parsing

Challenges in model-based causality inference

2. ***Ambiguity*** in parsing

– Some system calls in audit-logs can be parsed to multiple causal model instances.



*Different causalities are derived from different model instances, causing **incorrect causality***

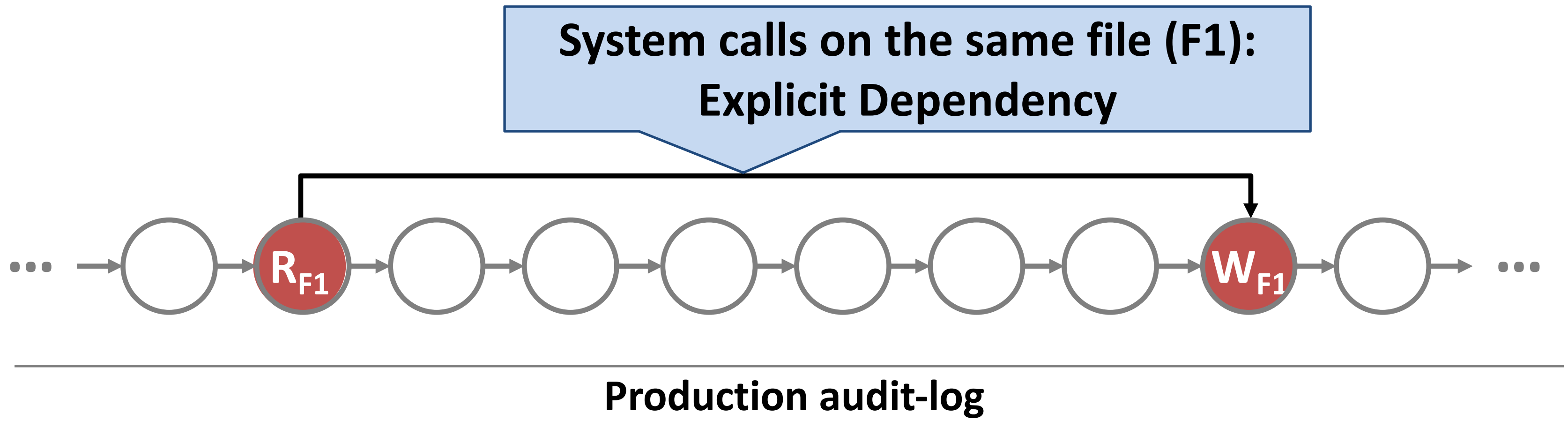
Overcoming challenges by ***leveraging dependencies*** in audit-logs

Problem

Treating an audit-log as a ***plain*** sequence of system calls ***without dependencies***

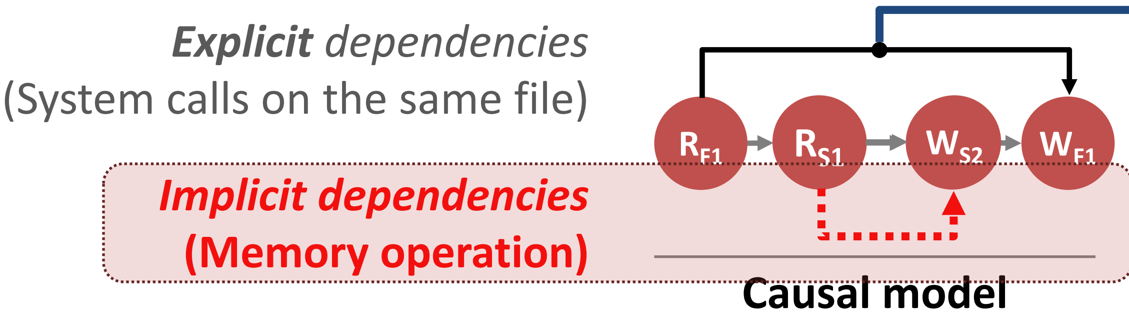
Observation

Certain dependencies can be extracted by preprocessing audit-logs to ***reduce language complexity and ambiguity***

