Définition Formelle de la Relation de Dépendance Causale entre Événements Journalisés

Charles XOSANAVONGSA
CentraleSupélec & Thales

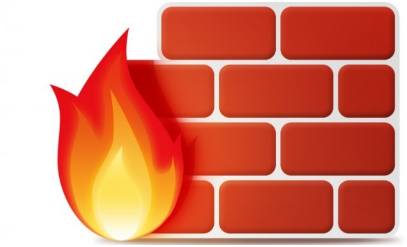
Encadrants :
Eric TOTEL - CentraleSupélec
Olivier BETTAN - Thales

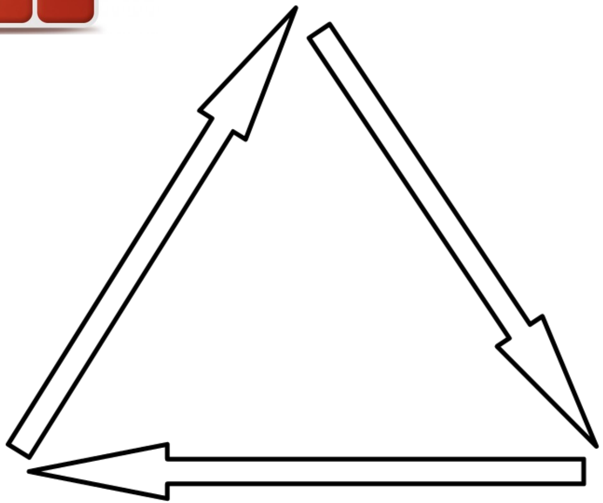
Agenda

Objectif : Susciter votre Curiosité pour la session Poster

- Contexte
- Objectif
- Contribution

Contexte – Un autre triangle de la Sécurité





Prevention

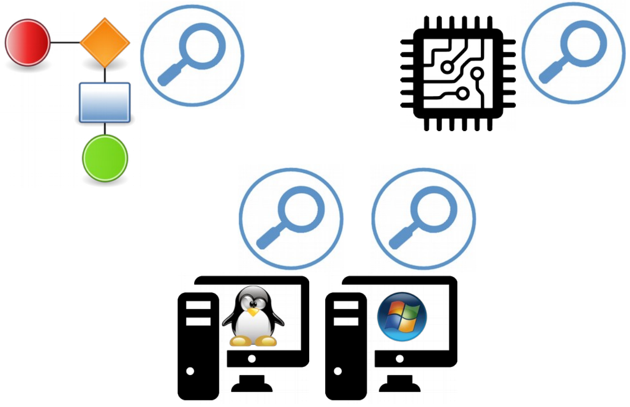
Reaction

Detection

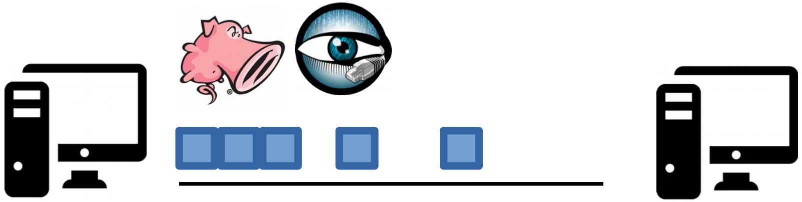
Contexte – Le besoin de Supervision en Sécurité

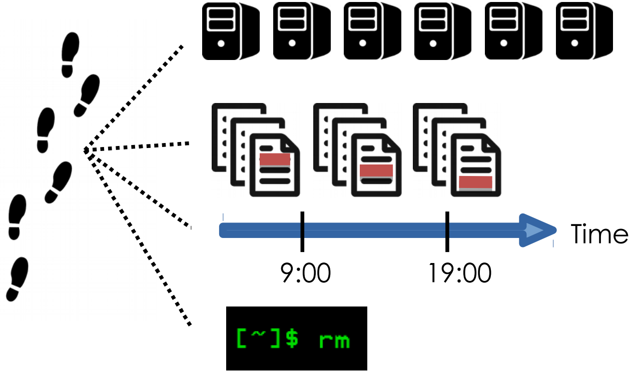
Différentes couches d'abstraction :

- Application (App Logs, CPU, ...)
- Système d'exploitation (Syscalls, ...)
- Réseau (DPI, NIDS, Netflow, ...)



Observation partielle des actions effectuées

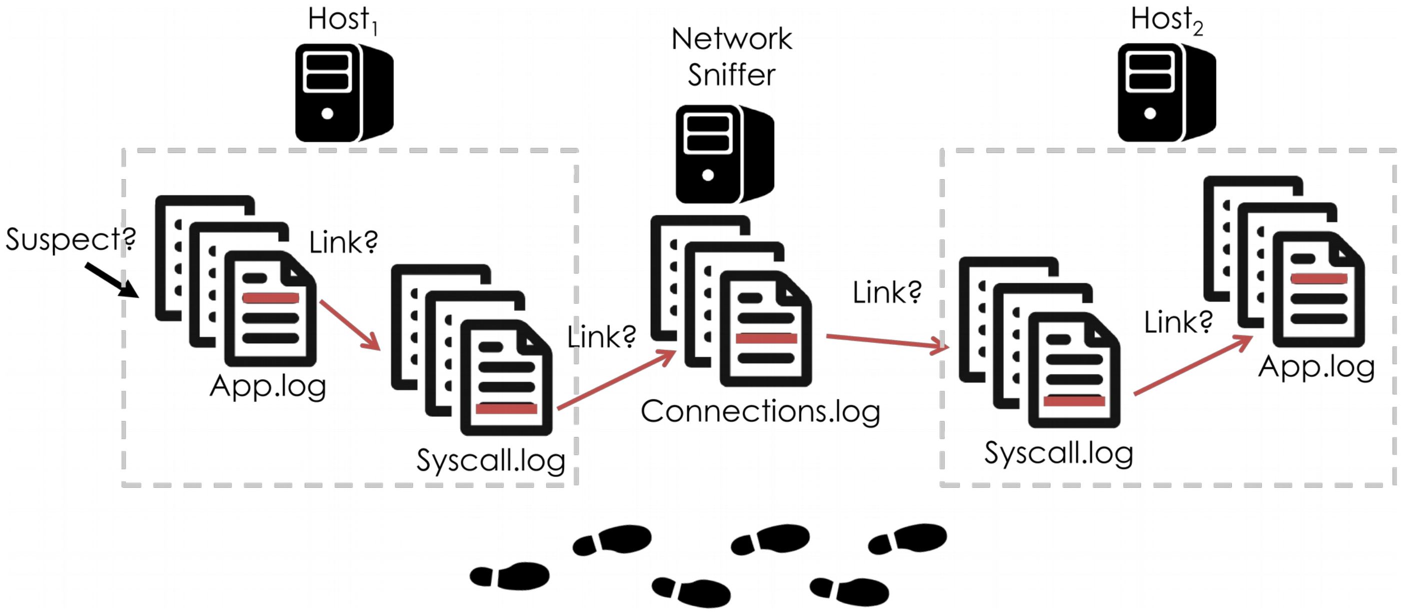




```
[~]$ rm
```

Objectif – Ce qu'un Analyste fait

Recherche de liens de corrélation entre les événements journalisés

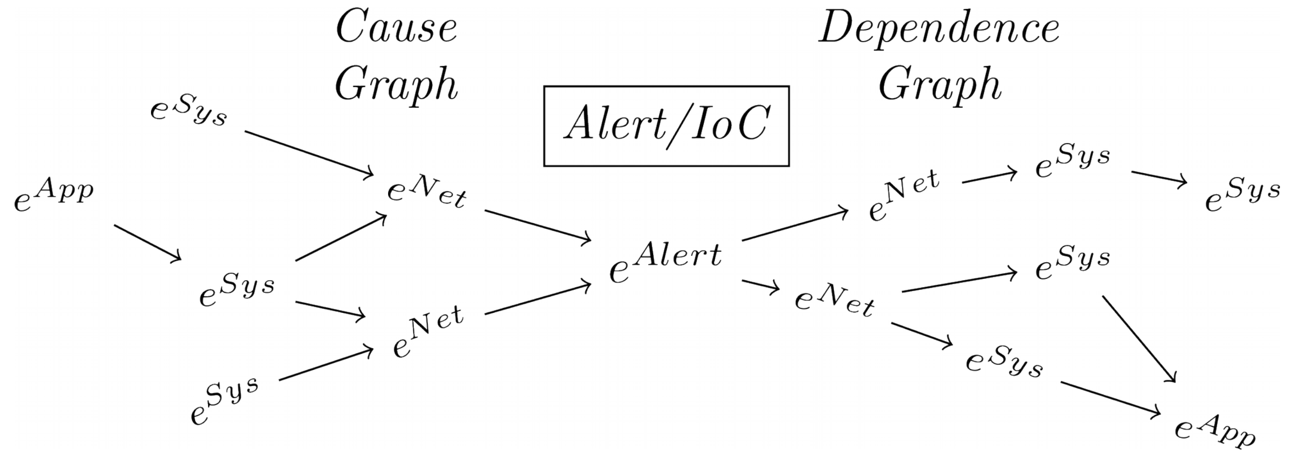


Objectif – Ce qu'un Analyste veut

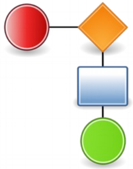
Retrouver l'ensemble des **actions** effectuées par un attaquant

