Shiqing Ma, Rutgers University

SYSTEM AUDIT

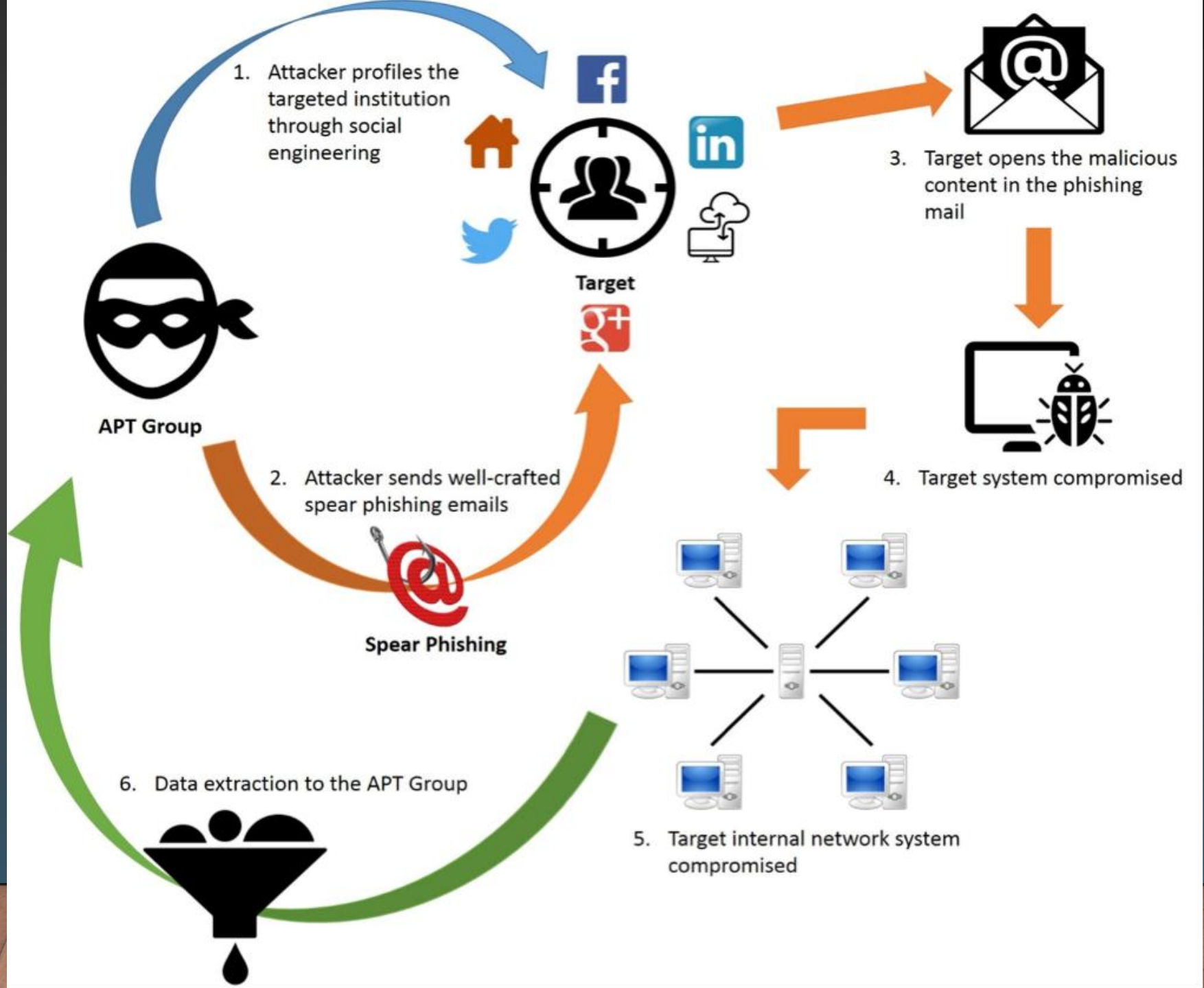
CS419: COMPUTER SECURITY

https://www.imperva.com/learn/application-security/apt-advanced-persistent-threat/
https://resources.infosecinstitute.com/category/enterprise/phishing/phishing-as-an-attack-vector/phishing-apts-advanced-persistent-threats/

STUXNET (2010)

- Olympic operation
  - Targets Iran
  - Involves five different countries – that is where the name is from
    - USA, Israel, Dutch, France, German
  - Attack is believed to last for 6 years
  - Eventually, bring Iran back to negotiation desk
  - Complex, novel, remote, evolution, hidden, highly targeted

ADVANCED PERSISTENT THREAT (APT)



APTS

The targets of these assaults, which are very carefully chosen and researched, typically include large enterprises or governmental networks. The consequences of such intrusions are vast, and include:

- Intellectual property theft (e.g., trade secrets or patents)
- Compromised sensitive information (e.g., employee and user private data)
- The sabotaging of critical organizational infrastructures (e.g., database deletion)
- Total site takeovers

APTS

APT attacks differ from traditional web application threats, in that:

- They're significantly more complex.
- They're not hit and run attacks — once a network is infiltrated, the perpetrator remains in order to attain as much information as possible.
- They're manually executed (not automated) against a specific mark and indiscriminately launched against a large pool of targets.
- They often aim to infiltrate an entire network, as opposed to one specific part.

APT ATTACKS

- Existing defense
  - Firewall, email filtering, end-point security product (e.g., anti-virus software), secure network configuration … …
  - Educated employees
- Have failed …
  - APTs are highly targeted: customized software, phishing emails … …

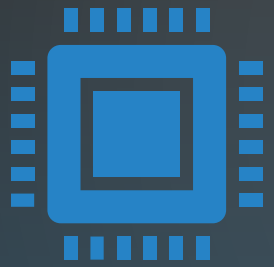
FORENSICS

"

Forensic analysis refers to a detailed investigation for detecting and documenting the course, reasons, culprits, and consequences of a security incident or violation of rules of the organization or state laws.

"

LOGGING



Logging is a simple but effective technique

Record the dynamic information (ex. syscalls) during system execution

Logging is widely used

For attack investigation – Forensic analysis

For failure diagnosis – Execution replay

EXISTING LOGGING FACILITIES

- Windows
  - Event Tracing for Windows (ETW)
  - Performance logging
  - Many 3rd party logging facilities
- *NIX
  - Shipped with Audit
  - 3rd party facilities

ANALYZE THE ATTACK

Identify a source of the attack

Understand the damage to the victim system

PROVENANCE BY AUDIT LOGGING

- Audit logging system
  - Records important system events during system execution
  - Is a default kernel module in Linux, FreeBSD and MacOS

PROVENANCE BY AUDIT LOGGING

- Audit logs show
  - Event information

```
type=SYSCALL (02/10/14 16:28:20.782) : syscall=open exit=3 a0=912e98
type=PATH : name=/etc/file1 inode=12345 mod=file,744
ppid=2537 pid=2557 comm=vim exe=/bin/vim
uid=tom gid=tom euid=tom
```

< Audit log entry for a file open event >

PROVENANCE BY AUDIT LOGGING

- Audit logs show
  - Event information
  - File information

```
type=SYSCALL (02/10/14 16:28:20.782) : syscall=open exit=3 a0=912e98
type=PATH : name=/etc/file1 inode=12345 mod=file,744
ppid=2537 pid=2557 comm=vim exe=/bin/vim
uid=tom gid=tom euid=tom
```

< Audit log entry for a file open event >

PROVENANCE BY AUDIT LOGGING

- Audit logs show
  - Event information
  - File information
  - Process information

```
type=SYSCALL (02/10/14 16:28:20.782) : syscall=open exit=3 a0=912e98
type=PATH : name=/etc/file1 inode=12345 mod=file,744
ppid=2537 pid=2557 comm=vim exe=/bin/vim
uid=tom gid=tom euid=tom
```

< Audit log entry for a file open event >

PROVENANCE BY AUDIT LOGGING

- Audit logs show
  - Event information
  - File information
  - Process information
  - User information

```
type=SYSCALL (02/10/14 16:28:20.782) : syscall=open exit=3 a0=912e98
type=PATH : name=/etc/file1 inode=12345 mod=file,744
ppid=2537 pid=2557 comm=vim exe=/bin/vim
uid=tom gid=tom euid=tom
```

< Audit log entry for a file open event >

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork **Malware**

Audit Log

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Backward analysis** - identify the source of an attack

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log

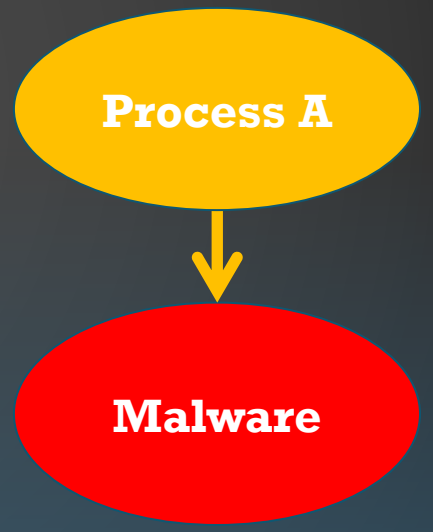
: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Backward analysis** - identify the source of an attack

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log



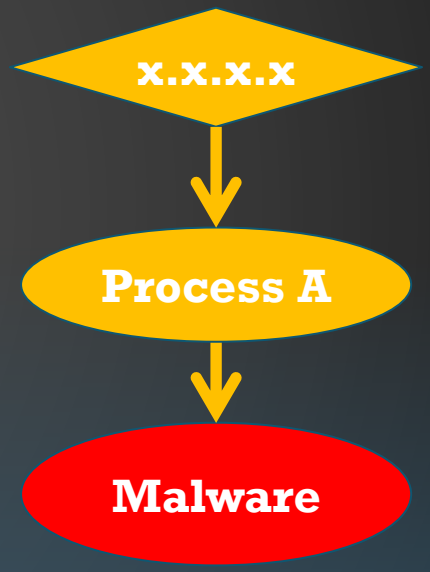
: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Backward analysis** - identify the source of an attack

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log



: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Forward analysis** - Understand damage to a system

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log

Malware

: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Forward analysis** - Understand damage to a system

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log

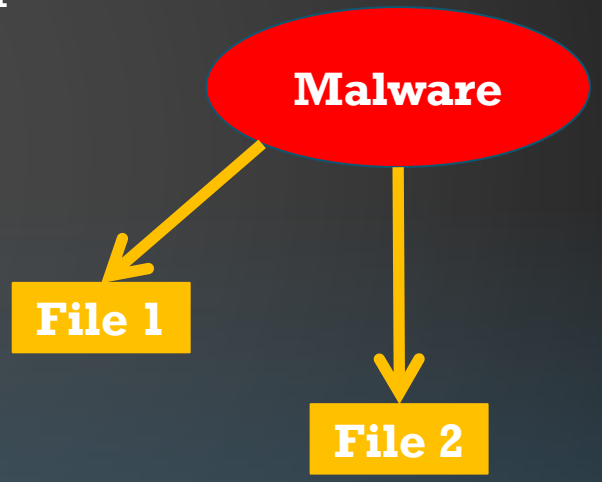
: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Analyze audit logs to generate a causal graph
  - **Forward analysis** - Understand damage to a system

(1)Proc_A recv <x.x.x.x>
(2)Proc_A fork Malware
(3)Malware write File1
(4)Malware write File2

Audit Log



: Process : File : Network Socket : Event

PROVENANCE BY AUDIT LOGGING

- Backward and forward analysis techniques
  - Sam T. King et al. "**Backtracking Intrusion**", In SOSP 2003
    - Beset Paper Award
  - Ashvin Goel et al. "**The Taser intrusion recovery system** ", In SOSP 2005

LIMITATIONS OF PREVIOUS WORKS

Dependence explosion

Generated causal graph is too large and contains many bogus information

Almost infeasible to be inspected by human

Sheer size of audit logs

Backtracker[SOSP'03] : 1.2GB/day

Taser[SOSP'05] : 1.9GB/day

Purdue Web Server : 3.7GB/day

Regular Client : 1.2GB/day

DEPENDENCE EXPLOSION - EXAMPLE

SOCIAL ENGINEERING ATTACK BY PHISHING E-MAIL

- Social engineeri

From: Chase Online [mailto:smrfs@chaseonline.com]
Sent: Wednesday, July 11, 2012 8:27 PM
Subject: Verification of Recent Activities Required

CHASE

URGENT: Verification of Recent Activities Required
Your Chase Bank Account

Dear Customer:

As part of our ongoing effort to protect your account and our relationship, we monitor your account for possible fraudulent activity. **We need to confirm that you or someone authorized to use your account made the following sign in error attempt on your Chase Bank account:**

1) Sign in Error Attempt was noticed and registered at 70.43.95.130. Chantilly, Virginia United State on or around 2012-07-11 at 05:01AM.

2) Sign in Error Attempt was noticed and registered at 68.170.136.81. Commack, New York United State on or around 2012-07-11 at 8:30PM.

3) Sign in Error Attempt was noticed and registered at 74.11.185.43 Delray Beach, Florida United State on or around 2012-07-11 at 8:20PM.

4) Sign in Error Attempt was noticed and registered at 68.46.148.86, Egg Harbor Township, New Jersey, United States on or around 2012-07-11 at 6:39AM.

Please click on the link below to sign in correctly to re-activate your online banking access:

www.chase.com

Your satisfaction is important to us, and we appreciate your prompt attention to this matter. If you already had the opportunity to discuss this matter with us, please disregard this message.

