Accurate, Low Cost and Instrumentation-Free Security Audit Logging for Windows

Shiqing Ma, Kyu Hyung Lee, Chung Hwan Kim, Junghwan Rhee,
Xiangyu Zhang, Dongyan Xu

PURDUE UNIVERSITY

The University of Georgia

NEC Laboratories America
Relentless passion for innovation

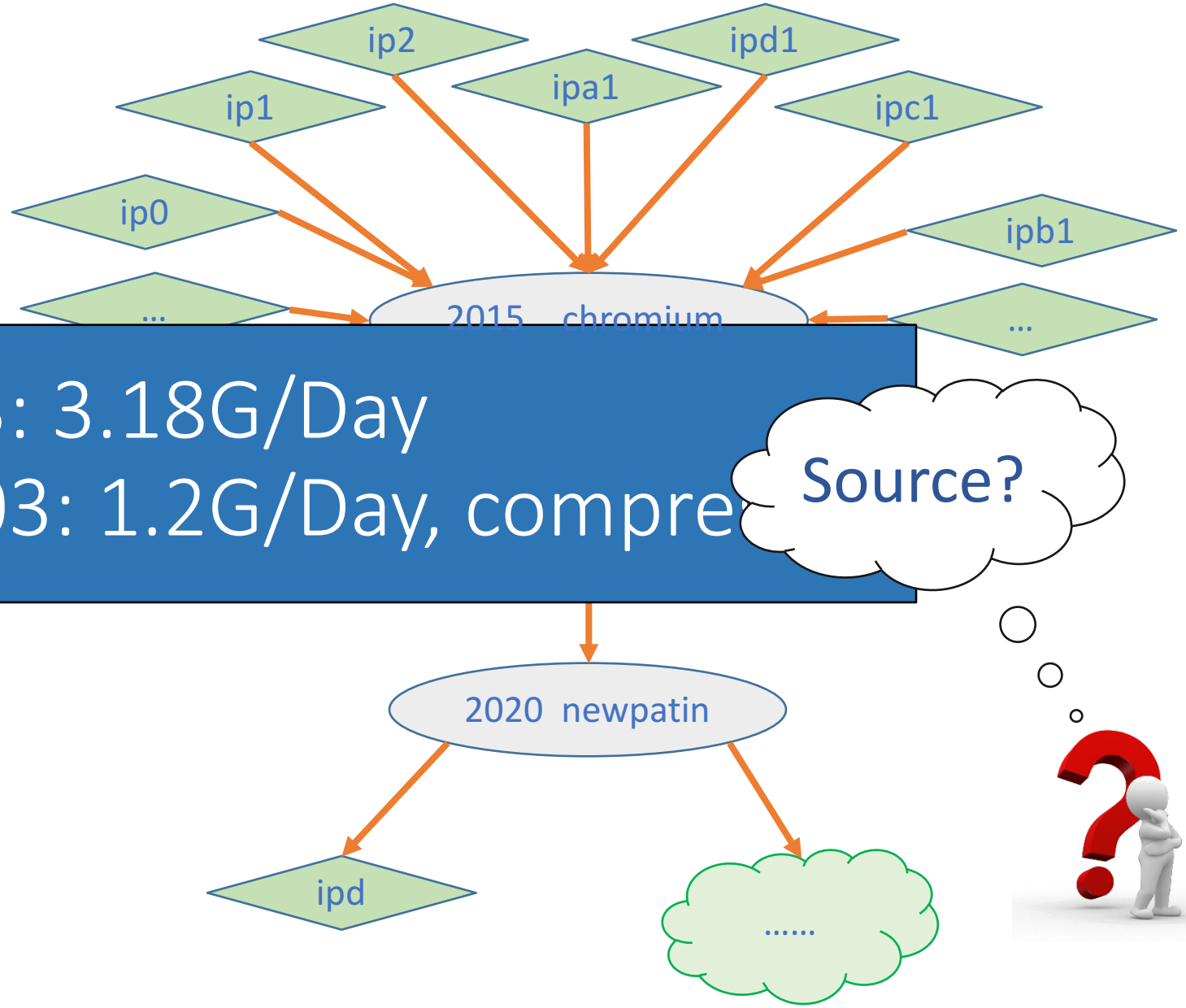
Advanced Cyber Attacks (e.g. APTs): What can we do?