⇒ Découverte de scénarios d'attaque

⇒ Recherche des liens de **dépendances causales** entre les événements correspondant aux actions de l'attaquant



Objectif – Définition de la Dépendance Causale



Constat :

Pas de définition formelle de la dépendance causale entre événements journalisés hétérogènes

definition of
causal dependency?

$$e_1 \xrightarrow{?} e_2$$

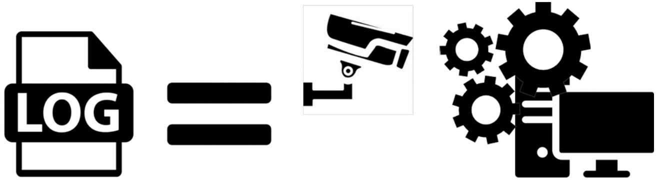
~~Modélisation d'Attaque (Arbres d'Attaque, Graphes d'Attaque, ...)~~

Contribution – Raisonnement

Objectif : Définition de la relation de Dépendance Causale entre Événements

Définition d'Événement : « une action identifiable ayant lieu sur un dispositif et étant enregistrée comme une entrée de journal ».

[European Commission, 2010]



[European Commission, 2010] Standard on logging and monitoring.

Contribution – À propos d'Actions et d'États

Relation de D'Ausbourg [d'Ausbourg, 1994] entre les **états** des **objets** d'un système

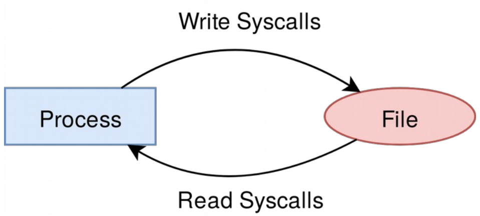
⇒ **flux d'information**

$$(o,t) \rightarrow (o',t')$$

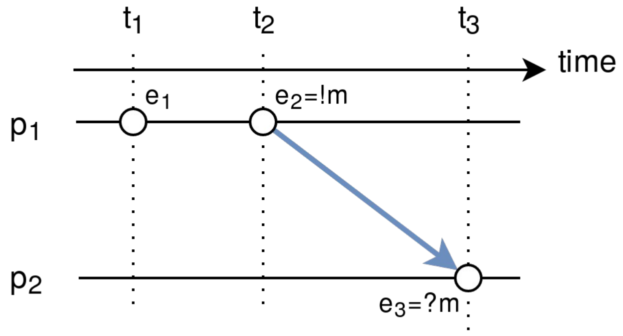
```
a := 1
a := a + 1
a := 0
```

$(a,1) \rightarrow (a,2)$

$(a,2) \neg\!\rightarrow (a,3)$



Relation de Lamport [Lamport, 1978] entre les **actions** effectuées par les processus d'un système distribué.



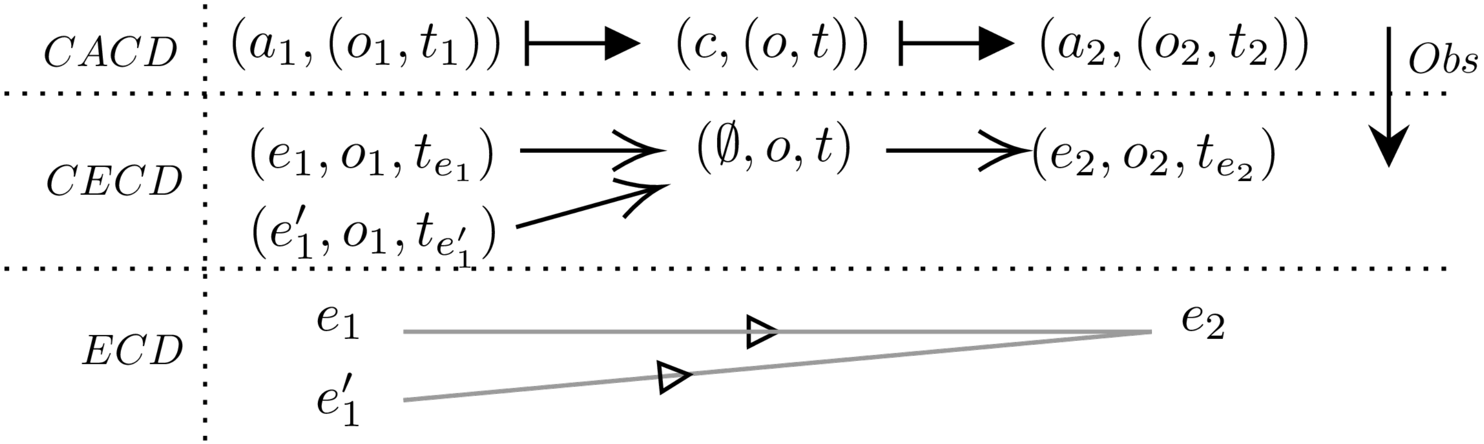
[d'Ausbourg, 1994] Implementing secure dependencies over a network by designing a distributed security subsystem. ESORICS.

[Lamport, 1978] Time, clocks, and the ordering of events in a distributed system. Communications of the ACM.

Contribution – Nouvelles Dépendances Causales

Définitions de 3 nouvelles relations de dépendances causales :

1. Actions Contextuelles (CACD)
2. Événements Contextuels (CECD)
3. Événements Bruts (ECD)



$CACD$
$(a_1,(o_1,t_1)) \longmapsto (c,(o,t)) \longmapsto (a_2,(o_2,t_2))$
Obs
$CECD$
$(e_1,o_1,t_{e_1}) \rightarrow (\emptyset,o,t) \rightarrow (e_2,o_2,t_{e_2})$
$(e'_1,o_1,t_{e'_1})$
ECD
e_1
e'_1
e_2

[XOSANAVONGSA et al. 2019] Discovering Correlations: A Formal Definition of Causal Dependency Among Heterogeneous Events. IEEE EuroS&P.

Conclusion

- Nouveau cadre théorique
  ⇒ Dépendance Causale entre Événements Journalisés ;

- Découverte de liens entre Événements ;

- Implémentation dans un environnement Linux

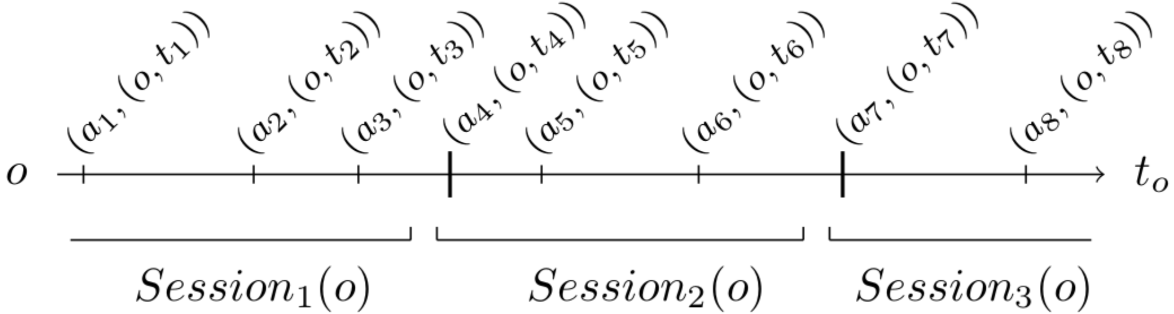
Merci de votre attention

Contribution – Introduction des Sessions

Volonté d’avoir un modèle plus précis

⇒ 2 États consécutifs peuvent être indépendants

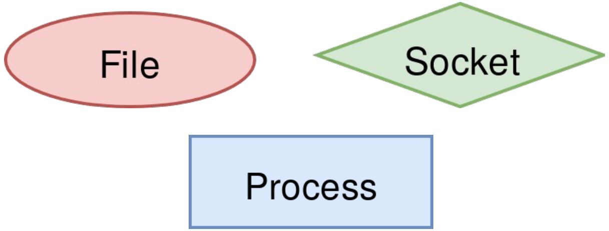
$$
\begin{aligned}
Session_n(o) = \{(a_i, (o, t_i)) \ / \ & \\
& (o, t_i) \rightarrow (o, t_{i+1}) \\
& \wedge (o, t_{end_{n-1}}) \not\rightarrow (o, t_{start_n}) \\
& \wedge (o, t_{end_n}) \not\rightarrow (o, t_{start_{n+1}})\}
\end{aligned}
$$