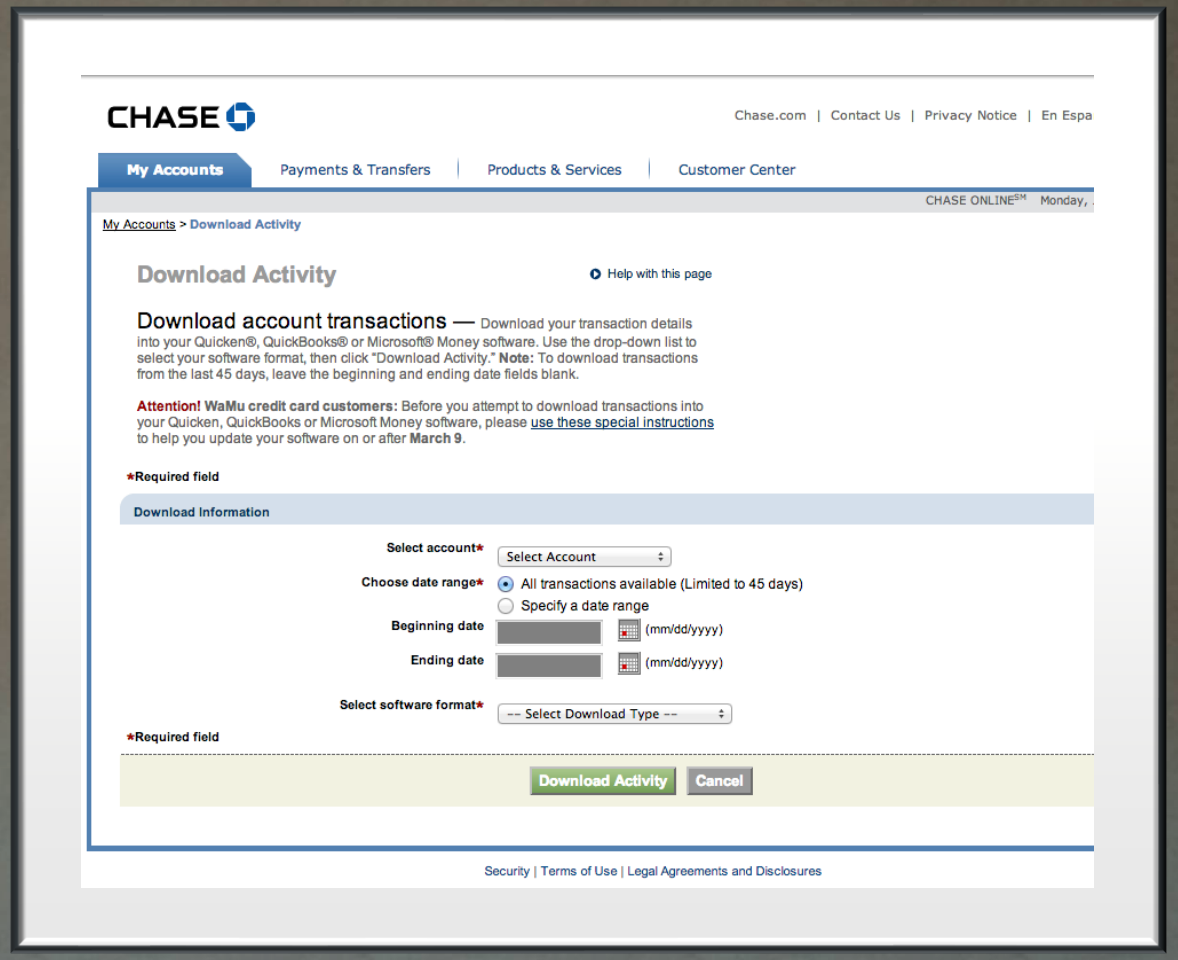
Thank you for being our customer.

Sincerely,

Christopher J. Palumbo
Senior Vice President
Chase Fraud Prevention

ATTACK PROVENANCE - EXAMPLE

- Social Engineering Attack
  - Phishing e-mail with a phishing link



28

CHASE

Chase.com | Contact Us | Privacy Notice | En Español | LOG OFF

My Accounts | Payments & Transfers | Products & Services | Customer Center

CHASE ONLINE℠ Monday, January 13, 2014

My Accounts > Download Activity

Download Activity

Help with this page

Download account transactions — Download your transaction details into your Quicken®, QuickBooks® or Microsoft® Money software. Use the drop-down list to select your software format, then click "Download Activity." **Note:** To download transactions from the last 45 days, leave the beginning and ending date fields blank.

Attention! WaMu credit card customers: Before you attempt to download transactions into your Quicken, QuickBooks or Microsoft Money software, please use these special instructions to help you update your software on or after **March 9**.

*Required field

Download Information

Select account* Select Account

Choose date range* All transactions available (Limited to 45 days)

Specify a date range

Beginning date (mm/dd/yyyy)

Ending date (mm/dd/yyyy)

Select software format* -- Select Download Type --

*Required field

Download Activity | Cancel

Security | Terms of Use | Legal Agreements and Disclosures

CHASE

Chase.com | Contact Us | Privacy Notice | En Español | LOG OFF

My Accounts | Payments & Transfers | Products & Services | Customer Center

CHASE ONLINE℠ Monday, January 13, 2014

My Accounts > Download Activity

Download Activity

Download account trans
into your Quicken®, QuickBooks® or
select your software format, then click
from the last 45 days, leave the begin

Attention! WaMu credit card custom
your Quicken, QuickBooks or Microso
to help you update your software on o

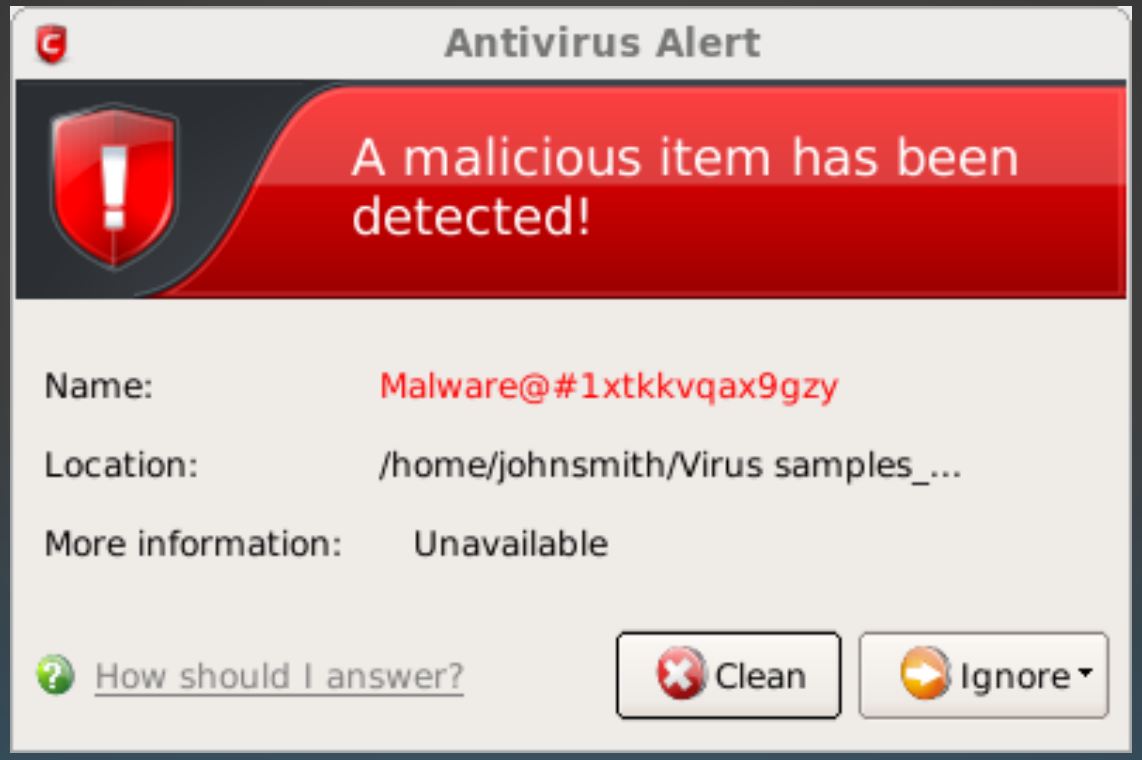
*Required field

Download Information

Selec

*Required field

[redacted]


Antivirus Alert
A malicious item has been detected!
Name:
Malware@#1xtkkvqax9gzy
Location:
/home/johnsmith/Virus samples_...
More information:
Unavailable
How should I answer?
Clean
Ignore

Malware

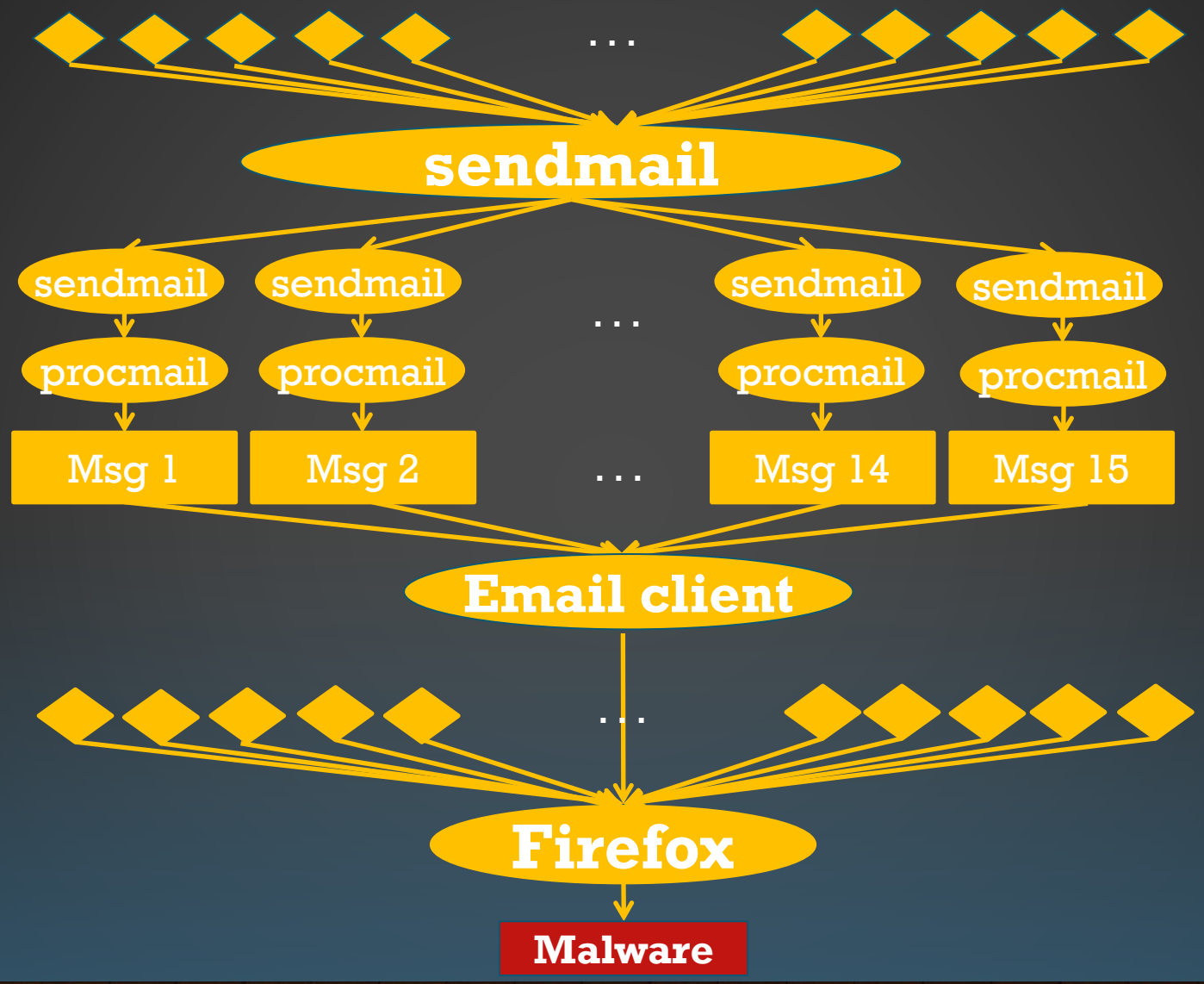

Firefox
Malware

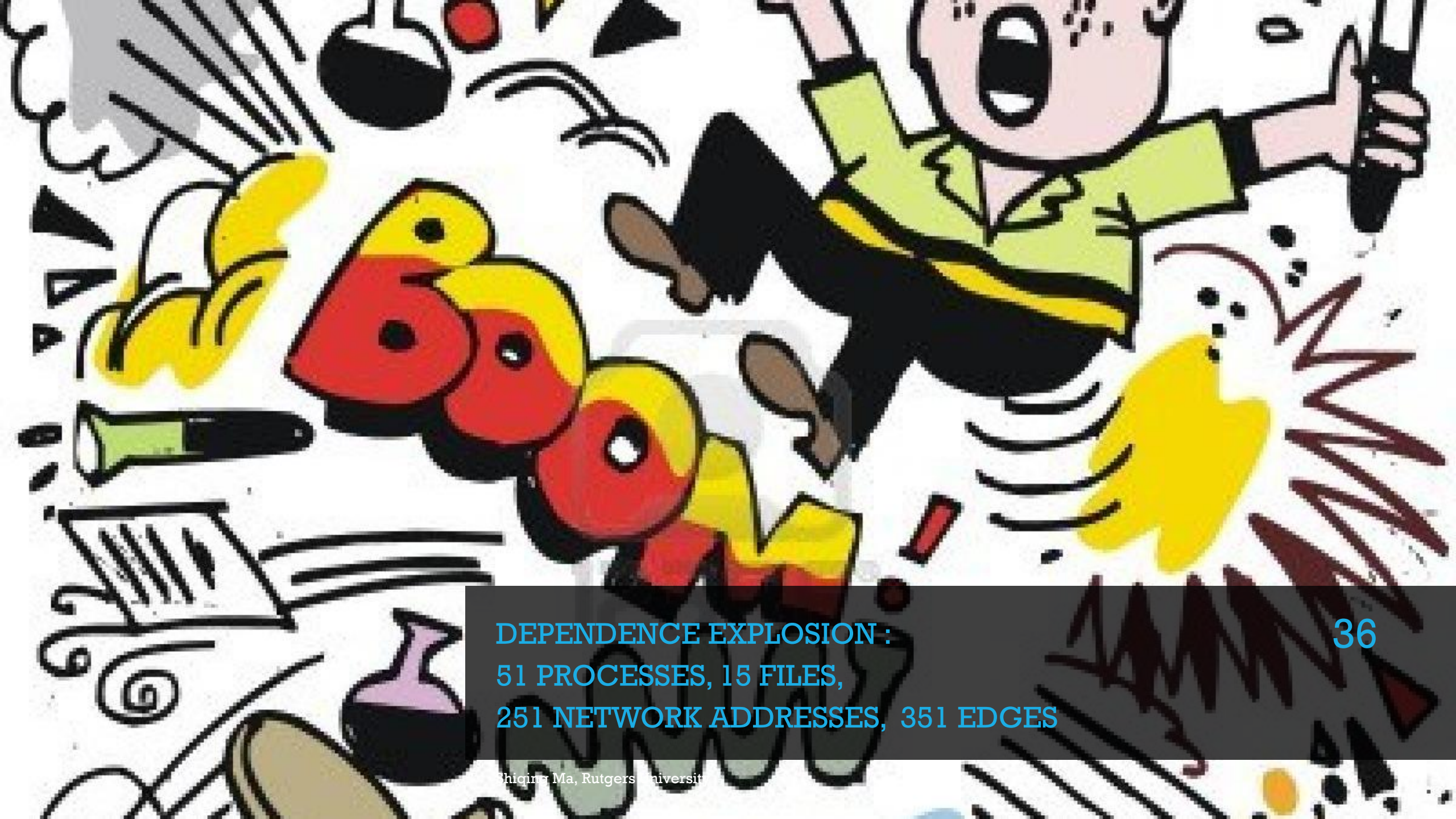
The user visited 11 web sites

Dependence explosion!!