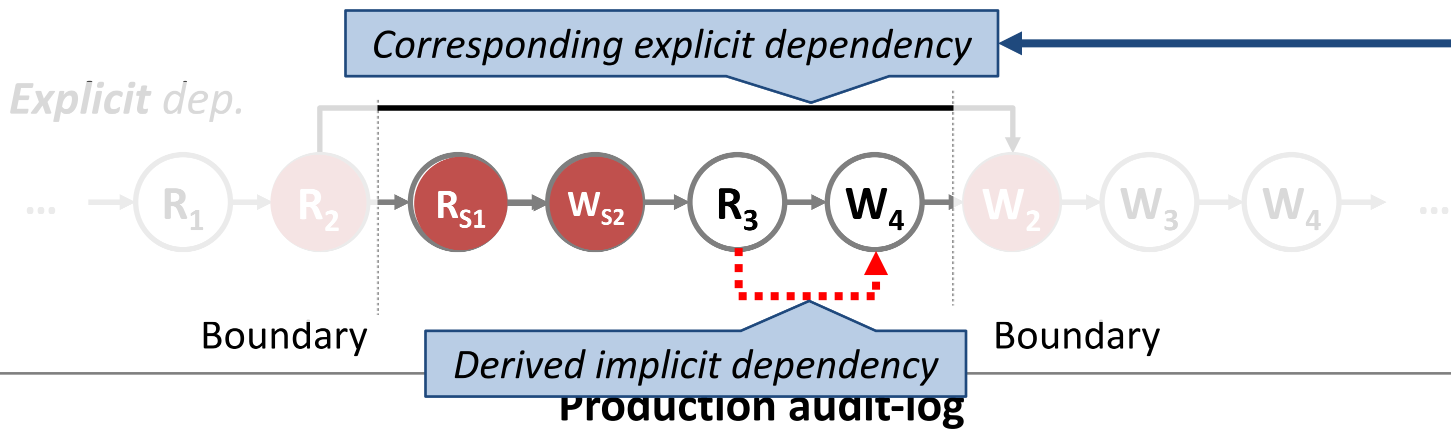


Segmented Parsing by leveraging *explicit dependencies*

- Causal models have ***explicit and implicit*** dependencies



- Idea: Identify *corresponding explicit dependencies* and parse *segments* to derive *implicit dependencies* from causal models



Practical instrumentation free *causality inference:* ***Scalable*** to real-world workloads

- A ***week*** long ***system-wide*** experiments
  - ***Large size programs:*** *Web browser (Firefox), web servers (Apache and nginx), P2P program (Torrent), …*

0.8% FP and 0.6% FN

(ground-truth is obtained by LDX)

- ***3 months*** of ***Purdue*** *web server* workload and ***2 months*** of ***NASA*** *web server* workload
  - ***9 million*** requests (***4.2 million*** unique requests)

2.5% FP and 0.15% FN

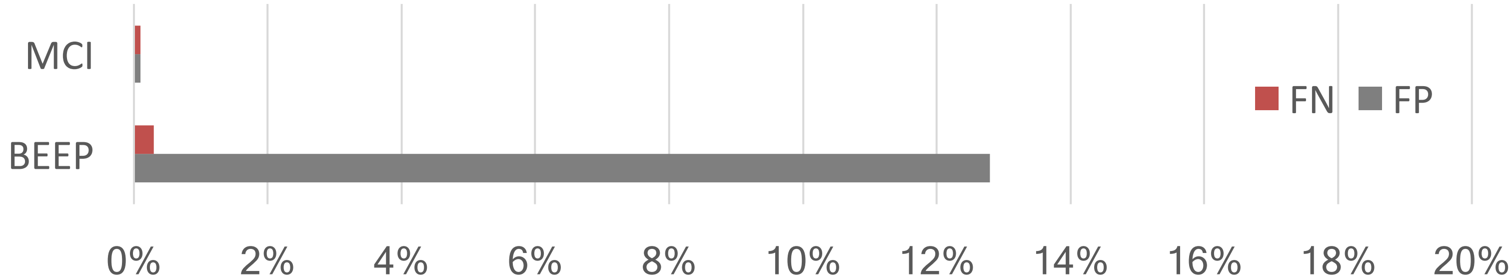
Accurate Causality Inference:

More accurate than *BEEP*

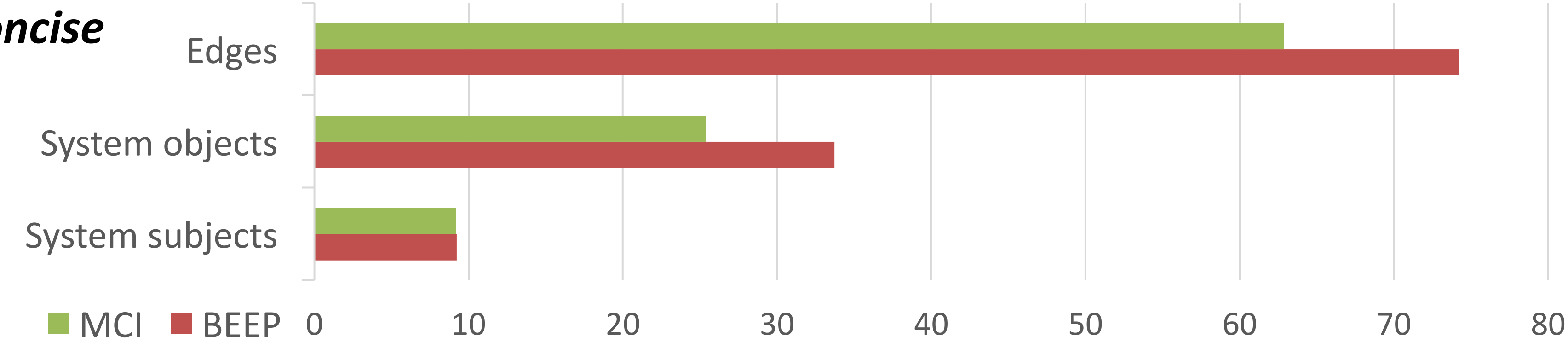
(*state-of-the-art* audit-logging tech. based on *execution partition*)

- Graph by MCI is *accurate and concise*
  - Randomly select *100 system objects* (e.g., files/network addresses) and build causal graphs

Accurate



Concise

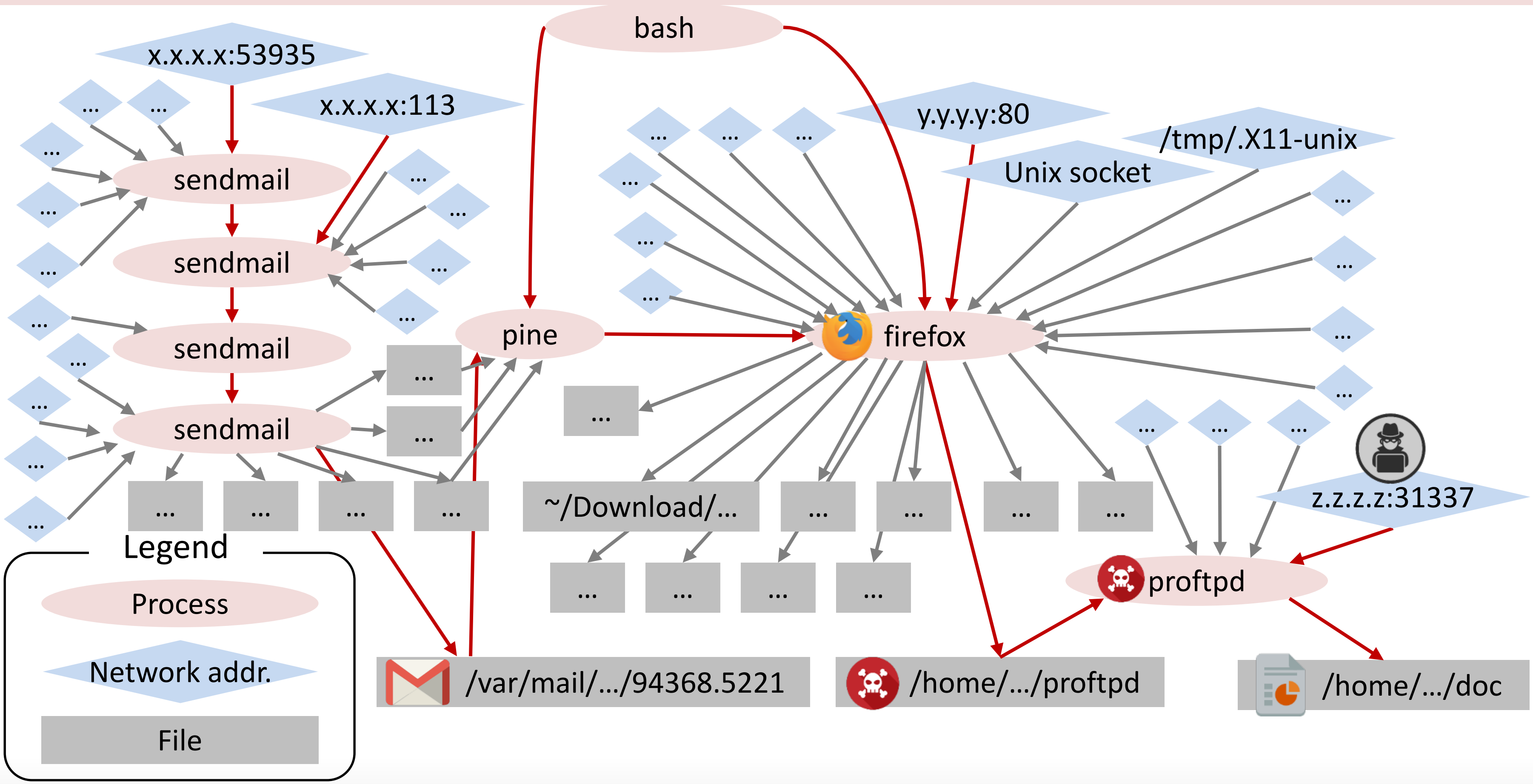


APT attack constructed by professionals

Phishing email + Backdoored FTP + Data exfiltration

A graph generated by state-of-art audit-logging based technique

(19 files, 33 network addrs., 8 processes + @)