Contribution – [Def] Action Contextuelle

Deux types d'Objets :

- Passifs (ex : containers d'information) => ~~Actions~~
- Actifs (ex : processus ou le réseau) => Actions



Une **Action** est effecutée dans un certain **État**

Def : Action Contextuelle (a , (o , t))

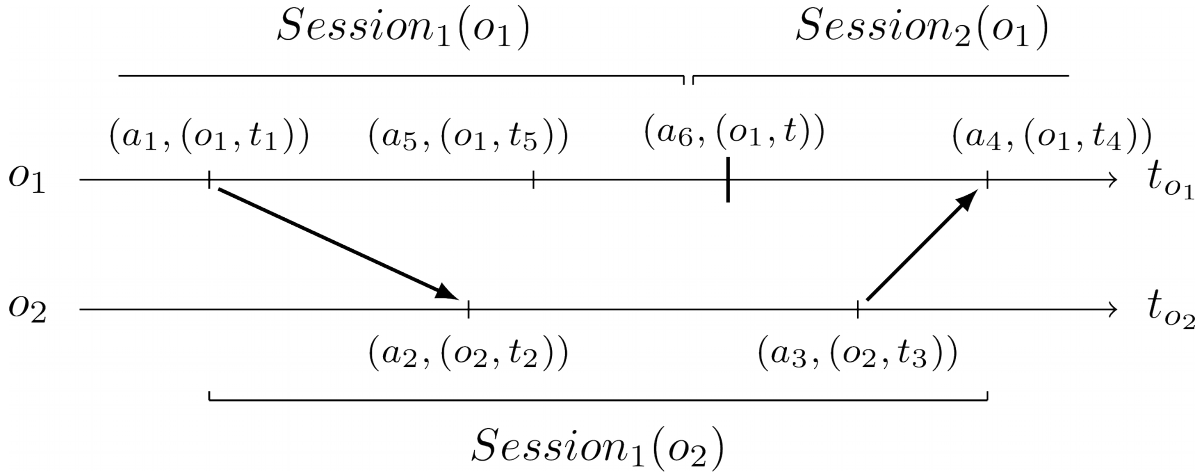
- **(Action, État) == (Action, (Objet, Horodatage))**
- $a \in \mathrm{ObjectActions}(o)$ avec $\mathrm{ObjectActions}(o) = \{ a_i \} \cup \boldsymbol{\{\emptyset\}}$

Contribution – Action Contextuelle

Dépendances Causales :

$$(\text{Action}, \underbrace{(\text{Objet, Horodatage})}_{\text{État}})$$

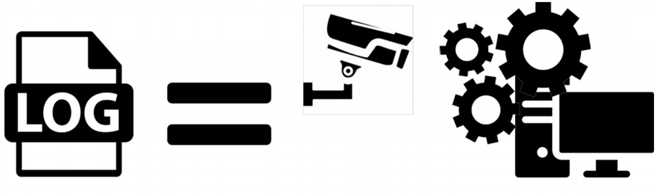
- Actions effectuées par les Objets (ex : send & recv message)
- États des Objets (c-à-d flux d'information)



Contribution – [Def] Événement Contextuel

Action Contextuelle (a , (o , t)) :

- Observée … ou pas
- Par une ou plusieurs sondes

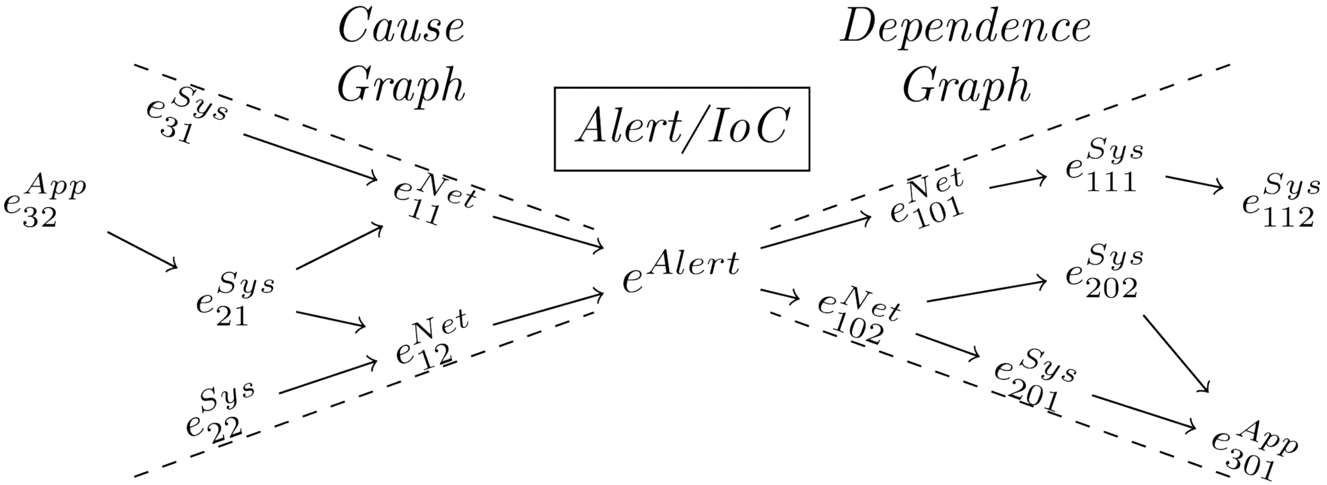


Événement Contextuel (e, o, t_e) ⇒ Observation d'une Action Contextuelle

$$\textbf{Obs}\big((a,(o,\ t_a\ ))\big) = \{(e_i\ ,\ o,\ t_{ei}\ )\} \cup \{(\varnothing,\ o,\ t_a\ )\}$$

Contribution – Cause & Dependence Graphs

Cause & Dependence Graphs can be computed for each layer depending on the use-case

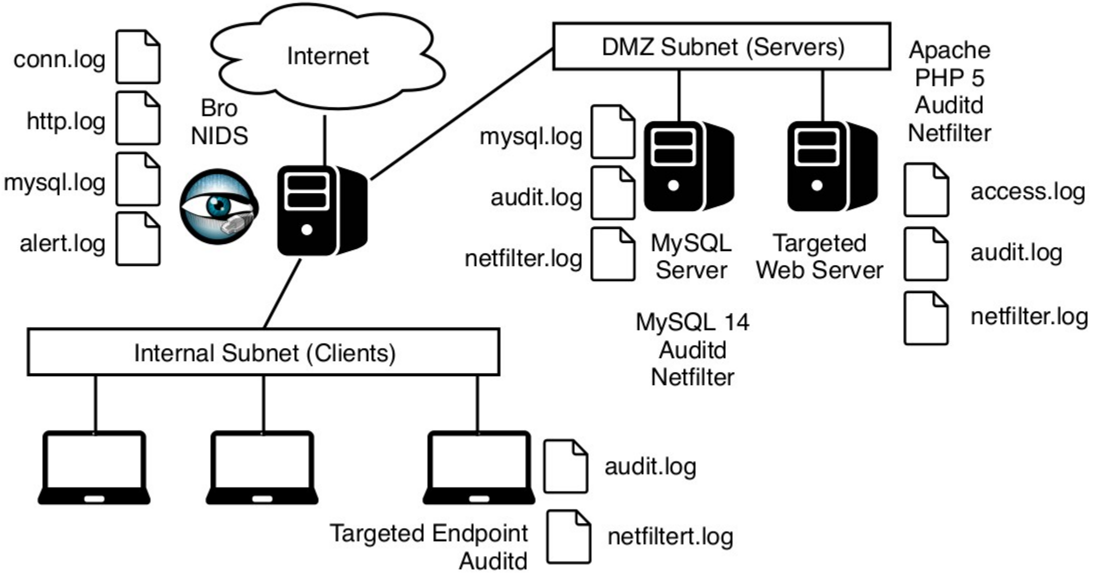


Evaluation – Challenges

- No datasets with heterogeneous events publicly available
- Need to create our own test environment
- Need to elaborate our own Attack Scenarios
- How to evaluate our implementation of the model ?