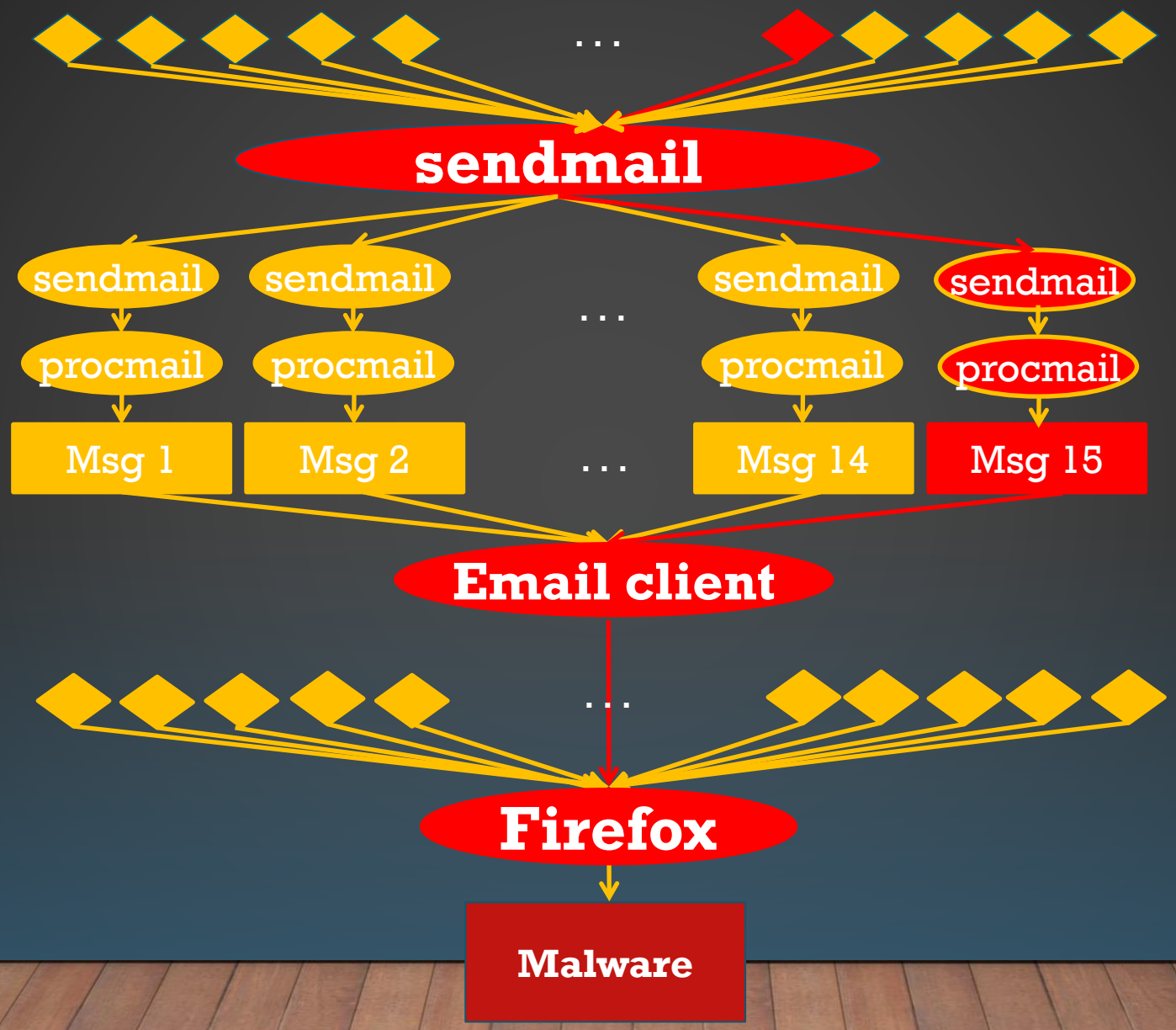
(229 IP Addresses)


Email client
...
Firefox
Malware


...
sendmail
sendmail
sendmail
sendmail
sendmail
...
procmail
procmail
procmail
procmail
Msg 1
Msg 2
...
Msg 14
Msg 15
Email client
...
Firefox
Malware

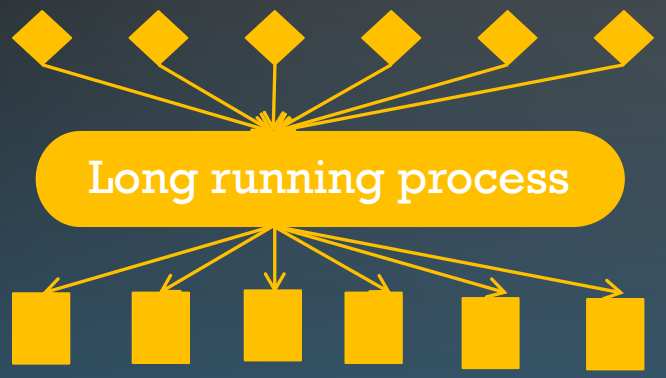


DEPENDENCE EXPLOSION :
51 PROCESSES, 15 FILES,
251 NETWORK ADDRESSES, 351 EDGES


...
sendmail
sendmail
sendmail
...
sendmail
sendmail
procmail
procmail
procmail
procmail
Msg 1
Msg 2
...
Msg 14
Msg 15
Email client
...
Firefox
Malware

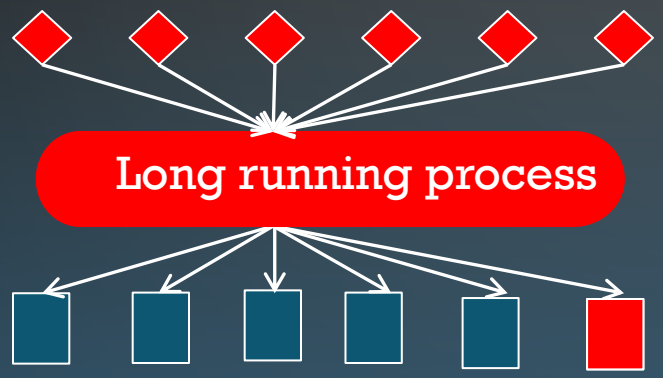
DEPENDENCE EXPLOSION – ROOT CAUSE

- Caused by **long-running processes**
  - Receive many inputs and produces many outputs



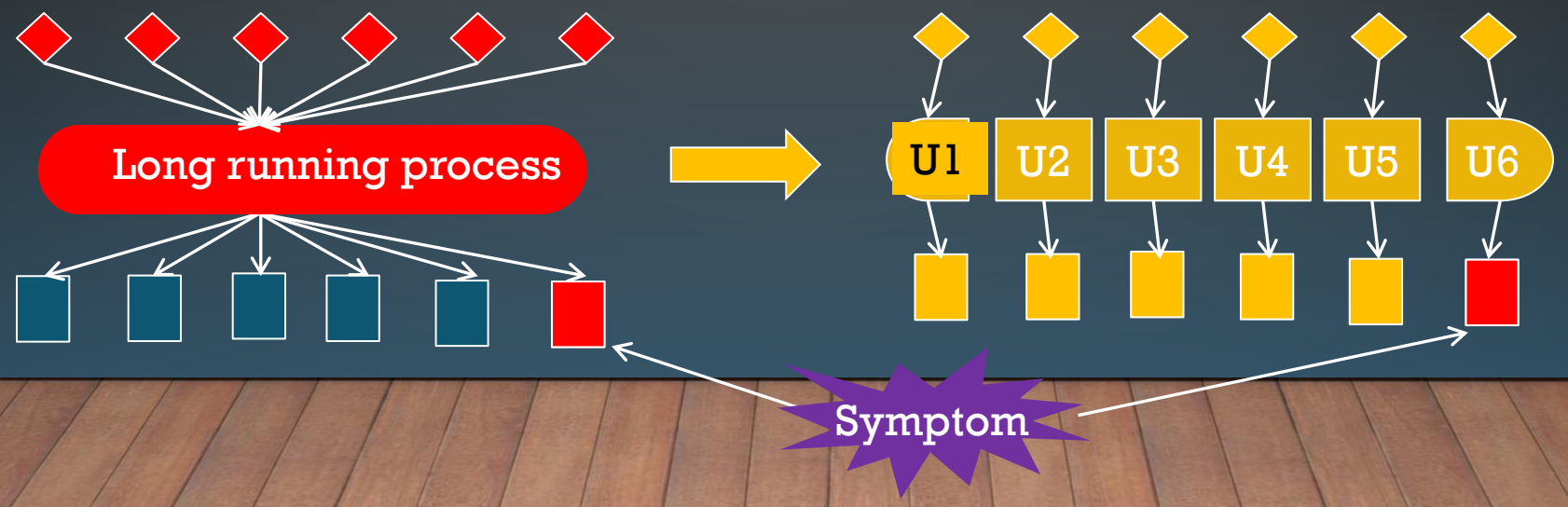
DEPENDENCE EXPLOSION – ROOT CAUSE

- Caused by long-running processes
  - Receive many inputs and produces many outputs
  - Any output is potentially related to all preceding inputs



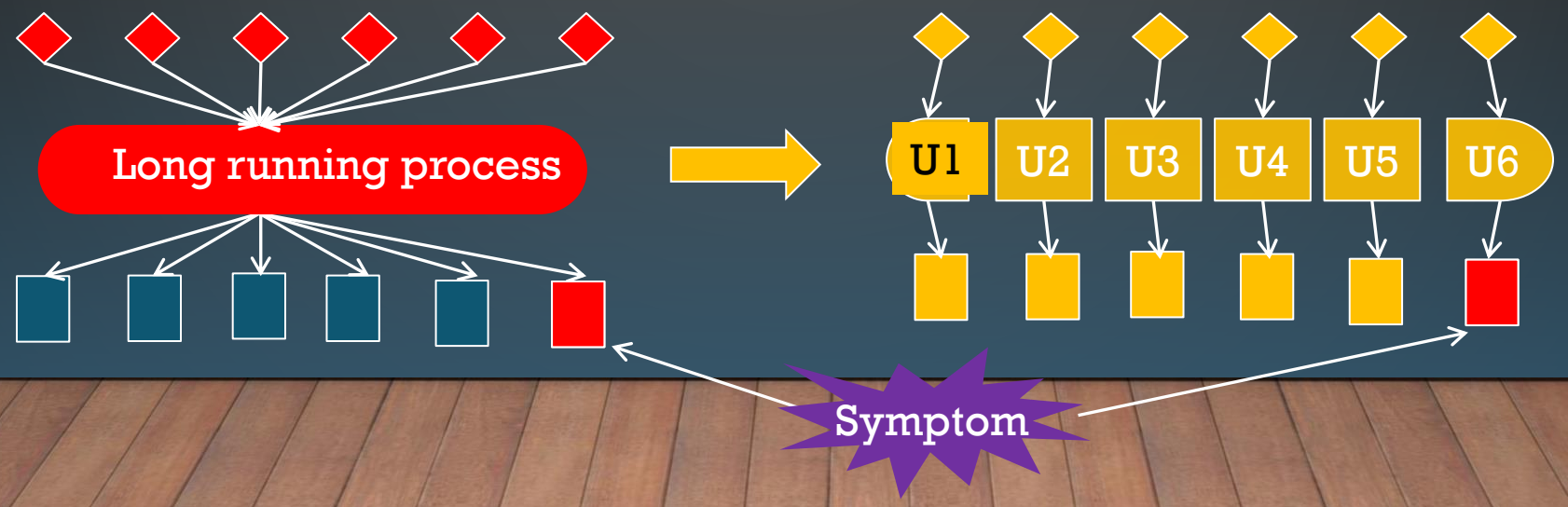
BEEP : BINARY-BASED EXECUTION PARTITION

- Finer-grained subject : Execution "UNIT"
  - Dynamically partition the execution of a process into **autonomous** execution segments
  - Units are not always independent
    - Detect causality between units