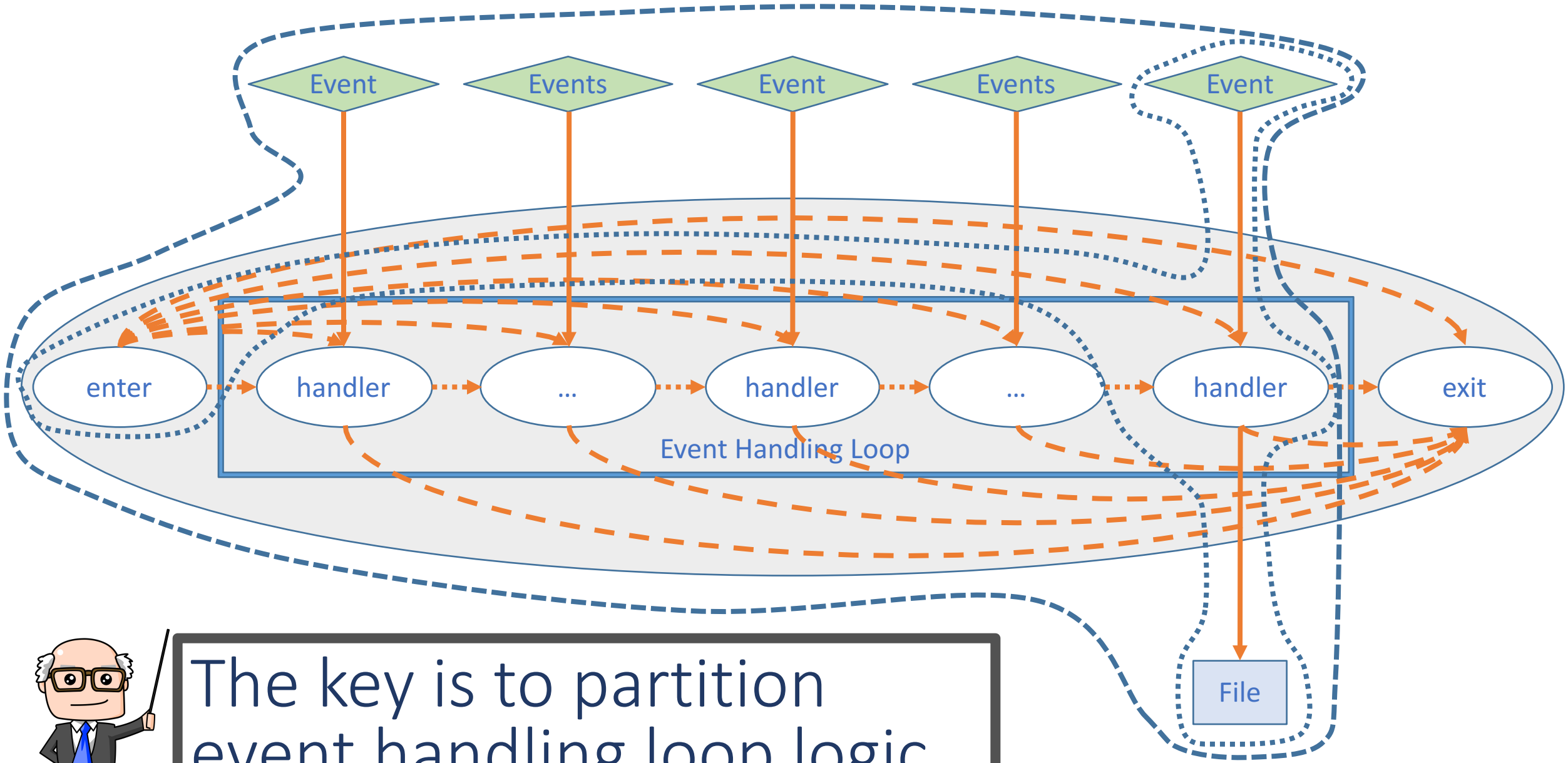
- Defense!
  - Firewall
  - Anti-virus software etc.
- There are no one-time-for-all solutions
  - “Because there is no patch for human stupidity”
  - “There are no secure systems, only degrees of insecurity”
- Fast response is important.
  - Forensics: understand what happened and how.
  - Backward/forward tracing