Evaluation – Test Environment

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.



Evaluation – Attack Scenarios & Data Generation

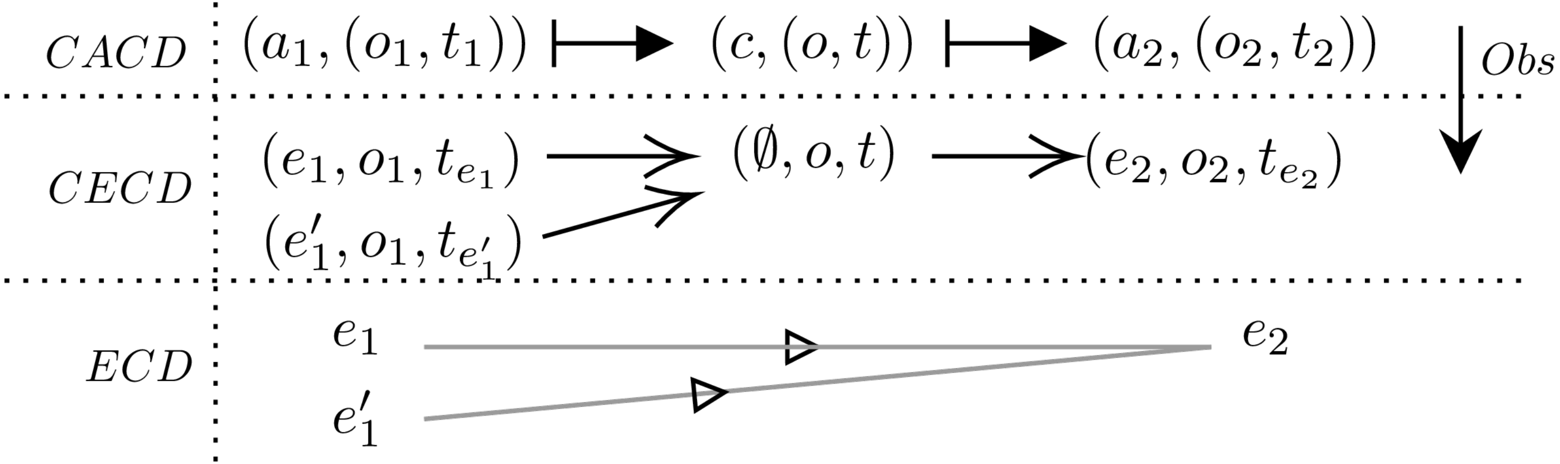
1. SQL Injection against a vulnerable PHP Script;

2. Trojan Software against an End Point machine (Ubuntu);

3. Command Injection Attack leveraging the ShellShock Bash Vulnerability (CVE-2014-6271) against Apache Web Server.

Evaluation – Two Point of Views

- Bottom-Up Approach => Current Practices in SIEM
- Top-Down Approach => Instrumentation



Future Works

- Applying the bottom-up to Windows use-cases

  Leveraging Syscalls API using ProcMon or LogMan

- Top-down approach by building all layers from Contextual Actions to Contextual Events.

  Towards a record and replay system to compute objects' states.

  ==> Leveraging Dynamic Information Flow Tracking (DIFT)

  Message passing system within the Network layer of the Kernel.

Conclusion

- Introduction d’un nouveau cadre théorique permettant de raisonner sur la notion de Dépendance Causale entre Événements Journalisés ;

- Bottom-up approach with a lightweight implementation.
  Building an approximation of the model from the logged events.

- Current methods and implementations allow the observation and recording of different subsets of actions.
  /!\ existing work only enables an approximation of the correct model.

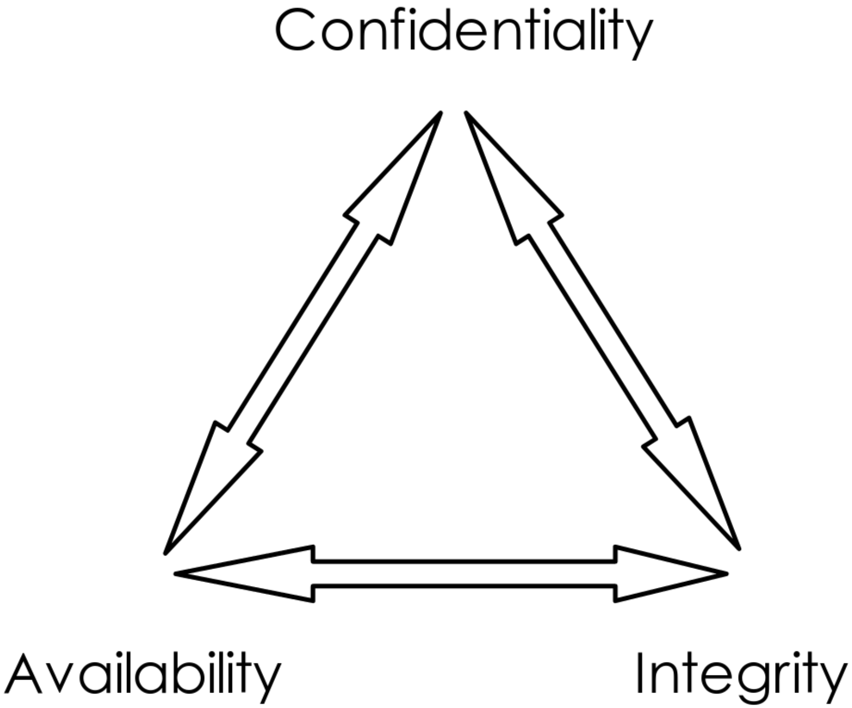
Merci de votre attention

Qui je suis

Charles XOSANAVONGSA

- Doctorant CIFRE en 3ème année
- CentraleSupélec – Équipe CIDRE (Rennes)
- Thales Six GTS France – Équipe Études Amonts (Palaiseau)
- Sécurité Informatique
    - Détection d’intrusion
    - Analyse de Logs
    - Corrélation d’alertes & d’événements

Context – Security Triangle



Context – You will be breached…

Attacker's Goal:

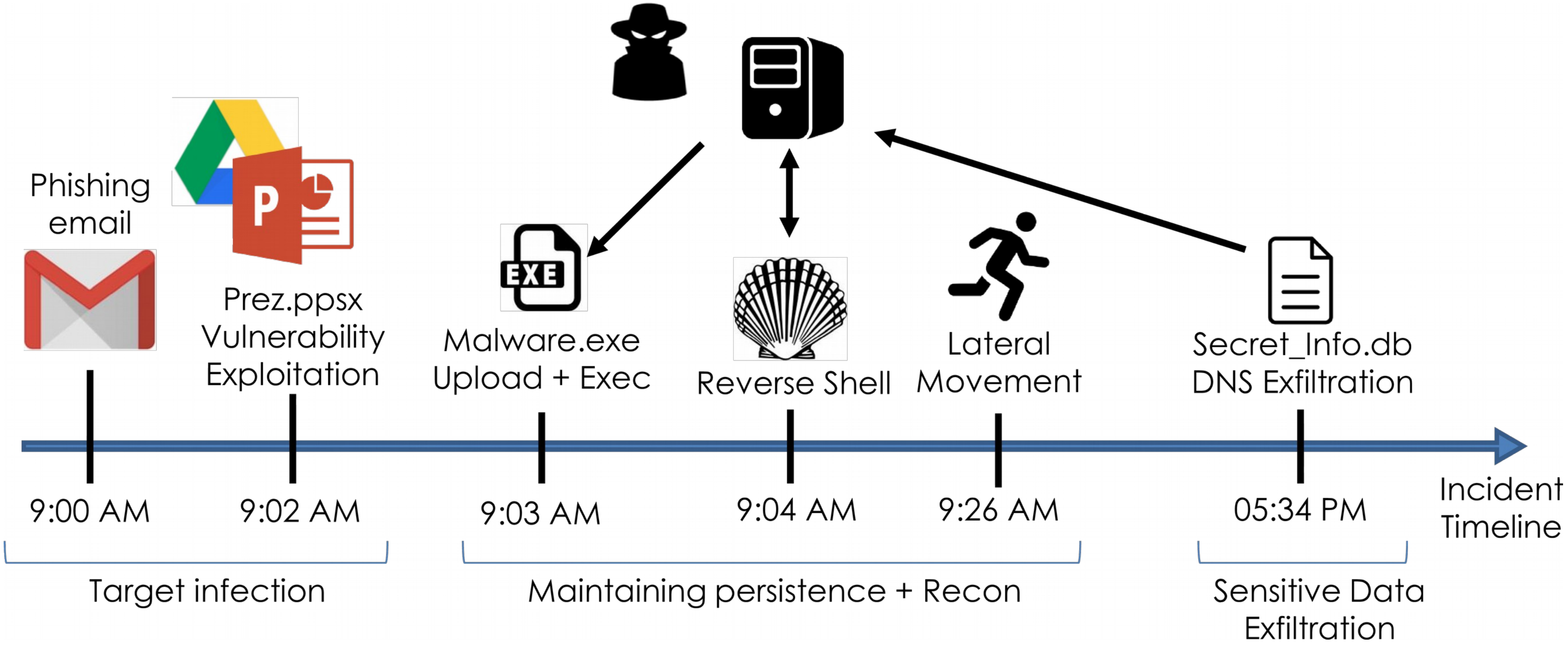
Gaining Foothold

Observation:

They eventually succeed

~~Prevention Mechanisms~~

Context – Attack Scenario Example



Phishing email

Prez.ppsx Vulnerability Exploitation

Malware.exe Upload + Exec

Reverse Shell

Lateral Movement

Secret_Info.db DNS Exfiltration

9:00 AM

9:02 AM

9:03 AM

9:04 AM

9:26 AM

05:34 PM

Incident Timeline

Target infection

Maintaining persistence + Recon

Sensitive Data Exfiltration

Related Work – Journey Overview

- Alert & Event Correlation
- Information Flow Tracking
- D’Ausbourg’s Causal Dependency among Objects’ States
- Lamport’s Happened-Before Relation among Processes’ Actions

Related Work – Explicit Alert & Event Correlation

- Attack Specification based
- Expression of Cause and Effects Relations between Events

- What we want:
  Discovering Attack Scenarios, without specifying them, through Heterogeneous Logs Analysis

Related Work – D’Ausbourg’s Model

Relation of **causal dependency**, leveraging **information flows** between **states** of the system :(o,t) → (o’,t’)

A state (o, t) is the value of an **object** o at a given time t.

No constraints on the definition of objects

- Program Variables
- Files, Sockets, Pipes, Memory, Processes…

Drawbacks :

- Does not take objects’ actions into account

```
a := 1
a := a + 1
a := 0
```

(a,1) → (a,2)

(a,2) ⌐→ (a,3)

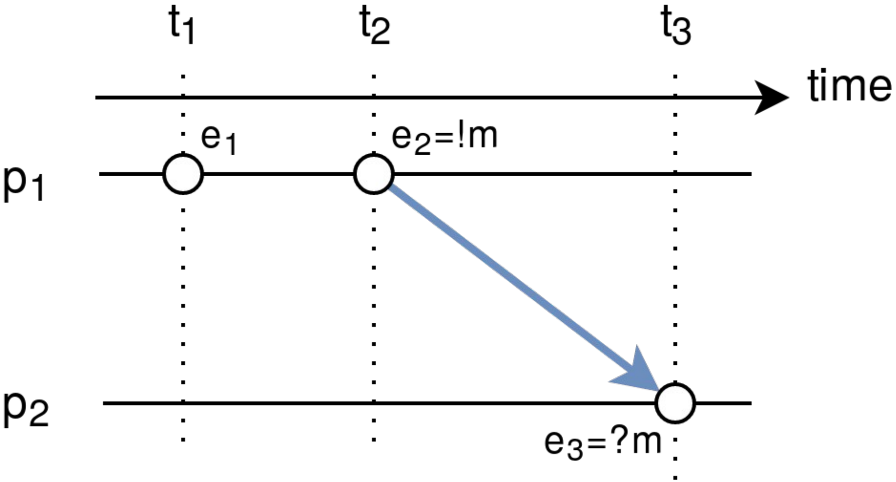
[d’Ausbourg, 1994] Implementing secure dependencies over a network by designing a distributed security subsystem.

Related Work – Lamport's Model

Temporal causality between **actions** performed by processes of a **distributed system**.
=> No **global clock** in the distributed system.

Partial order relation. $a \prec b$ if :

1. a is performed before b on the same process
2. $a = !m \;\&\&\; b = ?m$
3. $a \prec c \;\&\&\; c \prec b$

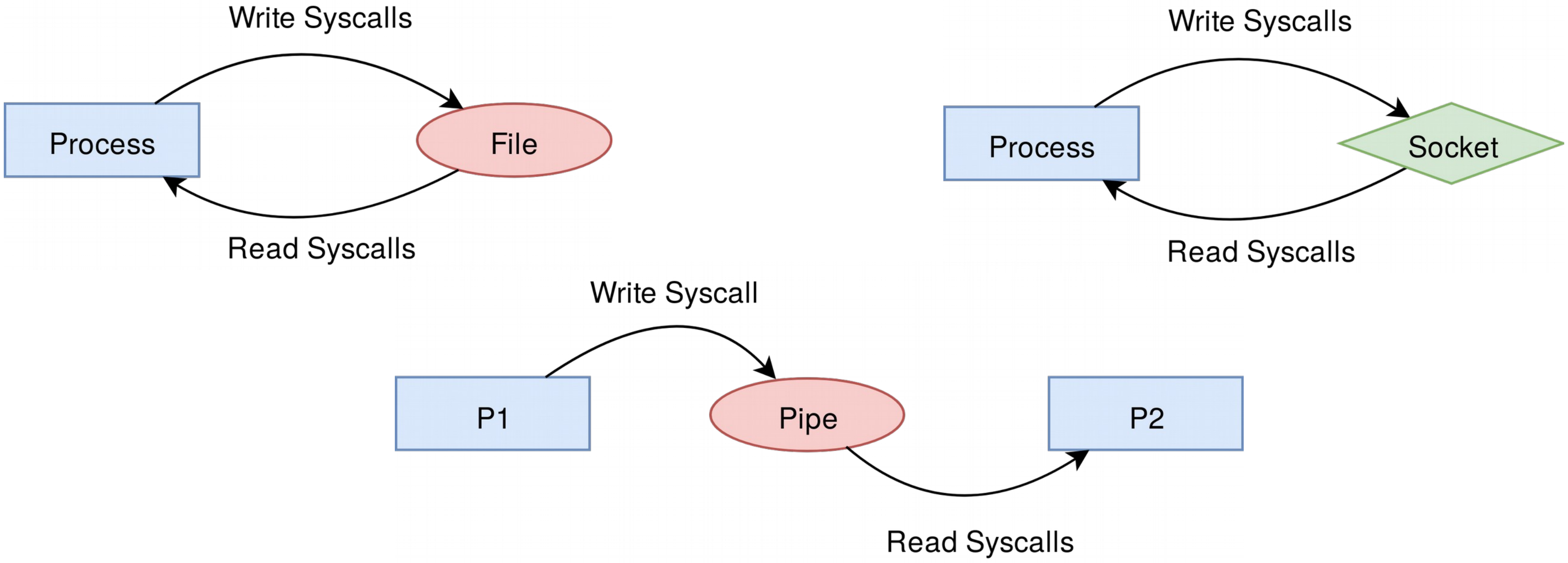


[Lamport, 1978] Time, clocks, and the ordering of events in a distributed system.