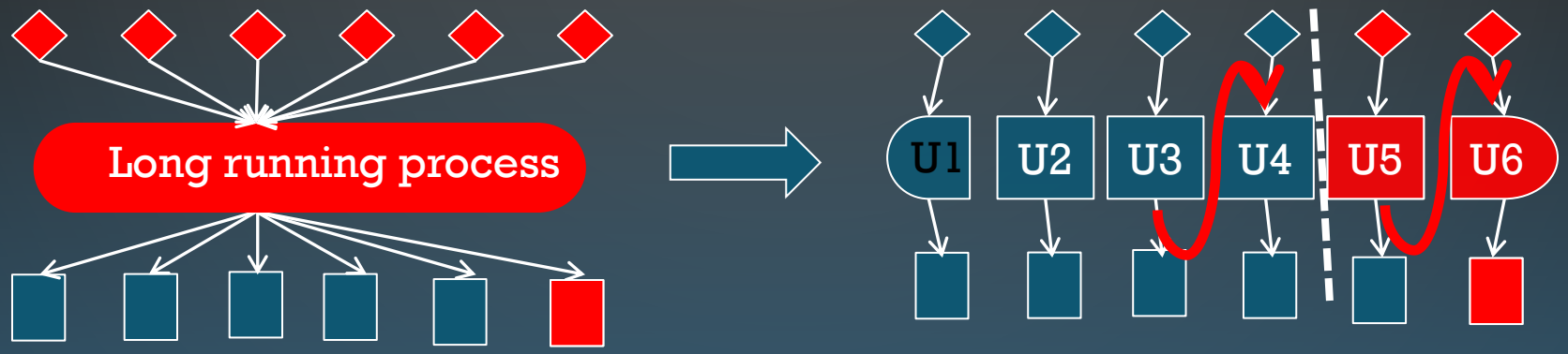
BEEP : BINARY-BASED EXECUTION PARTITION

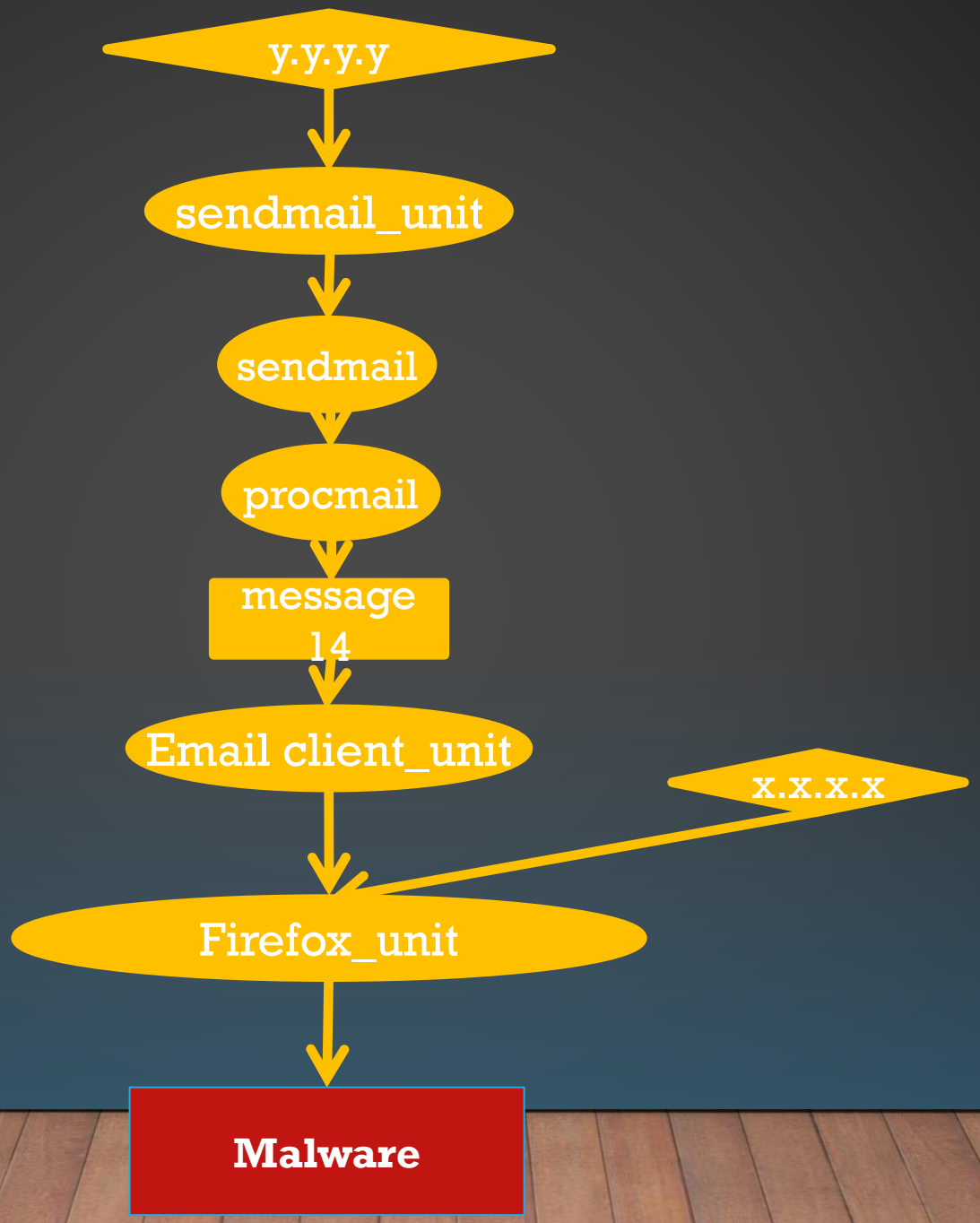
- Finer-grained subject : Execution "UNIT"
  - Dynamically partition the execution of a process into **autonomous** execution segments



BEEP : BINARY-BASED EXECUTION PARTITION

- Finer-grained subject : Execution "UNIT"
  - Dynamically partition the execution of a process into **autonomous** execution segments
  - Units are not always independent
    - Detect causality between units




y.y.y.y
sendmail_unit
sendmail
procmail
message 14
Email client_unit
x.x.x.x
Firefox_unit
Malware

Previous approaches [SOSP'03, SOSP'05] :
51 Processes, **15** Files,
251 Network addresses, **351** Edges

16.3 times smaller

BEEP :
10 Processes, **2** Files,
6 Network addresses, **23** Edges

OBSERVATIONS FROM 100 APPLICATIONS

	Category	Total Applications	Loop structured Applications
Servers	Web server	13	13
	Mail server	8	8
	FTP server	6	6
	SSHD server	2	2
	DNS server	9	9
	Database server	4	4
	Proxy server	2	2
	Media server	5	5
	Directory server	3	3
	Version control server	2	2
	Remote desktop server	2	2
UI Programs	Web browser	5	5
	E-mail client	5	5
	FTP client	5	5
	Office	2	2
	Text Editor	3	3
	Image tool	4	4
	Audio player	2	2
	Video player	4	4
	P2P program	6	6
	Messanger	2	2
	File manager	2	2
	Shell program	3	3

OBSERVATIONS FROM 100 APPLICATIONS

Characteristics of long running programs

- Driven by **external requests**
- Dominated by **event processing loops**

IDENTIFYING UNIT LOOPS

EVENT PROCESSING LOOP

```
void main(..) {
  while(..) {
    // Receive request
    // Process request
    // Send result
  }
}
```

IDENTIFYING UNIT LOOPS

- Event processing loop

```
void main(..) {
  while(..) {
    // Receive request
    // Process request
    // Send result
  }
}
```

Execution Unit :

Iteration of the event processing loop

IDENTIFYING UNIT LOOPS

- Detecting unit loops
  - High level loop

```
void main(..) {
[redacted]
  while(..)
    // Event process
[redacted]
```

IDENTIFYING UNIT LOOPS

- Detecting unit loops
  - High level loop

```
void main(..) {
  while(..)
    // Argument handling
  ..
  while(..)
    // Memory pool initialize
  ..
  while(..)
    // Event process

  while(..)
    // Free allocated memory
```

IDENTIFYING UNIT LOOPS

- Detecting unit loops
  - High level loop
  - Receive inputs and produce outputs

```
void main(..) {
  while(..)
    // Argument handling
  ..
  while(..)
    // Memory pool initialize
  ..
  while(..)
    // Event process
  ..
  while(..)
    // Free allocated memory
```

accept(…)
recvfrom(…)
read(…)
write(…)
sendto(…)

IDENTIFYING UNIT LOOPS

- Analyze a program binary
  - Construct **control flow graph** and **call graph** to identify loop heads and exits

Step 1 : Static Analysis

IDENTIFYING UNIT LOOPS

- Loop analysis
  - Track loop iterations and system calls to identify event processing loops

Step 1 : Static Analysis

Step 2 : Dynamic Analysis (Training Runs)

INTER-UNIT DEPENDENCES

- A unit alone may not correspond to a semantically independent sub-execution

INTER-UNIT DEPENDENCES

```
593   void *listener_thread(..) {
 ...
631     while(1) {
 ...
742       req=accept_func(..);
 ...
768       ap_queue_push(req);
 ...
798     } // while end
 ...
810   } // listener_thread end
```

```
820   void *worker_thread(..) {
 ...
842     while(!worker_may_exit) {
 ...
862       req=ap_queue_pop();
 ...
894       process_request(req);
 ...
899     } // while end
 ...
906   } // worker_thread end
```

<Apache-2.2.21> - **Multi-thread**

INTER-UNIT DEPENDENCES

```
593   void *listener_thread(..) {
...
631     while(1) {
...
742       req=accept_func(..);
...
768       ap_queue_push(req);
...
798     } // while end
...
810   } // listener_thread end
```

```
820   void *worker_thread(..) {
...
842     while(!worker_may_exit) {
...
862       req=ap_queue_pop();
...
894       process_request(req);
...
899     } // while end
...
906   } // worker_thread end
```

<Apache-2.2.21> - **Multi-thread**

INTER-UNIT DEPENDENCES

```
593   void *listener_thread(..) {
...
631     while(1) {
...
742       req=accept_func(..);
...
768       ap_queue_push(req);
...
798     } // while end
...
810   } // listener_thread end
```

```
820   void *worker_thread(..) {
...
842     while(!worker_may_exit) {
...
862       req=ap_queue_pop();
...
894       process_request(req);
...
899     } // while end
...
906   } // worker_thread end
```

<Apache-2.2.21> - **Multi-thread**

INTER-UNIT DEPENDENCES

```
593   void *listener_thread(..) {
...
631     while(1) {
...
742       req=accept_func(..);
...
768       ap_queue_push(req);
...
798     } // while end
...
810   } // listener_thread end
```

```
820   void *worker_thread(..) {
...
842     while(!worker_may_exit) {
...
862       req=ap_queue_pop();
...
894       process_request(req);
...
899     } // while end
...
906   } // worker_thread end
```

<Apache-2.2.21> - **Multi-thread**

INTER-UNIT DEPENDENCES

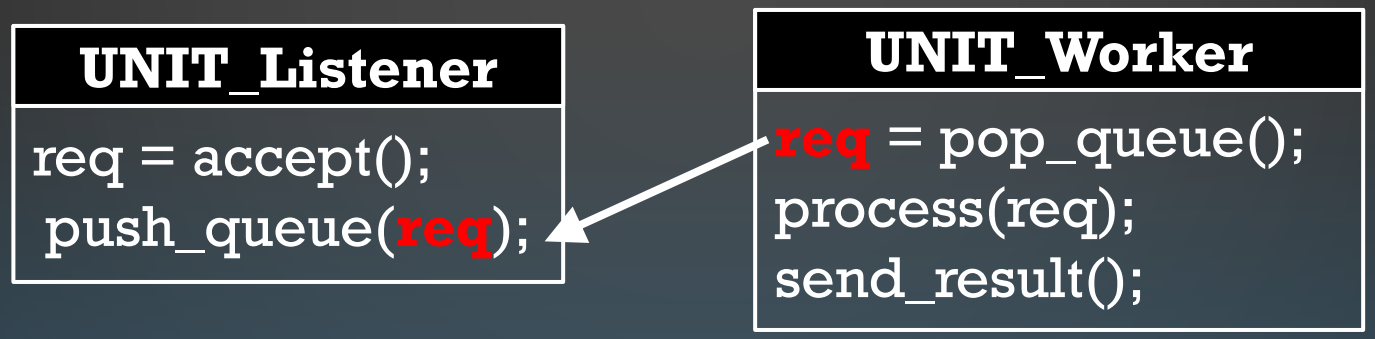
```
593   void *listener_thread(..) {
...
631     while(1) {
...
742       req=accept_func(..);
...
768       ap_queue_push(req);
...
798     } // while end
...
810   } // listener_thread end
```

```
820   void *worker_thread(..) {
...
842     while(!worker_may_exit) {
...
862       req=ap_queue_pop();
...
894       process_request(req);
...
899     } // while end
...
906   } // worker_thread end
```

<Apache-2.2.21> - **Multi-thread**

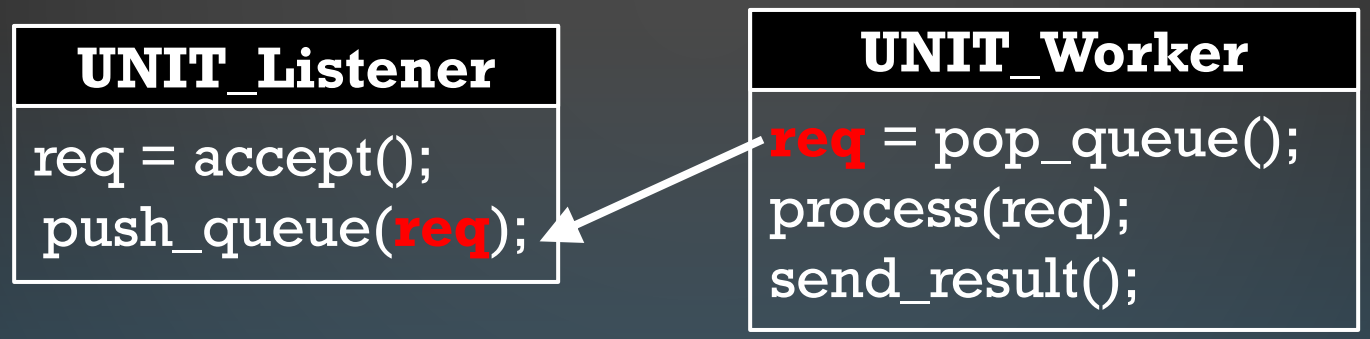
INTER-UNIT DEPENDENCES

- A unit alone may not correspond to a semantically independent sub-execution
  - A few units together compose an autonomous sub-execution



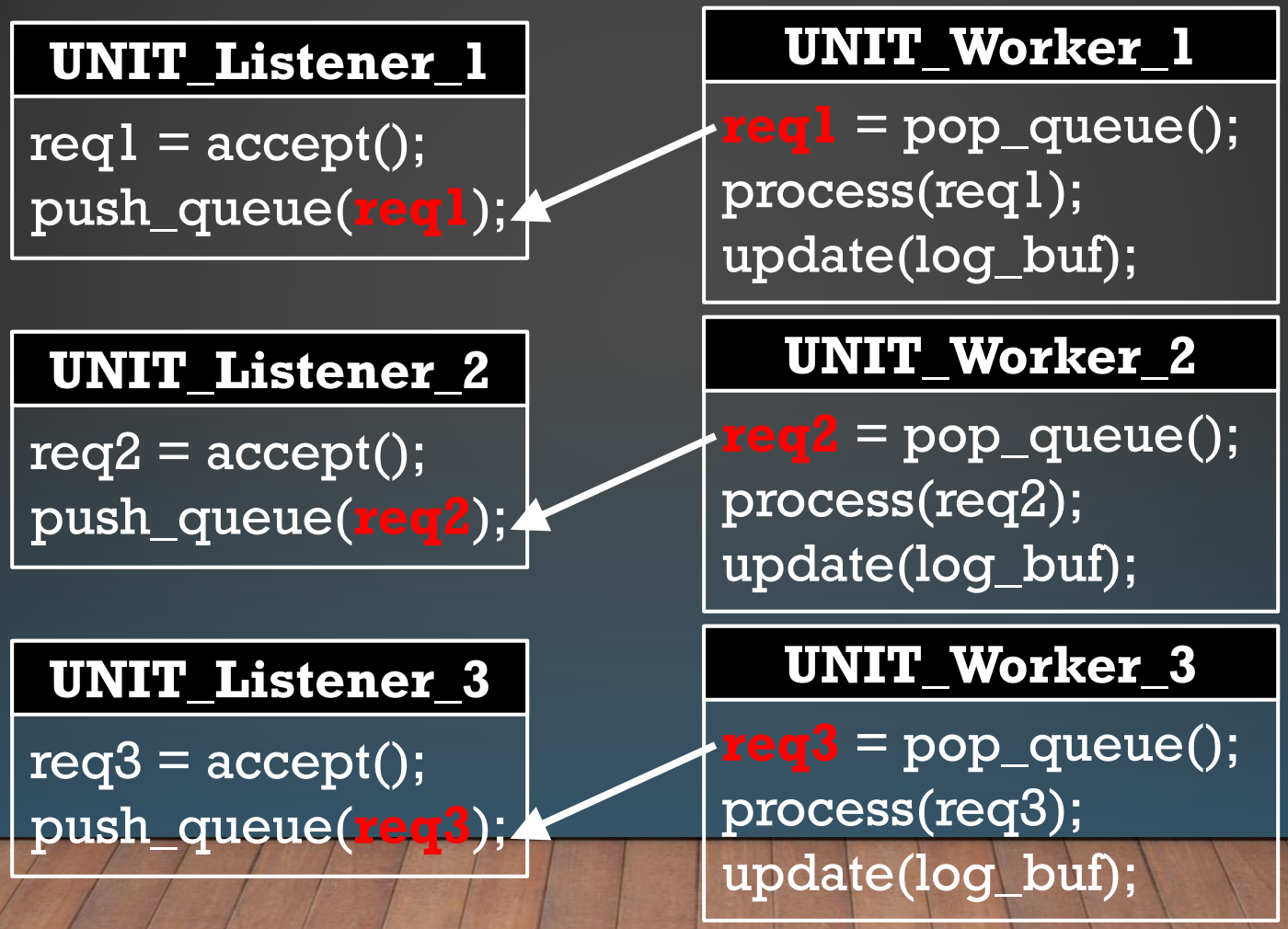
INTER-UNIT DEPENDENCES

- A unit alone may not correspond to a semantically independent sub-execution
  - A few units together compose an autonomous sub-execution