Process 2015 created, chromium from C:\programs\chromium.exe
2015 reads from ip0
2015 reads from ip1
2015 reads from ip2
……
2015 reads from ipa1
2015 reads from ipb1
2015 reads from ipd2
2015 reads from ipc1
2015 reads from ip100
2015 writes file C:\Downloads\A.exe
2015 reads from ipc2
2015 writes file C:\Downloads\D.exe
2015 writes file C:\Downloads\B.docx
2015 writes file C:\Downloads\C.pptx
…
Process 2020 created, newpaint from C:\Downloads\D.exe
2015 reads from ip200
2020 sends to ipd
…

Process 2015 created, chromium from C:\programs\chromium.exe
2015 reads from ip0
2015 reads from ip1
2015 reads from ip2
……
2015 reads from ipa1
2015 reads from ipb1
2015 reads
2015 reads
2015 reads
2015 writes
2015 reads
2015 writes
2015 writes file C:\Downloads\B.docx
2015 writes file C:\Downloads\C.pptx
…
Process 2020 created, newpaint from C:\Downloads\D.exe
2015 reads from ip200
2020 sends to ipd
…

K.H. Lee, CCS'13: 3.18G/Day
S.T. King, SOSP'03: 1.2G/Day, compre





The key is to partition event handling loop logic.