Related Work – Information Flow Tracking (IFT)

Monitoring Information Flows between the System’s Objects

Use-case: Security Policies for Confidentiality and Integrity



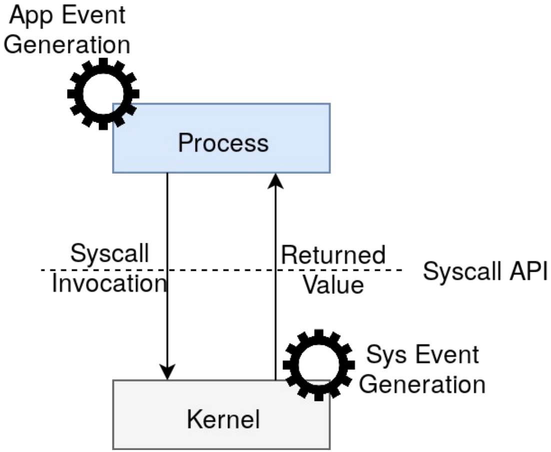
Related Work – Information Flow Tracking (IFT)

Allows to answer the following questions:

- Where does a given object come from?
- What are the objects that it influences?

Can be implemented in different Abstraction Layers:

- Dynamic Data Flow Analysis (CPU) [Kemerlis et al. 2012]
- Syscall (OS) [Georget et al. 2017]
- Application + OS [Muniswamy-Reddy et al. 2009]



[Kemerlis et al. 2012] libdft: Practical Dynamic Data Flow Tracking for Commodity Systems.

[Georget et al. 2017] Information Flow Tracking for Linux Handling Concurrent System Calls and Shared Memory.

[Muniswamy-Reddy et al. 2009] Layering in Provenance Systems.

Related Work – Dependency Explosion

Reasoning on Syscalls is too coarse-grained:

Over Approximation of Causal Dependencies

Given a Process, a given Syscall is supposed to be dependent on all the previous Syscalls invoked.

Prior work that proposes a solution:

- Binary Analysis in order to Identify the different Units inside processes and the information flows among them. [Lee et al. 2013]
  => Leverages LibC & Binary Instrumentation
- Rareness Score Computation in order to identify paths that might represent attack scenarios. [Liu et al. 2018]

[Lee et al. 2013] High Accuracy Attack Provenance via Binary-based Execution Partition.

[Liu et al. 2018] Towards a Timely Causality Analysis for Enterprise Security.

Related Work – D’Ausbourg’s Model

Relation of **causal dependency**, leveraging **information flows** between **states** of the system :(o,t) → (o’,t’)

A state (o, t) is the value of an **object** o at a given time t.

No constraints on the definition of objects

- Program Variables
- Files, Sockets, Pipes, Memory, Processes…

Drawbacks :

- Does not take objects’ actions into account

```
a := 1
a := a + 1
a := 0
```

(a,1) → (a,2)

(a,2) ⌐→ (a,3)

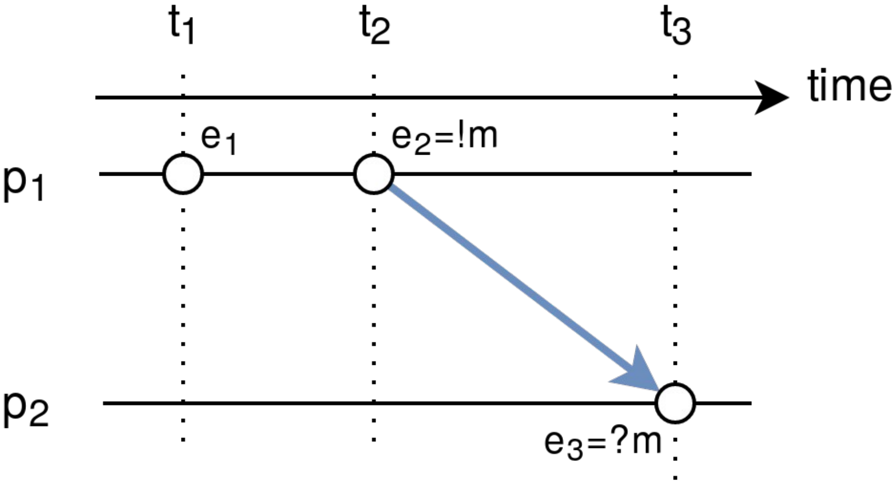
[d’Ausbourg, 1994] Implementing secure dependencies over a network by designing a distributed security subsystem.

Related Work – Lamport's Model

Temporal causality between **actions** performed by processes of a **distributed system.**
=> No **global clock** in the distributed system.

Partial order relation. a $\prec$ b if :

1. a is performed before b on the same process
2. a = !m && b = ?m
3. a $\prec$ c && c $\prec$ b

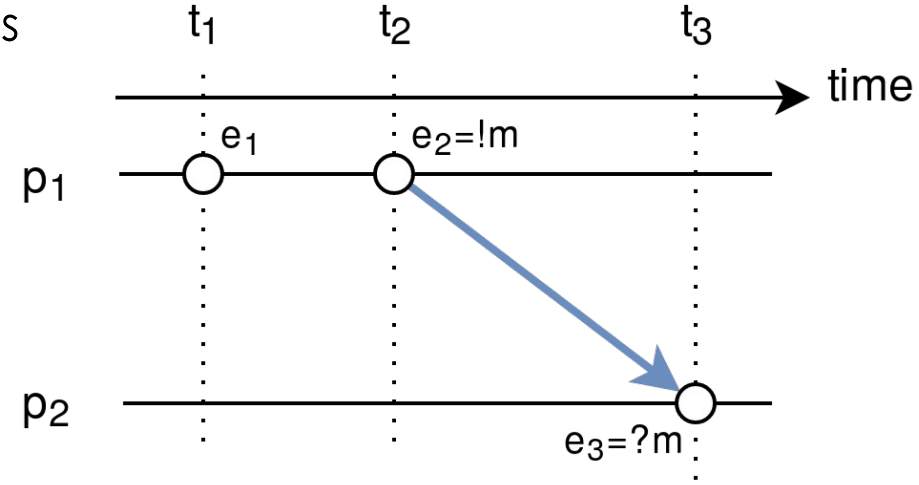


[Lamport, 1978] Time, clocks, and the ordering of events in a distributed system.

Related Work – Lamport’s Model

Drawbacks :

- Over approximation of Causal Dependencies
- Only deals with application level actions
  => No heterogeneous events



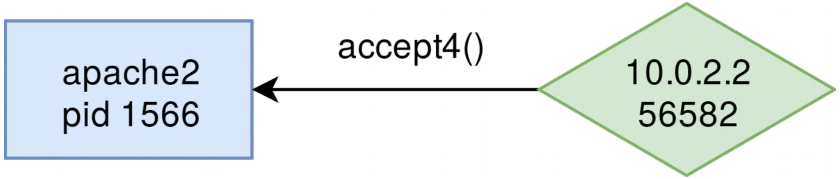
[Lamport, 1978] Time, clocks, and the ordering of events in a distributed system.

Evaluation – Bottom-Up Approach – Data Generation

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.