  - Dependences through memory object
    - Ex) queue – enqueue, dequeue

WORKFLOW DEPENDENCE

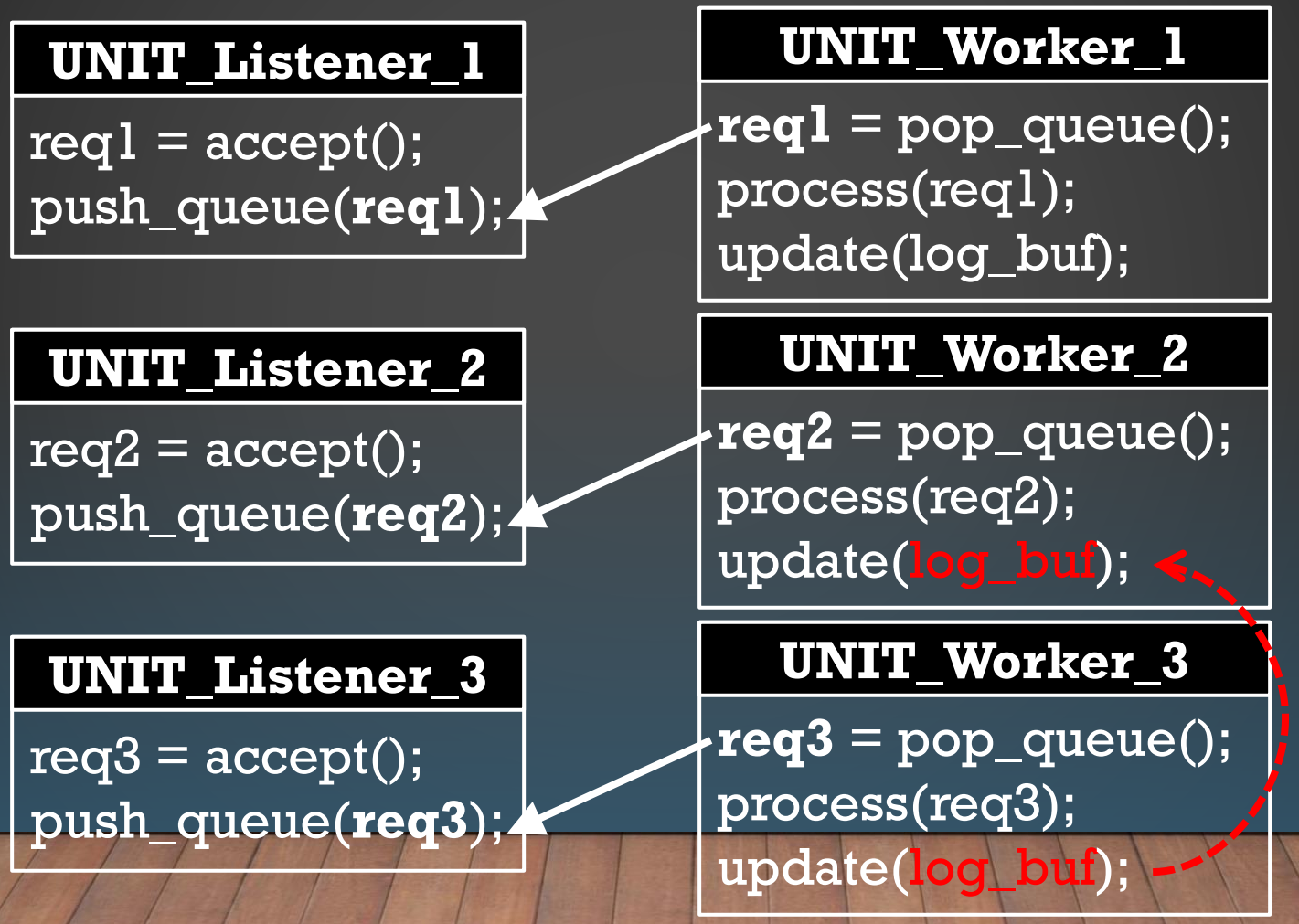


WORKFLOW DEPENDENCE

Workflow Dependences

- Represent high-level program workflow
- Detect semantically relevant units

WORKFLOW DEPENDENCE



LOW LEVEL (BOGUS) DEPENDENCE

Low Level Dependences

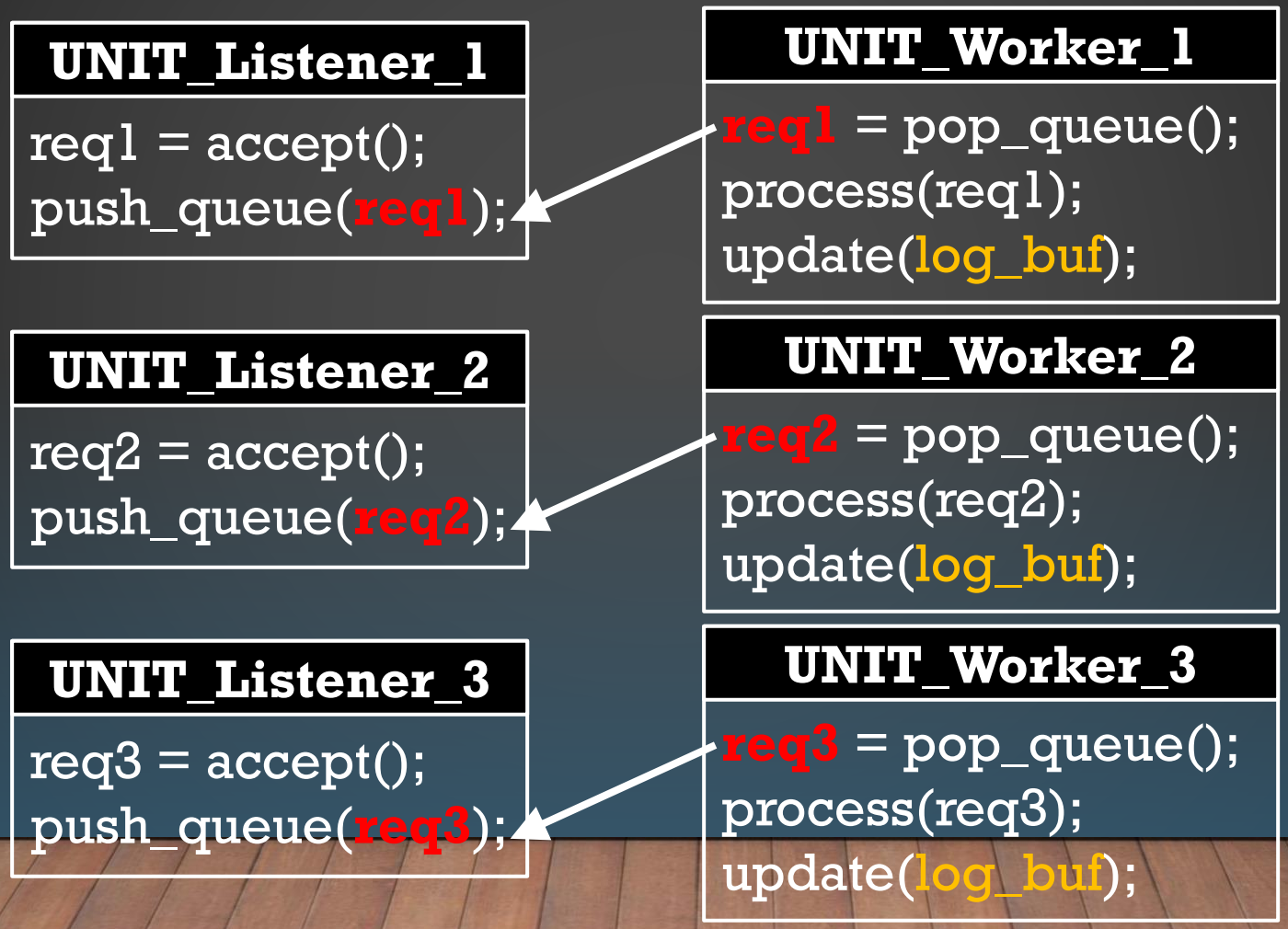
- Not part of the workflow
- Caused by low level behavior

INTER-UNIT DEPENDENCES

Characteristic of workflow objects

- Never shared by units from the same loop

WORKFLOW DEPENDENCE



INTER-UNIT DEPENDENCES

Step 1 : Static Analysis

Step 2 : Dynamic Analysis (Training Runs)

INSTRUMENTATION

Step 1 : Static Analysis

Step 2 : Dynamic Analysis (Training Runs)

Step 3 : Instrumentation

Step 1 : Static Analysis

Step 2 : Dynamic Analysis (Training Runs)