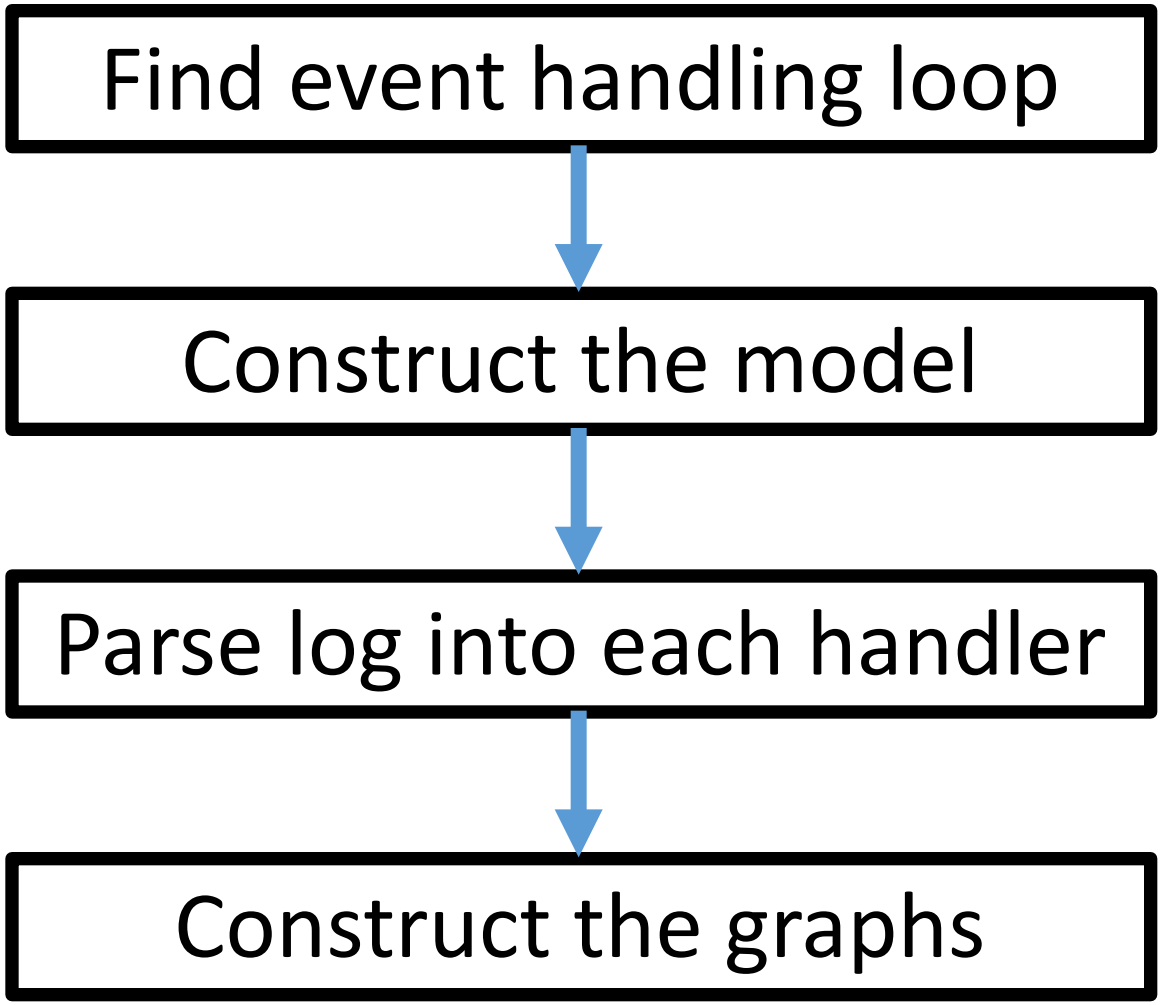
Solution

State-of-the-art work

- Requires training
- Requires instrumentation

Our solution

- Solves this problem
- Native run



Event Tracing for Windows (ETW)

Event(TimeStamp,Processor)

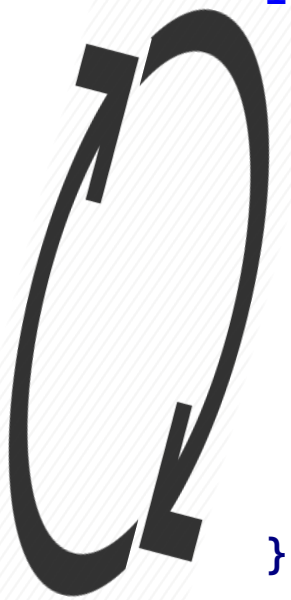
- Event Type
  - FileRead etc.
- Event specific
  - FileObject, IOFlags etc.
- Process ID, Thread ID etc.
- Stack

Stack

1. TurboDispatchJumpAddressEnd+0x690@wow64cpu.dll
2. …
3. winnt_get_connection+0x4b@libhttpd.dll
4. worker_main+0x27@libhttpd.dll
5. …
6. RtlInitializeExceptionChain+0x36@ntdll.dll

Find the event handling loop

```
1. void main() {
2.     init();
3.     while(True) {
4.         read_cmd();
5.         if (cmd == FileDownload) {
6.             if(file ready) {
7.                 fd = open_file(file_name);
8.                 if(open fails)
9.                     errmsg_continue(MSG2);
10.                buf = memory_allocation(size);
11.                while(transfer not done) {
12.                    read_file(fd, buf);
13.                    wirte-data(socket, buf);
14.                }
15.                memory_free(buf);
16.                close_file(fd);
17.            } else
18.                errmsg_continuemsg(socket, MSG3);
19.        } else if(cmd ==...) { ... }
20.    }// end while
21.    server_exit();
22.}
```



SocketRead:...-> main -> read_cmd ->...

FileOpen:...-> main -> open_file ->...

FileRead:...-> main -> read_file ->...

SocketWrite:...-> main -> write_data ->...

FileClose:...-> main -> close_file ->...

Model Construction

```
1. void main() {
2.     init();
3.     while(True) {
4. F1:     read_cmd();
5.         if (cmd == FileDownload) {
6.             if(file ready) {
7. F2:             fd = open_file(file_name);
8.                 if(open fails)
9. F3:                 errmsg_continue(MSG2);
10.                buf = memory_allocation(size);
11.                while(transfer not done) {
12.F4:                 read_file(fd, buf);
13.F5:                 wirte-data(socket, buf);
14.                }
15.                memory_free(buf);
16.F6:             close_file(fd);
17.            } else
18.F7:             errmsg_continuemsg(socket, MSG3);
19.        } else if(cmd ==...) { ... }
20.    }// end while
21.    server_exit();
22.}
```