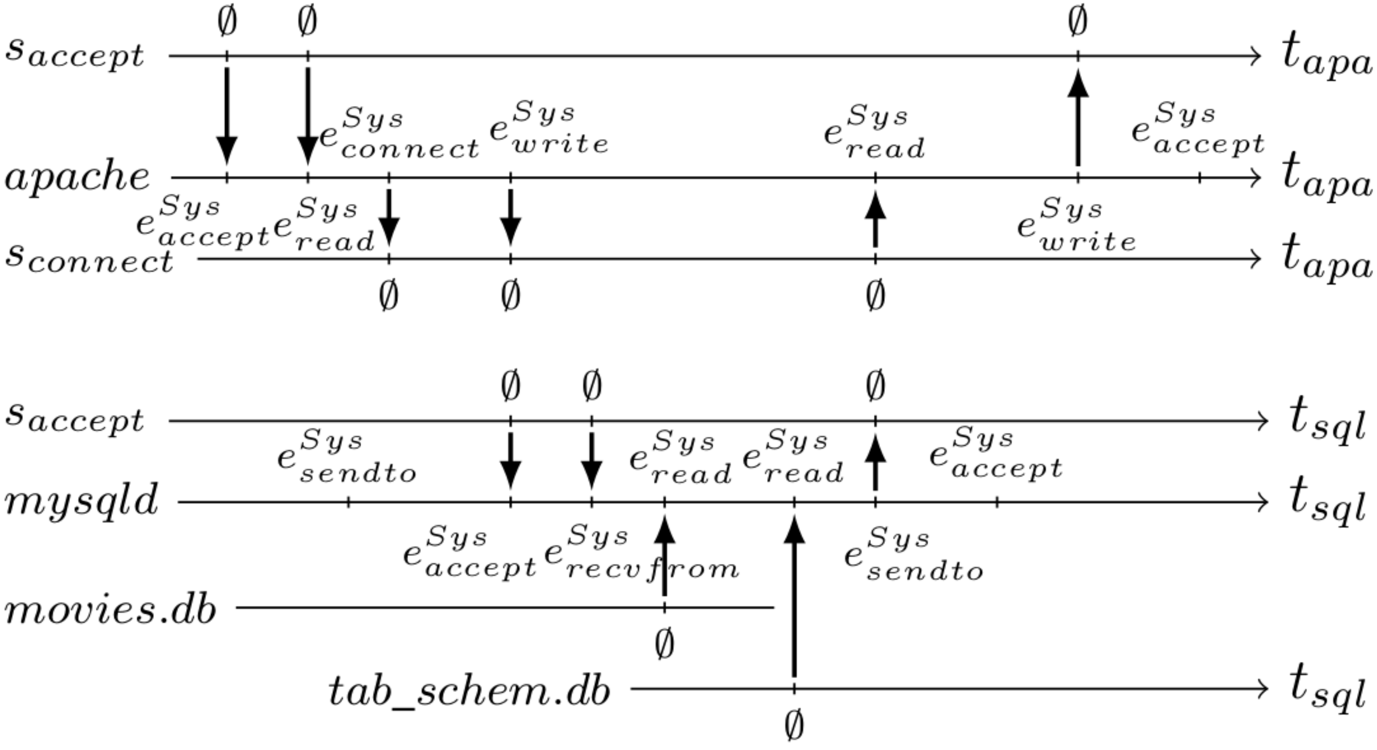
Syscall Netfilter PCAP Application

```
type=SYSCALL msg=audit(1541366508.539:47875): arch=c000003e syscall=288 success=yes
exit=10 a0=3 a1=7ffce59a1100 a2=7ffce59a10e0 a3=80000 items=0 ppid=1106 pid=1566
auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33
tty=(none) ses=4294967295 comm="apache2" exe="/usr/sbin/apache2" key=(null)
type=SOCKADDR msg=audit(1541366508.539:47875): saddr=0200DD060A0002020000000000000000
(saddr= (AF INET) 10.0.2.2 : 56582)
type=PROCTITLE msg=audit(1541366508.539:47875):
proctitle=2F7573722F7362696E2F61706163686532002D6B007374617274
(proctitle=/usr/sbin/apache2 -k start)
```



Evaluation – Illustration of Contextual Event

Syscall : 1 Active Object + 1 Passive Object + 1 Information Flow Action



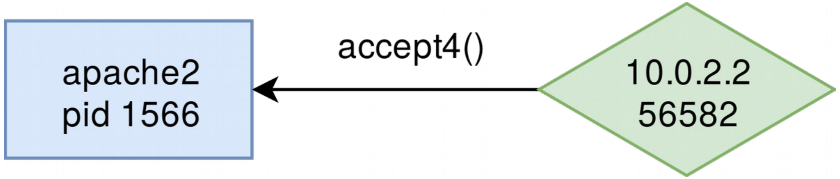
Evaluation – Bottom-Up Approach – Data Generation

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.

Syscall **Netfilter** PCAP Application

A connection is identified by a quadruplet
IP/Port Src & Dst

Bridging the gap between Syscalls and PCAP



Evaluation – Bottom-Up Approach – Data Generation

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.

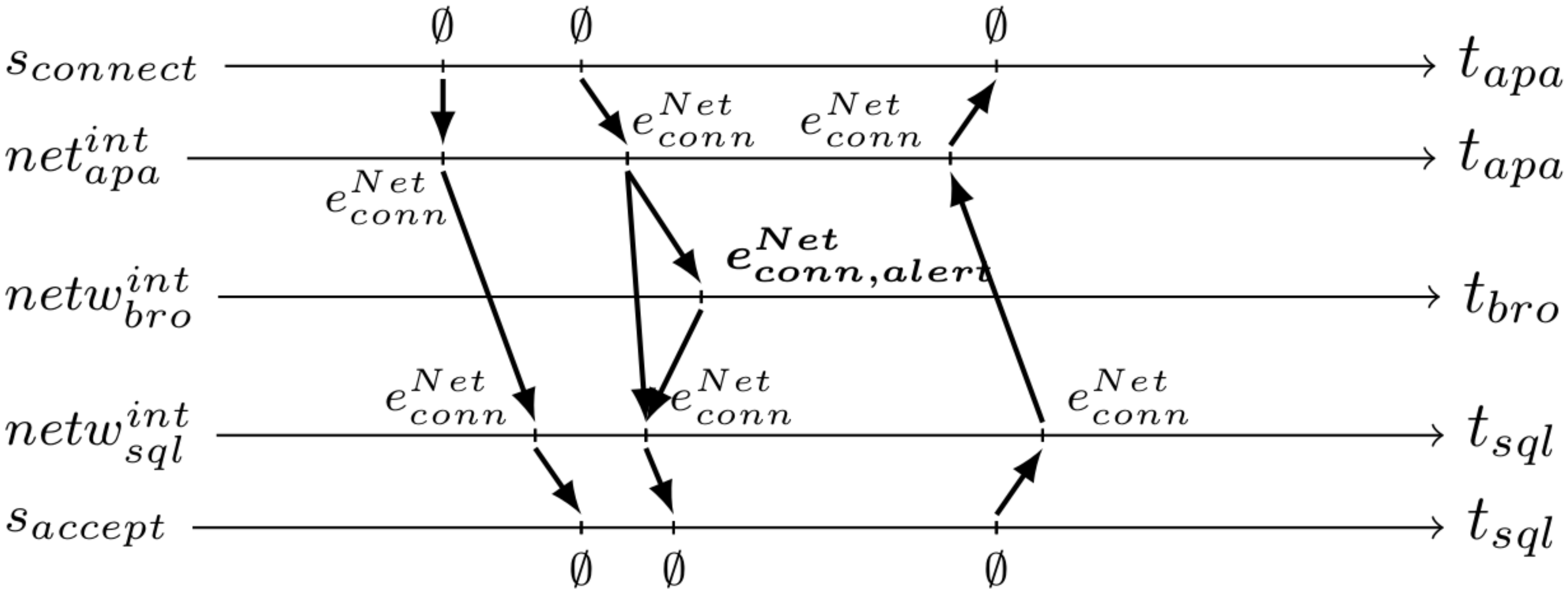
Syscall Netfilter **PCAP** Application

2 Passive Objects (Network Sockets) + Message Passing

```
2018-11-04T21:50:55.001600Z CgGkAp4P6ThQzD2Wg 192.168.1.2 48218 192.168.1.3 3306
tcp MySQL::Sqli SELECT * FROM movies WHERE title LIKE ‘%%’ UNION ALL SELECT
table schema,table name, null, null, null, null, null from information schema.tables;-
%’ SQLi Attempt : Suspect syntax detected. [‘Notice::ACTION LOG’]
```

Evaluation – Illustration of Contextual Event

PCAP : 2 Passive Objects (Network Sockets) + Message Passing



Evaluation – Bottom-Up Approach – Data Generation

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.