Step 3 : Instrumentation

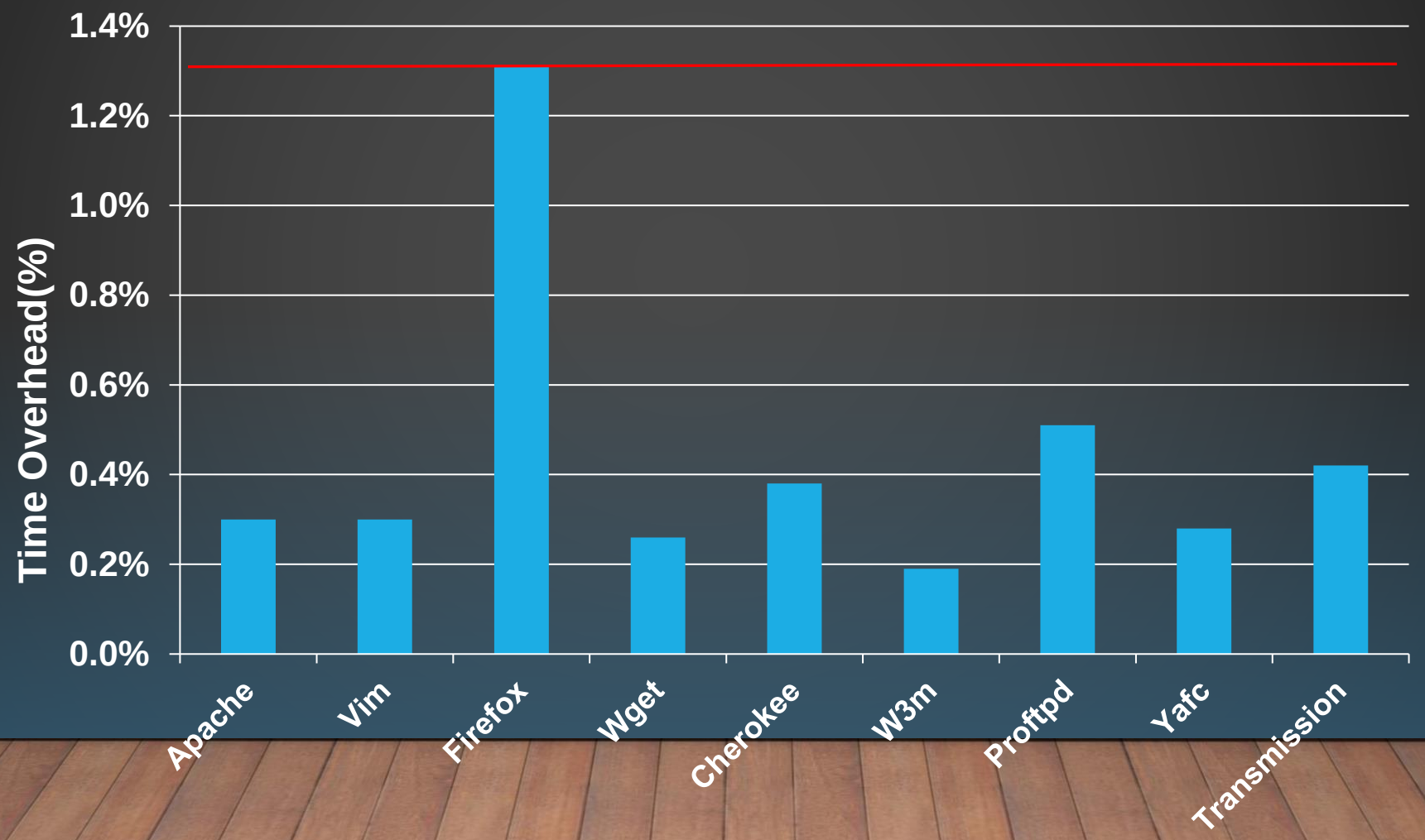
Step 4 : Audit Logging

Step 5 : Log Analysis

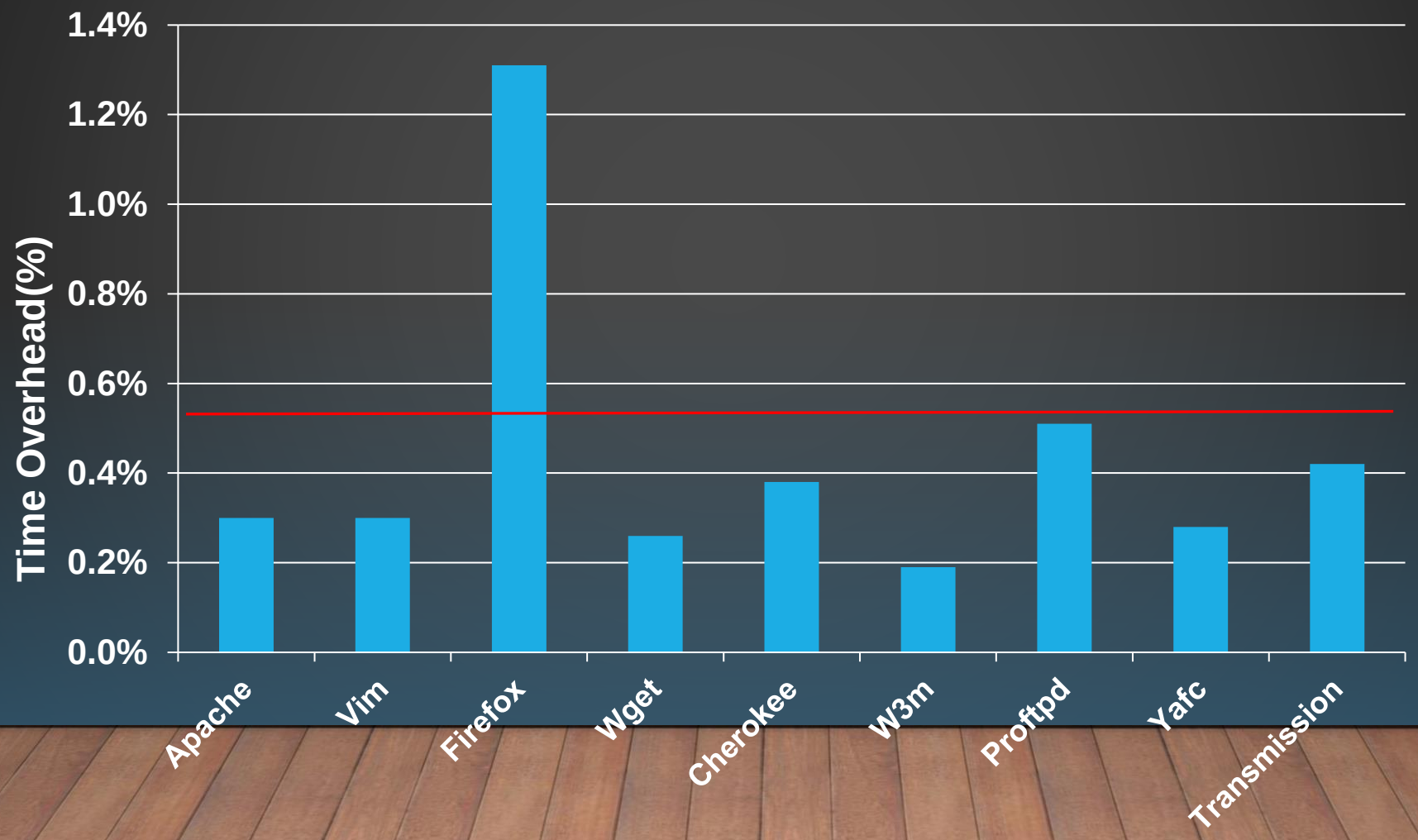
EVALUATION PROGRAMS

	Applications
Servers	Sshd-5.9
	Sendmail-8.12
	Proftpd-1.3.4
	Apache-2.2.21
	Cherokee-1.2.1
UI Programs	Wget-1.13
	W3m-0.5.2
	Pine-4.64
	MidnightCommand-4.6.1
	Vim-7.3
	Bash-4.2
	Firefox-11
	Yafc-1.1.1
	Transmission-2.6

LOGGING OVERHEAD (TIME)

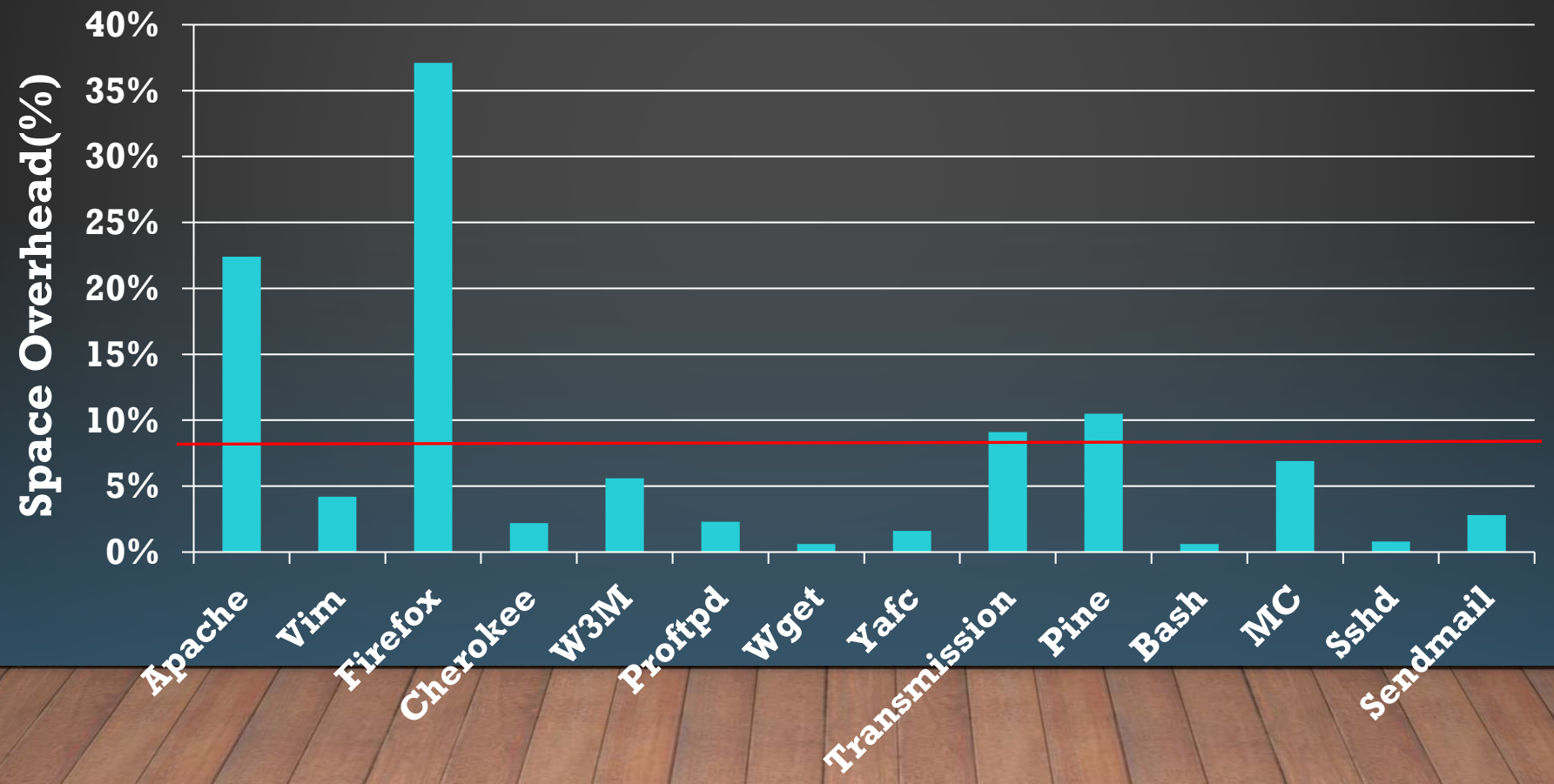


LOGGING OVERHEAD (TIME)



LOGGING OVERHEAD (SPACE)

- Additional space consumption for storing unit / unit dependence events



BEEP LIMITATIONS

- Low level events
  - Mouse click
- Training for inter-unit dependences
  - Very heavy process
- Excessive Units
  - Mouse movement events
- Meaningful execution units
  - Tab
- Multiple Perspectives
  - E.g. tabs v.s. pages
  - No training
- Less units
  - Drop units if possible

MPI

- Inspired by process isolation mechanisms in operating systems
- **IDEA**: Execution partitioning based on user-defined *Tasks*
  - Task: represented by data structures
- Different tasks indicate different perspectives
  - Firefox: Tabs, Pages

ANNOTATION

- Part of Clang/Gcc language extensions
- Widely used: Firefox
  - 926 different types of annotations
  - NS_STACK_CLASS: 406 annotated classes
- Adding customized *attributes* to Variables/Functions etc.
  - *__attribute__((annotate("annotation strings")))*

BASIC ANNOTATIONS

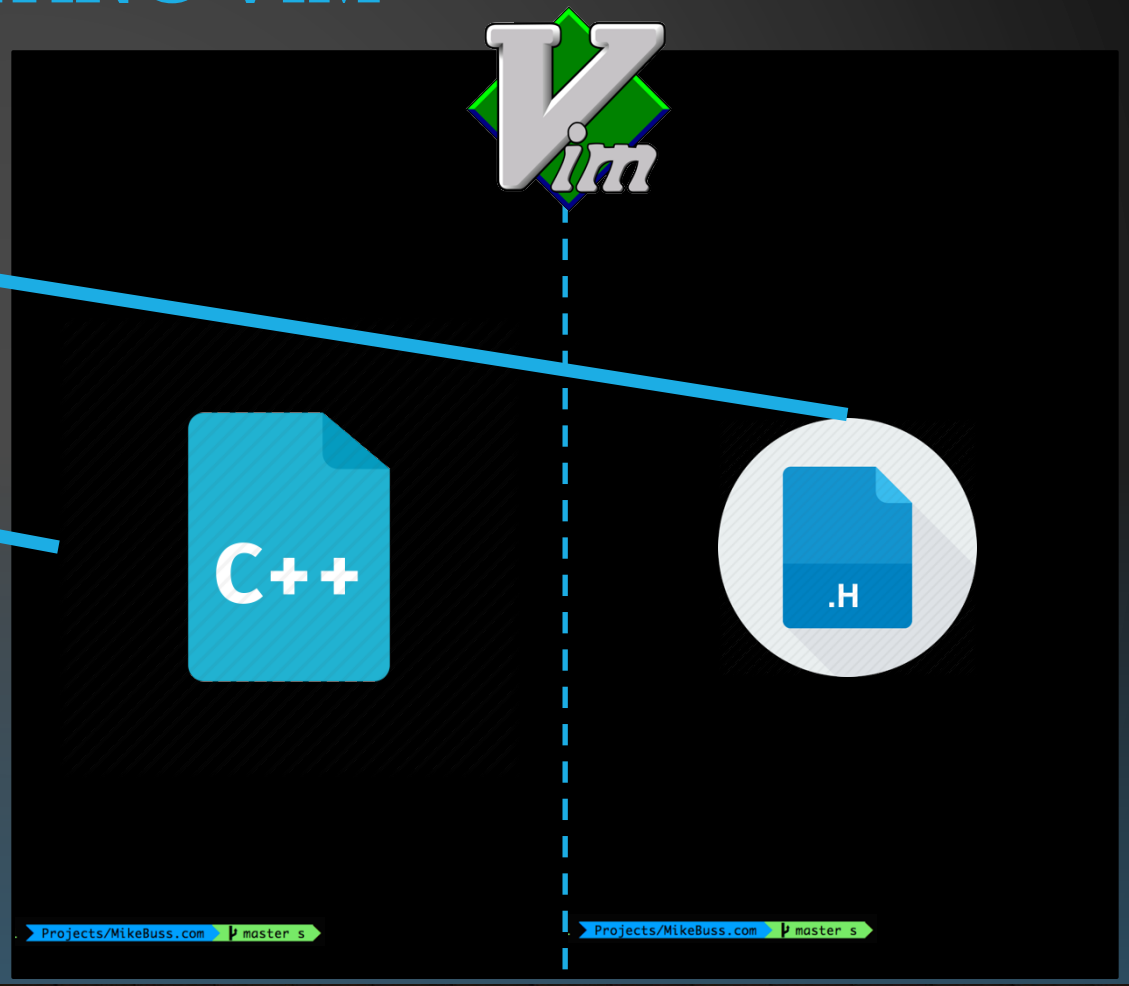
- *@indicator -> current* in OS
  - *Indicating the switches between task instances*
  - This defines the place (where) to instrument
- *@identifier -> pid* in OS
  - *Identifying different task instances*
  - This defines the the value (what) to expose
- *@channel -> IPCs* in OS
  - *Communication channels used between instances*
  - This helps find relationships of all instances

ANNOTATING VIM

```
struct buf_T {
  @identifier
  int buf_id;
  char* name;
  . . .
};
```

```
@indicator
buf_T* curbuf;
```

```
@channel
struct yankreg *y_current;
```



ANNOTATION EXAMPLE

```
do_ecmd(. . .){
  . . .
  //a new buffer
  curbuf = buf;
  curbuf->name = . . .
}
```

```
if (curbuf != oldbuf) {
  oldbuf = curbuf;
  expose(curbuf->buf_id);
}
```

```
expose(id){
  kill(-100, id);//for linux audit system
}
```