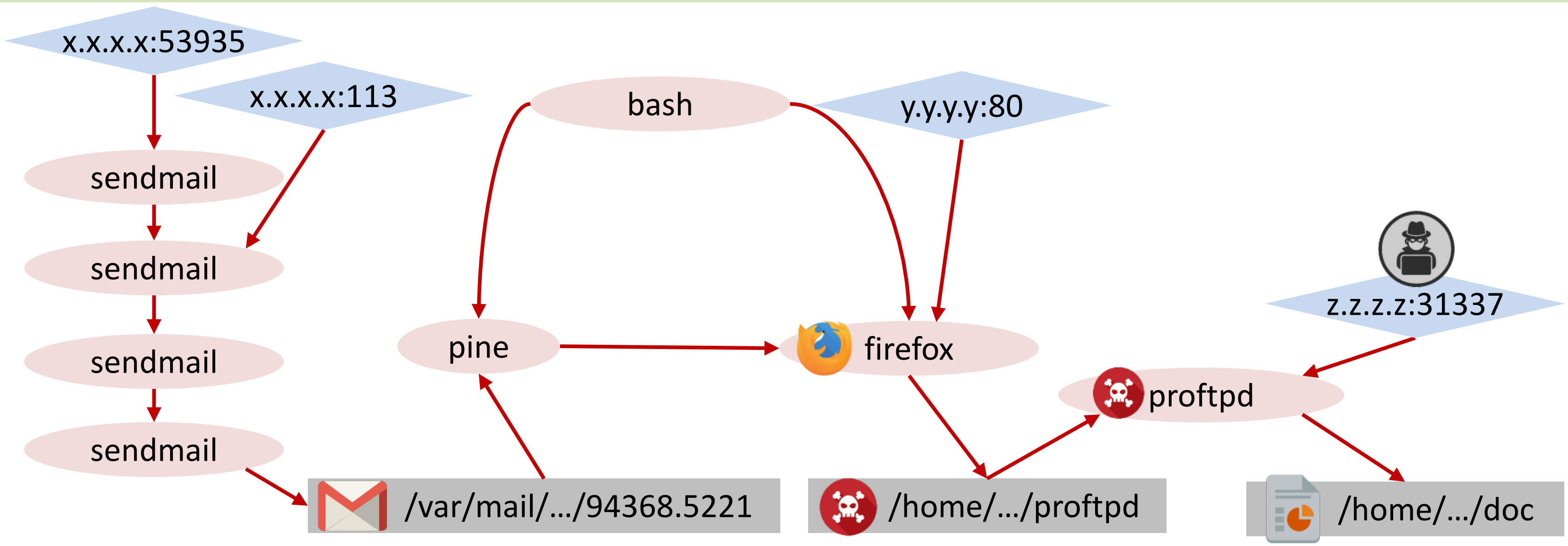


APT attack constructed by professionals

Phishing email + Backdoored FTP + Data exfiltration

A graph generated by **MCI**

(3 files, 4 network addrs., 8 processes)



Concise and precise causal graph including

all and only attack relevant subjects and objects

Conclusion

1. MCI ***directly works*** **on** ***production audit-logs*** without requiring any change on end-user systems (e.g., instrumentation and modified kernels)
2. MCI is ***scalable*** to cope with *large scale log from long-running applications* (e.g., A week long experiment with ***Firefox***)
3. MCI ***precisely*** *infers causality* with negligible FP (< 2.5%) and FN (< 1%)

Yonghwi Kwon, Fei Wang, Weihang Wang, Kyu Hyung Lee, Wen-Chuan Lee, Shiqing Ma, Xiangyu Zhang, Dongyan Xu, Somesh Jha, Gabriela Ciocarlie, Ashish Gehani, and Vinod Yegneswaran
MCI: Modeling-based Causality Inference in Audit Logging for Attack Investigation
25th Annual Network and Distributed System Security Symposium (NDSS), 2018