Model(3-20)$

=F1·Model(5-20)$

=F1·(Model(6-18) | …)$

=F1·((Model(7-16) | F7)| …)$

=F1·((F2·Model(8-16) | F7)| …)$

=F1·((F2·([F3] ·Model(11-16))|F7)|…)$

=F1·((F2·([F3] ·(Model(11-14)·F6))|F7)|…)$

=F1·((F2·([F3] ·((F4·F5)*·F6))|F7)|…)$

Log Partitioning

^F1·((F2·([F3] ·((F4·F5)*·F6))|F7)|…)$

```
1.……
2.SocketRead  : …-> main -> F1 ->…
3.FileOpen    : …-> main -> F2 ->…
4.FileRead    : …-> main -> F4 ->…
5.SocketWrite: …-> main -> F5 ->…
6.FileRead    : …-> main -> F4 ->…
7.SocketWrite: …-> main -> F5 ->…
8.FileClose   : …-> main -> F6 ->…
9.SocketRead  : …-> main -> F1 ->…
10.……
```

^F1·((F2·([F3] ·((F4·F5)*·F6))|F7)|…)$

^F1·(F2·([F3] ·((F4·F5)*·F6))|F7)$

^F1·(F2·((F4·F5)*·F6))$

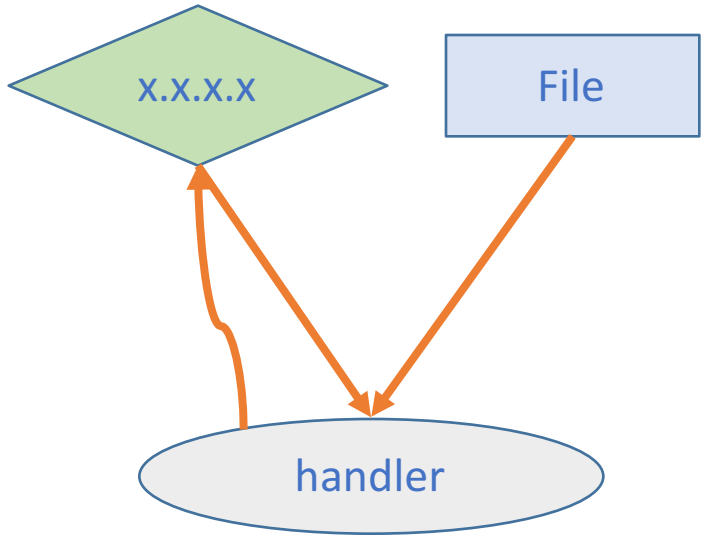
^F1·(F2·((F4·F5)*·F6))$

^F1·(F2·((F4·F5)*·F6))$

^F1·(F2·((F4·F5)*·F6))$

Graph Construction

```
1.……
2.SocketRead  : IP=x.x.x.x
3.FileOpen    : ObjID=0xff
4.FileRead    : ObjID=0xff, Offset=0, len=100
5.SocketWrite: IP=x.x.x.x
6.FileRead    : ObjID=0xff, Offset=100, len=10
7.SocketWrite: IP=x.x.x.x
8.FileClose   : ObjID=0xff
9.SocketRead  : ……
10.……
```