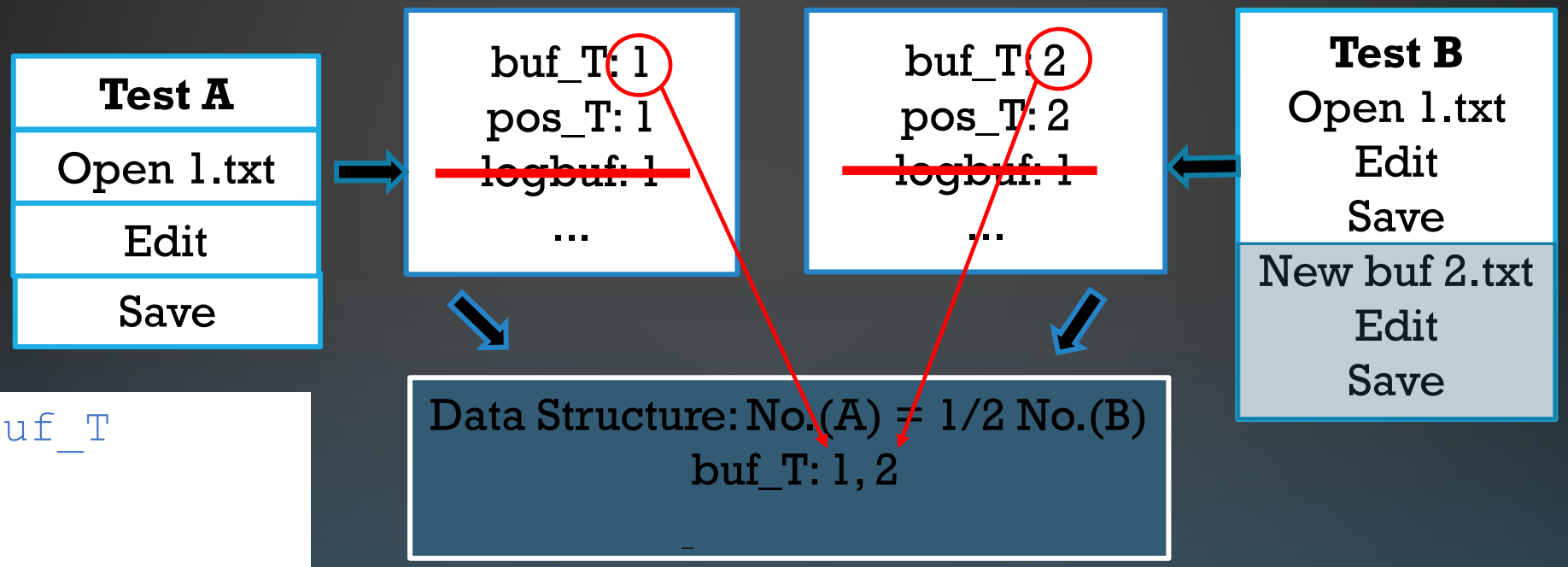
ANNOTATION MINER

- Tasks can be represented by data structures
  - *buf_T* in Vim

- Annotation miner is used to help developers find such data structures

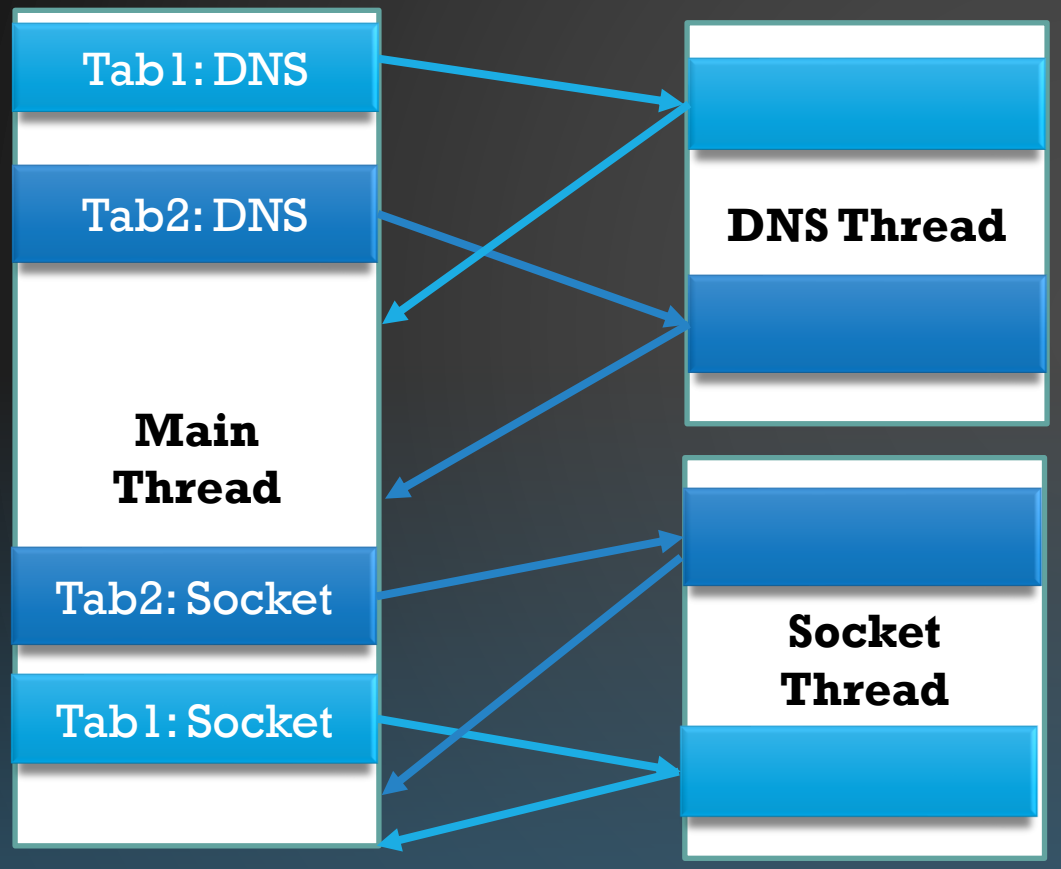
ANNOTATION MINER

Type: # of instances




```
struct buf_T
```

THREAD MODEL



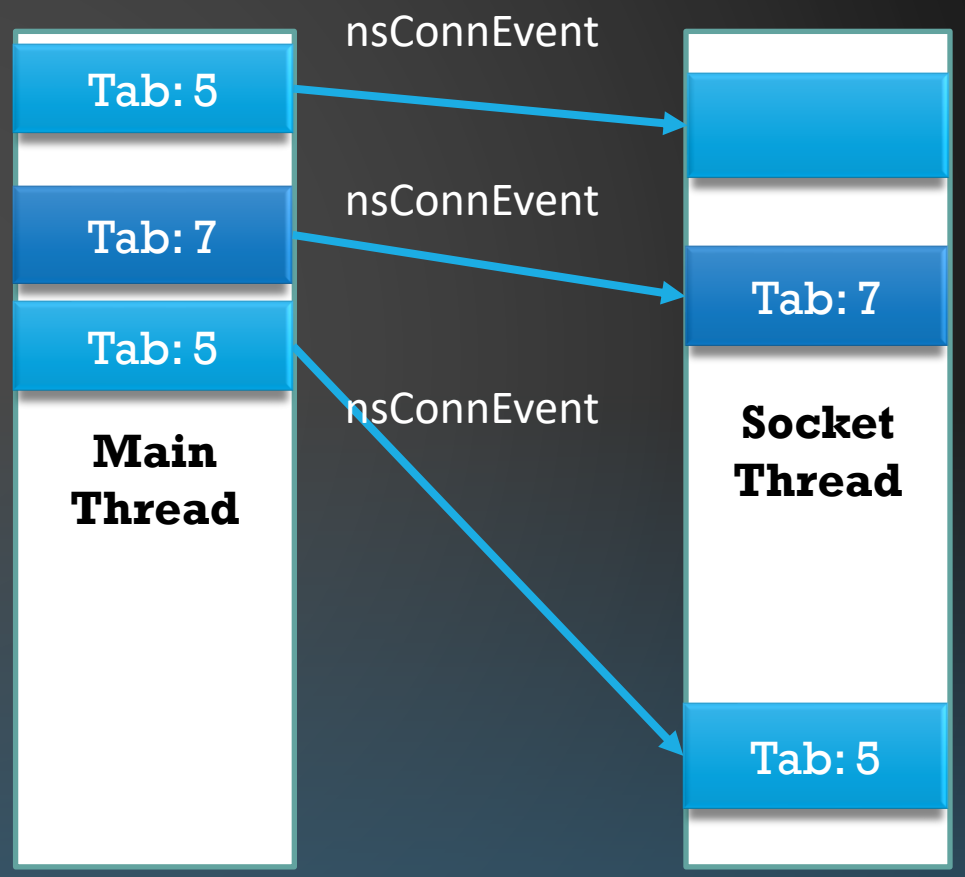
- Worker Threads
  - Working on the same type of tasks
  - Tasks in this thread are *sub-tasks*
  - One thread is serving for multiple *top-level tasks*
- *@delegator*
  - Add one field for top-leve task id
  - Inherit task identifiers from the top-level task

ANNOTATING FIREFOX

```
@delegator
calss nsConnEvent {};

PostEvent(...) {
   event = new nsConnEvent(...);
   rv = target->Dispatch(event);
}
```

```
event->global_id = current_id;
```



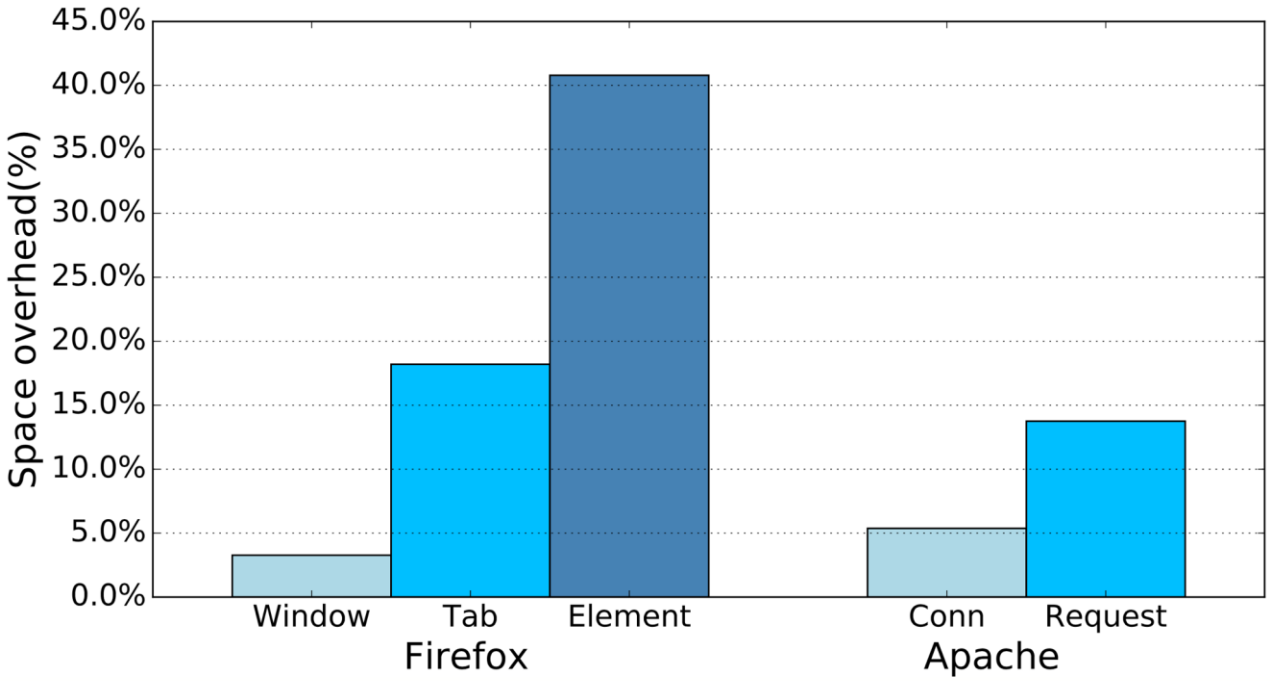
EVALUATION: ANNOTATIONS

Application	LOC	Annotation				Inst
		ID	IND	Chann	DEL	
Vim	313,283	3	3	2	0	878
Yafc	22,823	2	3	0	1	111
Firefox	8,073,181	3	32	0	1	6,867
TuxPaint	41,682	2	2	0	0	121
Pine	353,665	2	2	2	0	746
Apache	168,801	2	2	0	1	2,437
MC	135,668	2	2	1	0	3,332
ProFTPd	307,050	3	3	0	1	4,905
Transmission	111,903	2	4	0	1	66
W3M	67,291	2	2	0	1	3,718

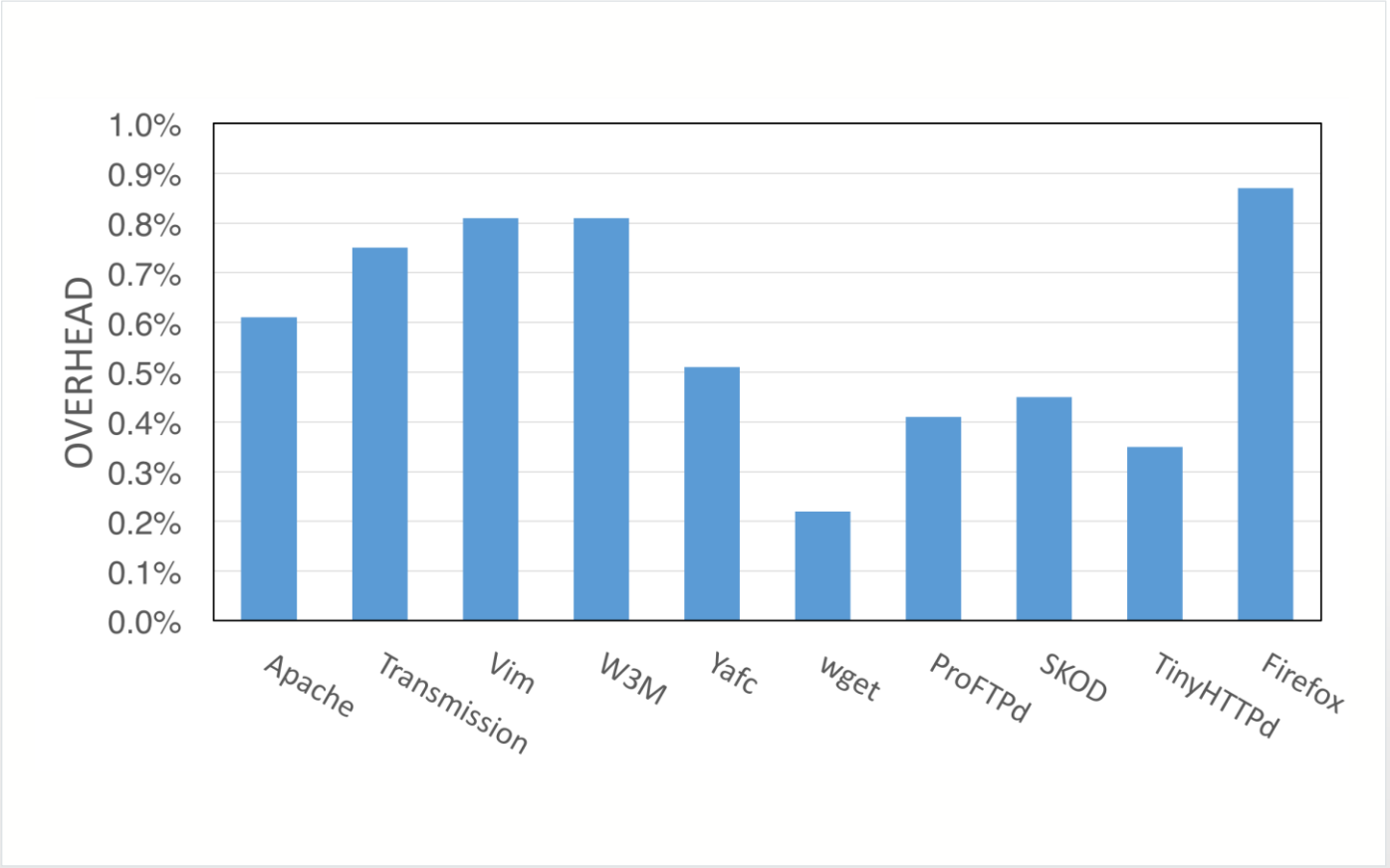
EVALUATION: SPACE OVERHEAD

Applicatoin	Perspective	BEEP-Audit	BEEP-HiFi	MPI-Audit	MPI-HiFi
Apache	Connection	15.38%	12.87%	5.37%	3.75
Bash	Command	0.45%	0.34%	0.41%	0.34%
Firefox	Tab	42.16%	38.23%	18.20%	13.24%
Pine	Command	8.11%	6.09%	7.28%	4.09%
Vim	File	2.23%	2.32%	0.13%	0.13%