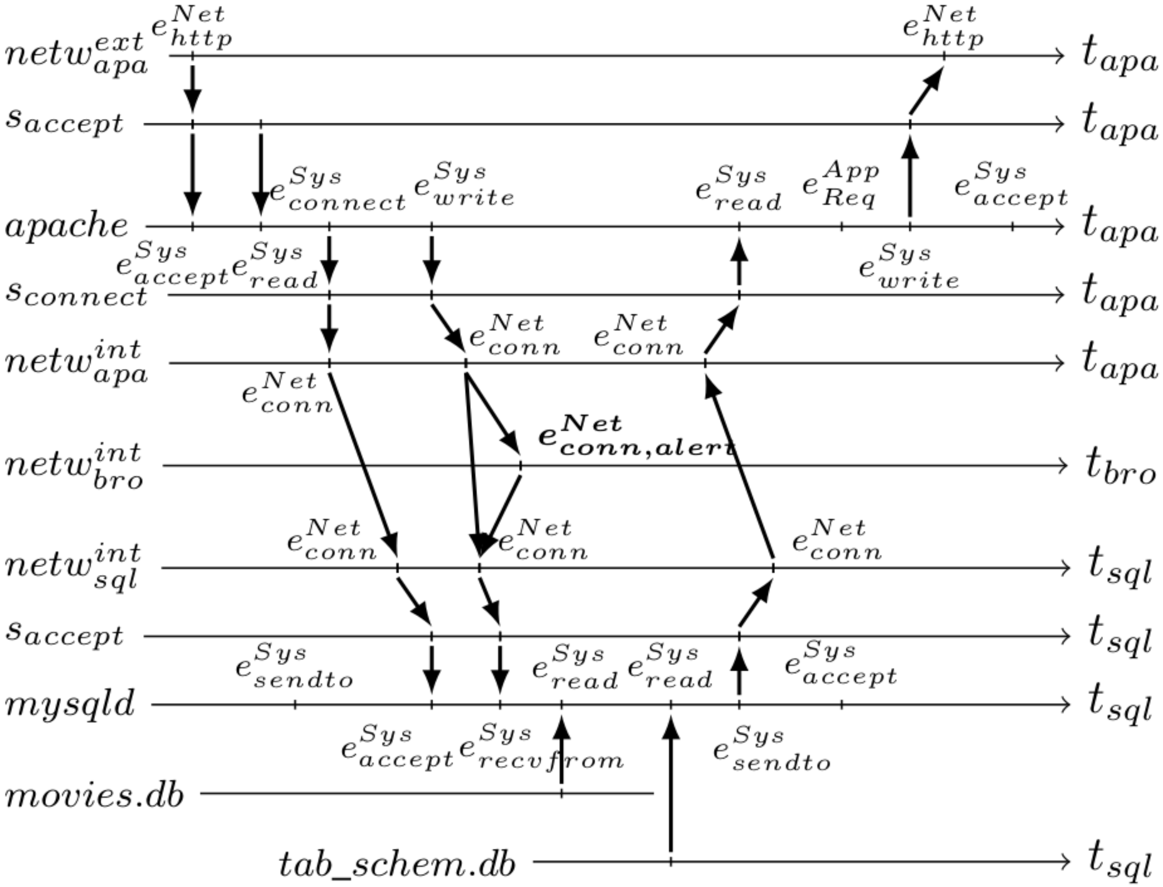
Syscall Netfilter PCAP **Application**

1 Active Object (The process with its PID)

```
[04/Nov/2018:21:50:54 +0000] 1566 10.0.2.15 80 10.0.2.2 56582
“POST /bWAPP/sqli 6.php HTTP/1.1” 200 6799 “http://10.0.2.15:80/bWAPP/sqli 6.php”
“Mozilla/5.0 (X11; Ubuntu; Linux x86 64; rv:61.0) Gecko/20100101 Firefox/61.0”
```

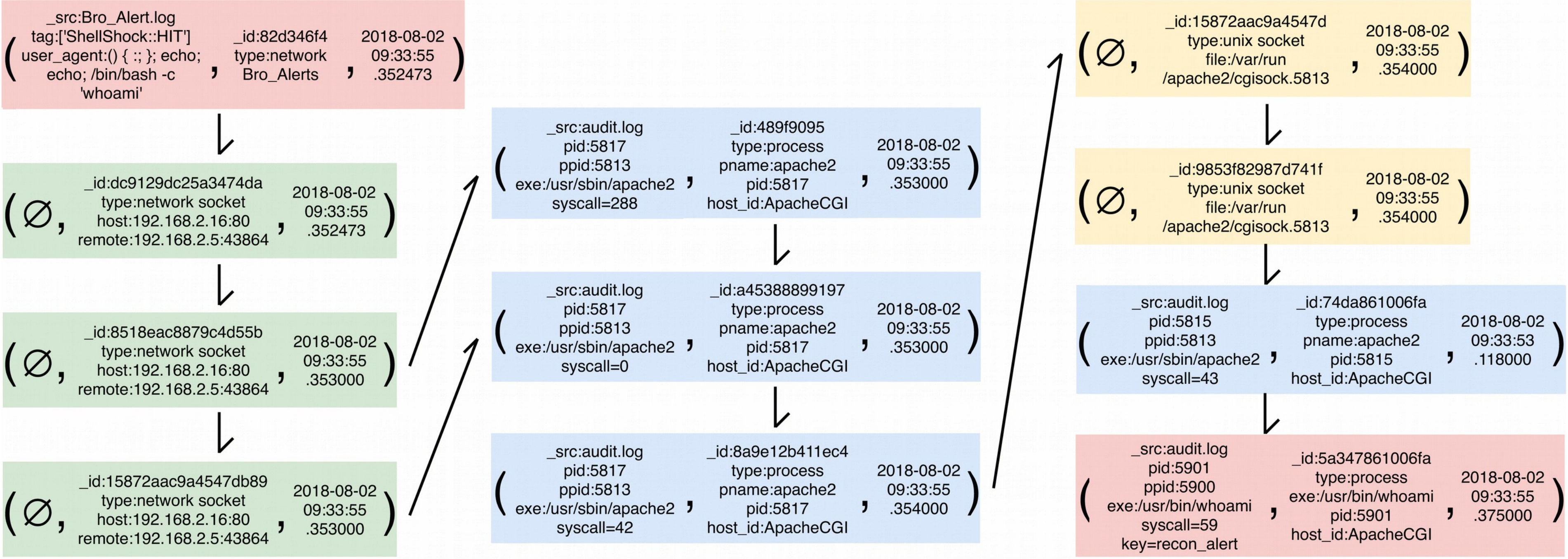
Evaluation – Illustration of Contextual Event

Putting it all together



Evaluation – ShellShock attack against Apache

Leveraging Logged Events Semantics to Compute an Approximation of the Contextual Event Causal Dependency Layer.



(_src:Bro_Alert.log
tag:['ShellShock::HIT']
user_agent:() { :; }; echo;
echo; /bin/bash -c
'whoami' , _id:82d346f4
type:network
Bro_Alerts , 2018-08-02
09:33:55
.352473)

(∅ , _id:dc9129dc25a3474da
type:network socket
host:192.168.2.16:80
remote:192.168.2.5:43864 , 2018-08-02
09:33:55
.352473)

(∅ , _id:8518eac8879c4d55b
type:network socket
host:192.168.2.16:80
remote:192.168.2.5:43864 , 2018-08-02
09:33:55
.353000)

(∅ , _id:15872aac9a4547db89
type:network socket
host:192.168.2.16:80
remote:192.168.2.5:43864 , 2018-08-02
09:33:55
.353000)

(_src:audit.log
pid:5817
ppid:5813
exe:/usr/sbin/apache2
syscall=288 , _id:489f9095
type:process
pname:apache2
pid:5817
host_id:ApacheCGI , 2018-08-02
09:33:55
.353000)

(_src:audit.log
pid:5817
ppid:5813
exe:/usr/sbin/apache2
syscall=0 , _id:a45388899197
type:process
pname:apache2
pid:5817
host_id:ApacheCGI , 2018-08-02
09:33:55
.353000)

(_src:audit.log
pid:5817
ppid:5813
exe:/usr/sbin/apache2
syscall=42 , _id:8a9e12b411ec4
type:process
pname:apache2
pid:5817
host_id:ApacheCGI , 2018-08-02
09:33:55
.354000)

(∅ , _id:15872aac9a4547d
type:unix socket
file:/var/run
/apache2/cgisock.5813 , 2018-08-02
09:33:55
.354000)

(∅ , _id:9853f82987d741f
type:unix socket
file:/var/run
/apache2/cgisock.5813 , 2018-08-02
09:33:55
.354000)

(_src:audit.log
pid:5815
ppid:5813
exe:/usr/sbin/apache2
syscall=43 , _id:74da861006fa
type:process
pname:apache2
pid:5815
host_id:ApacheCGI , 2018-08-02
09:33:53
.118000)

(_src:audit.log
pid:5901
ppid:5900
exe:/usr/bin/whoami
syscall=59
key=recon_alert , _id:5a347861006fa
type:process
exe:/usr/bin/whoami
pid:5901
host_id:ApacheCGI , 2018-08-02
09:33:55
.375000)