Log Reduction

len=110

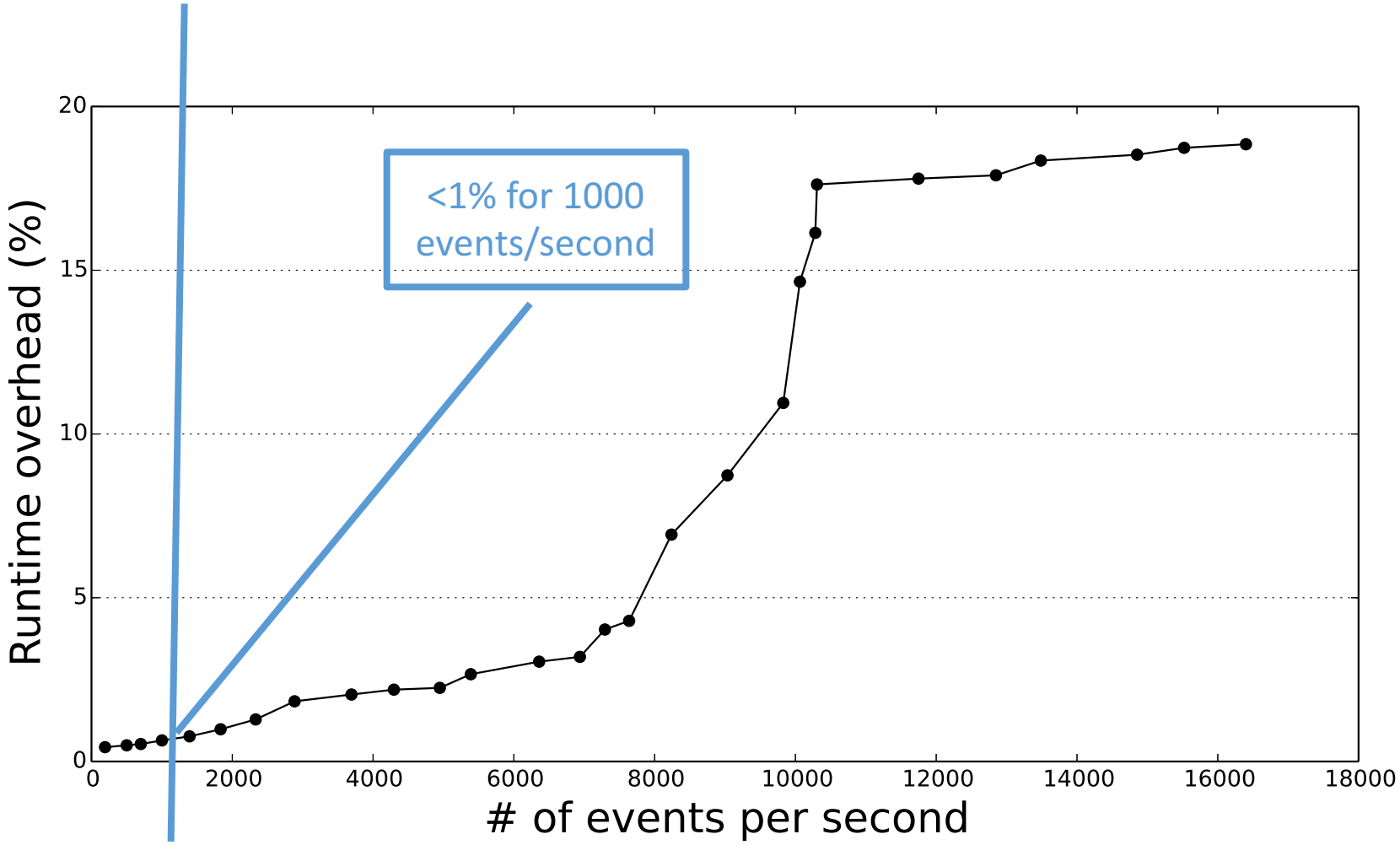
```
1.……
2.SocketRead  : IP=x.x.x.x
3.FileOpen    : ObjID=0xff
4.FileRead    : ObjID=0xff, Offset=0, len=100
5.SocketWrite: IP=x.x.x.x
6.FileRead    : ObjID=0xff, Offset=100, len=10
7.SocketWrite: IP=x.x.x.x
8.FileClose   : ObjID=0xff
9.SocketRead  : ……
10.……
```

LogGC from CCS'13

Evaluation Setup

- Hardware
  - CPU: Intel i7-3880
  - RAM: 12GB
- Operating System
  - Windows Server 2008 R2
  - 64-bit

Evaluation: usability



	#Event / second
User 1	133.04
User 2	**128.89**
User 3	184.30
Server 1	328.48
Server 2	**566.02**

[,]