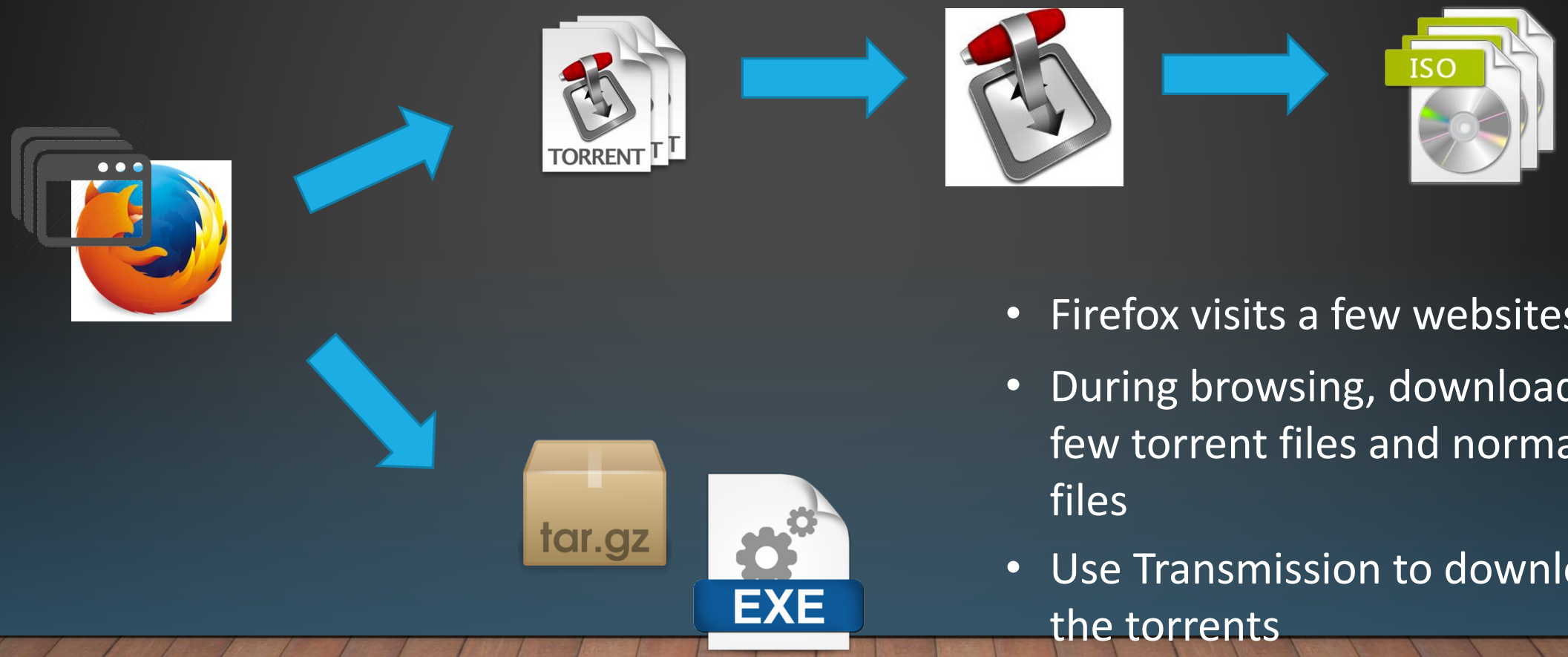
EVALUATION: SPACE OVERHEAD



EVALUATION: RUNTIME OVERHEAD CAUSED BY MPI

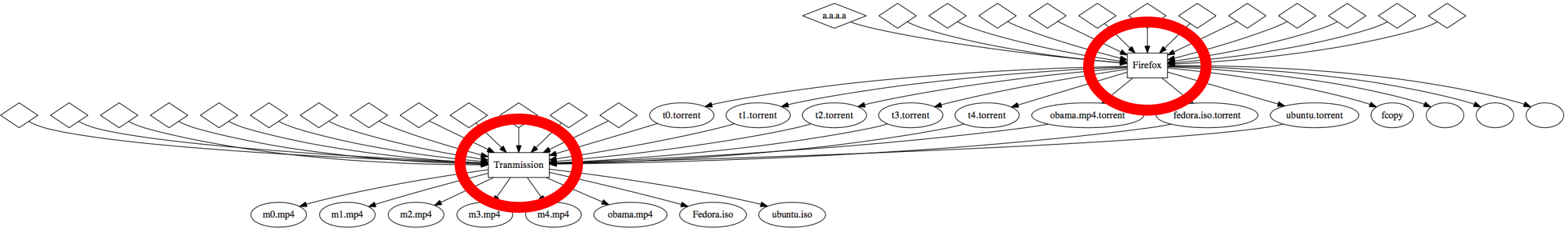


CASE STUDY

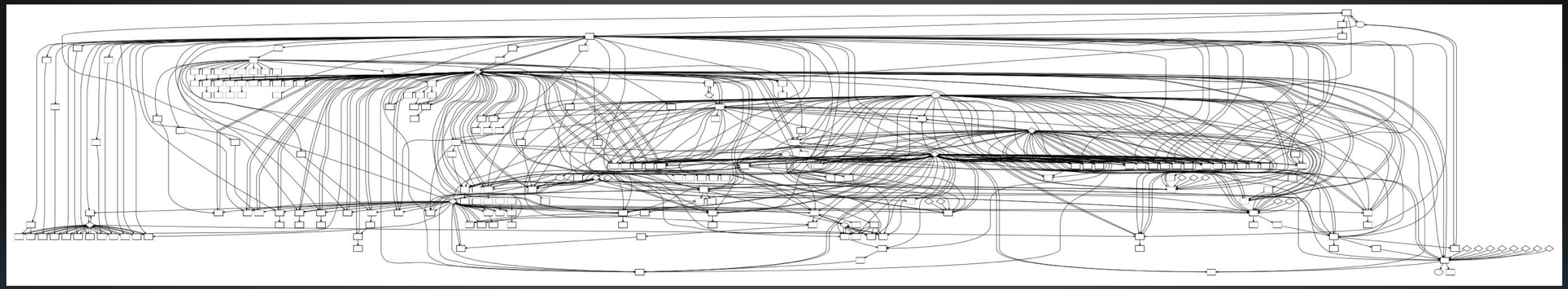


- Firefox visits a few websites
- During browsing, download a few torrent files and normal files
- Use Transmission to download the torrents

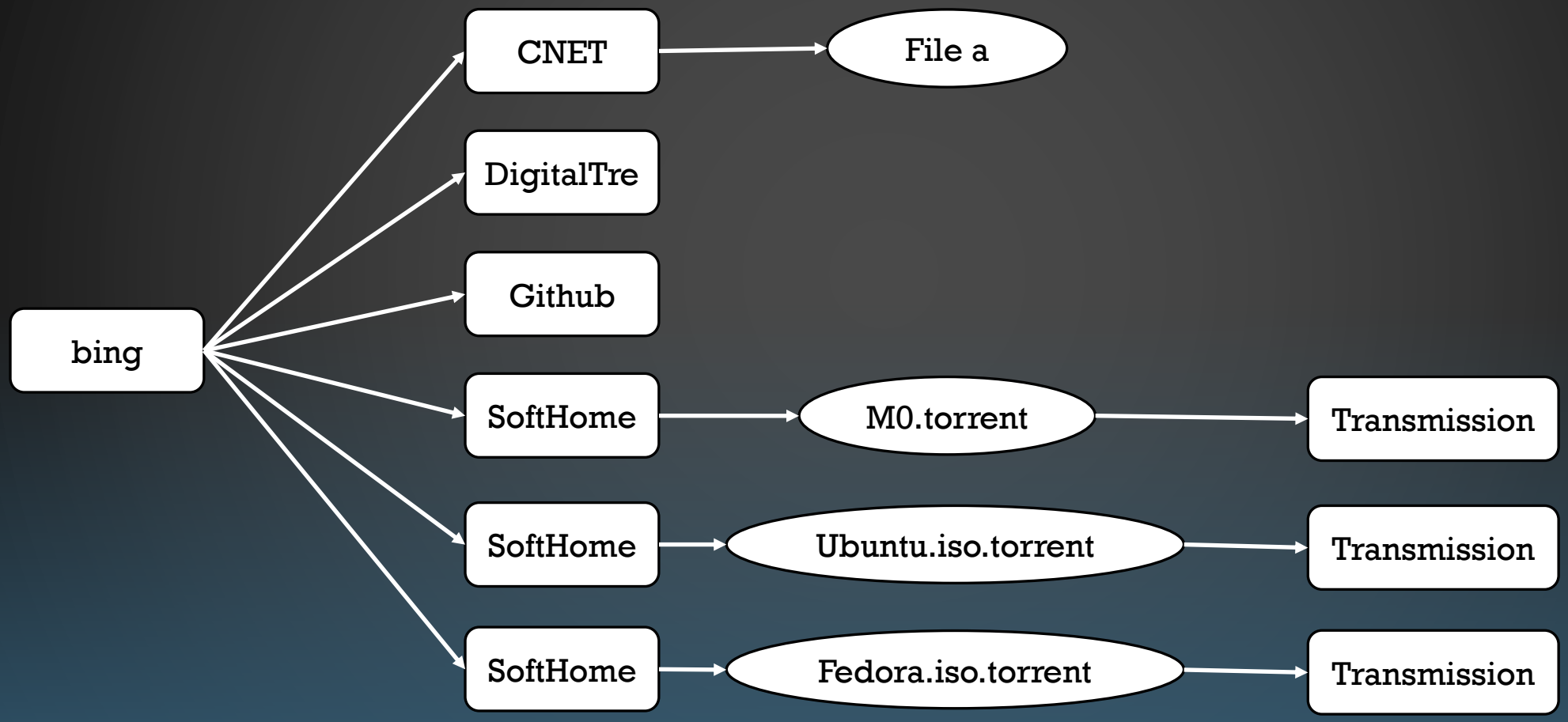
TRADITIONAL SOLUTION



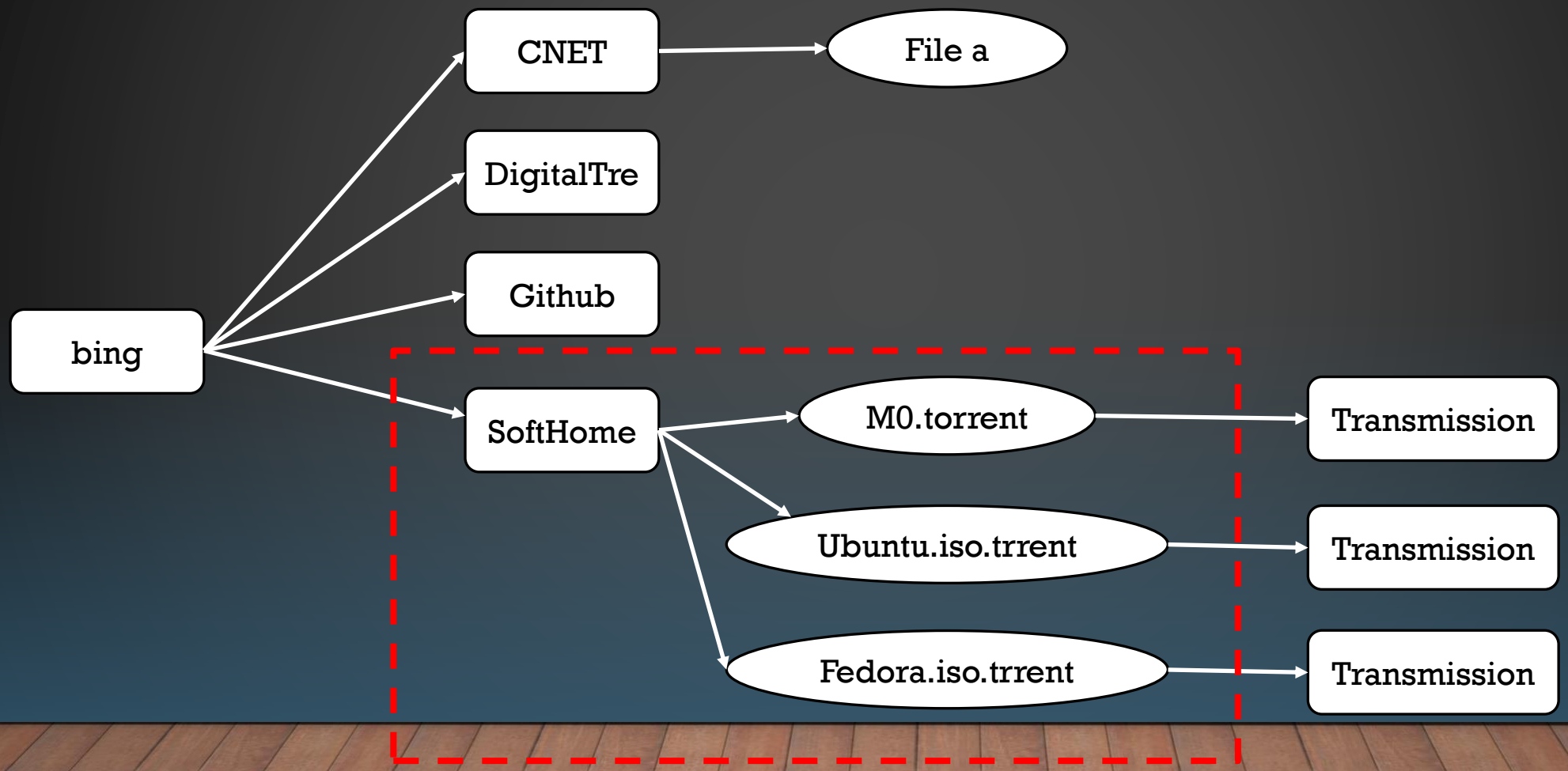
BEEP (ONLY SHOW THE FIREFOX PART)



MPI
(FIREFOX PARTITIONED BY TAB AND TRANSMISSION PARTITIONED BY INPUT FILE)



MPI
(FIREFOX PARTITIONED BY WEBSITE AND TRANSMISSION PARTITIONED BY INPUT FILE)



LIMITATIONS

Source code

LLVM-based solution

Annotation

Miner is helpful but not fully automated

Miner finds the data structure, not the @indicator/@identifier variables