Evaluation: reduction

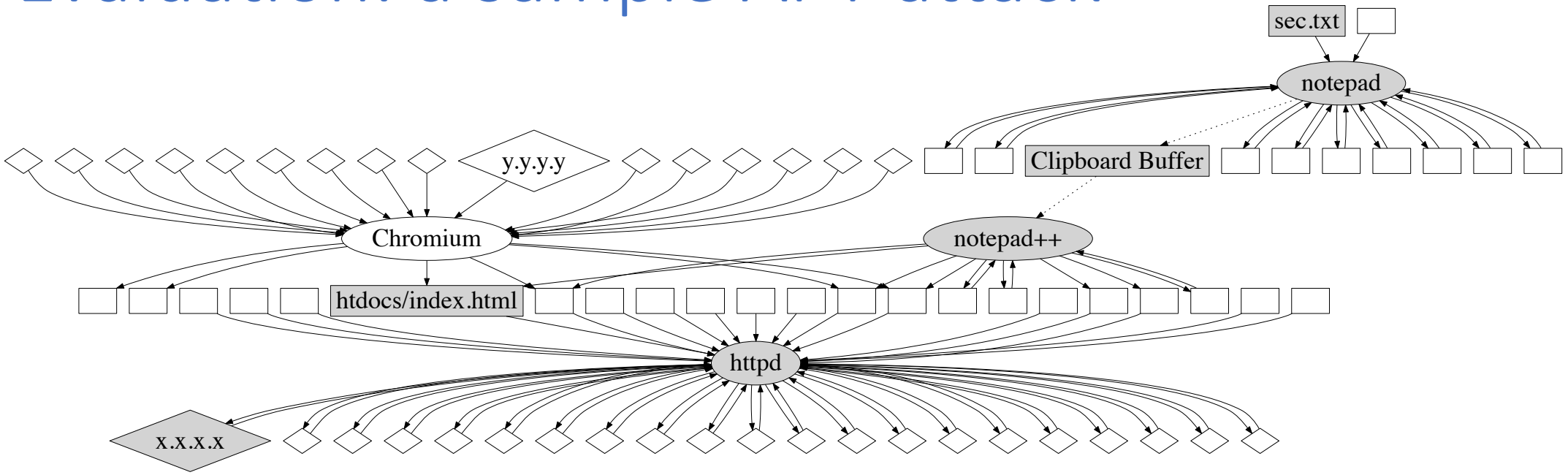
Program	# Events		
	Before	After	Ratio
TextTransfer	316	6	1.90%
Chromium	102,206	4,179	4.09%
DrawTool	15,438	74	0.48%
NetFTP	10,621	580	5.46%
AdvancedFTP	1,651	43	2.66%
HTTPD	37,171	2,052	5.52%
IE	29,969	2,275	7.59%
Paint	7,085	78	1.10%
Notepad	11,704	30	0.26%
Notepad++	5,516	136	2.47%
SimpleHTTP	779	40	5.13%
Sublime Text	30,372	316	1.04%

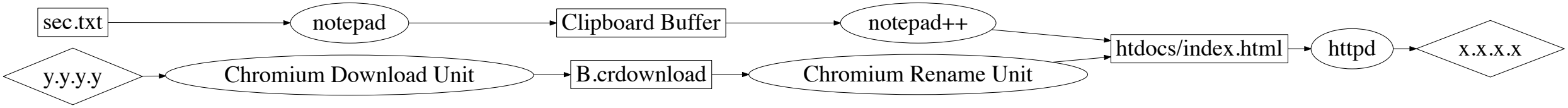
Evaluation: query logs

Scenario	# Nodes			# dges			Correctness	
	Raw	Unit	GC	Raw	Unit	GC	Back	For
Mis-configuration	173	10	10	204	10	10	[redacted]	✓
Phishing	573	21	21	693	32	32	✓	✓
Info leak	10,222	11	11	20,532	10	10	✓	✓
Spyware	9.282	9	9	11.244	8	8	✓	✓

Evaluation: a sample APT attack

Evaluation: a sample APT attack





Related work

- System level dependency tracing
  - S.King'03, J.Chow'04, A.Goel'05, X.Jiang'06, Muniswamy-Reddy'06, R.Hasan'09, T.Kim'10, N.Zhu'10, J.Newsome'10, Polhy'12, K.H.Lee'13, A.Bates'15
- Information flow tracing
  - J. Newsome'05, H.Yin'07, K.K.Muniswamy-Reddy'09, B.C.Tak'09, Enck'10, K.Jee'12, V.P. Kemerlis'12
- Log-based security applications
  - C. Kolbitsch'09, W.Xu'09, Xie'11, I. Beschastnikh'11, K.Xu'12, K.H.Lee'13, D.Arp'14, H.Zhang'14

Conclusion

- Our system for advanced attacks (e.g., APTs) investigation
  - Accurate
    - Solves the dependency explosion problem
  - Low-cost
    - Run time
      - ***<1%*** for normal usage
    - Storage
      - Removes more than ***92%*** original log entries
  - Advances start-of-the-art
    - ***No training or instrumenting applications needed***

This research was supported in part by DARPA
under contract FA8650-15-C-7562
and NSF under award 1409668.