

SysFlow: Scalable System Telemetry for Improved Security Analytics

—
Fred Araujo and Teryl Taylor

Network Monitoring

– Packet analysis is not feasible at scale

- suitable for in-depth analysis of specific conversations

– Flow analysis is a great idea :-)

- collect metadata from network traffic and group sequence of packets sharing the same properties
- applications include bandwidth monitoring, network threat detection, and performance analysis

– NetFlow

- Cisco's proprietary network protocol used for flow analysis
- collects and aggregates information about network traffic flowing through a device with an enabled NetFlow feature
- variations: IPFIX, sFlow, NetStream

flow cache

Destination IP	Source IP	Destination Port	Source Port	Source Interface	Protocol	Bytes
172.13.1.45	110.3.1.15	80	60444	1	TCP	245
172.13.1.45	110.3.1.15	80	60345	1	TCP	354

No.	Time	Source	Destination	Protocol	Length	Info
461	30.700595	52.37.243.173	9.74.62.27	TCP	66	443 → 63799 [ACK]
462	30.701370	52.37.243.173	9.74.62.27	TCP	66	443 → 63801 [ACK]
463	30.702290	52.37.243.173	9.74.62.27	TLSv1...	211	Application Data
464	30.702355	9.74.62.27	52.37.243.173	TCP	66	63799 → 443 [ACK]
465	30.705708	52.37.243.173	9.74.62.27	TCP	66	443 → 63808 [ACK]
466	30.706997	52.37.243.173	9.74.62.27	TLSv1...	211	Application Data
467	30.707085	9.74.62.27	52.37.243.173	TCP	66	63801 → 443 [ACK]
468	30.713938	52.37.243.173	9.74.62.27	TLSv1...	240	Application Data
469	30.714009	9.74.62.27	52.37.243.173	TCP	66	63799 → 443 [ACK]
470	30.719674	52.37.243.173	9.74.62.27	TLSv1...	174	Application Data
471	30.719678	52.37.243.173	9.74.62.27	TLSv1...	240	Application Data
472	30.719756	9.74.62.27	52.37.243.173	TCP	66	63808 → 443 [ACK]
473	30.719756	9.74.62.27	52.37.243.173	TCP	66	63808 → 443 [ACK]
474	30.891670	18.213.202.210	9.74.62.27	TLSv1...	1345	Application Data
475	30.891734	9.74.62.27	18.213.202.210	TCP	66	65279 → 443 [ACK]


```

▶ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: Cisco_f2:39:06 (d4:2c:44:f2:39:06), Dst: Apple_49:8f:92 (8c:85:90:49:8f:92)
▶ Internet Protocol Version 4, Src: 13.35.125.104, Dst: 9.74.62.27
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 65239, Seq: 1, Ack: 1, Len: 0
    
```



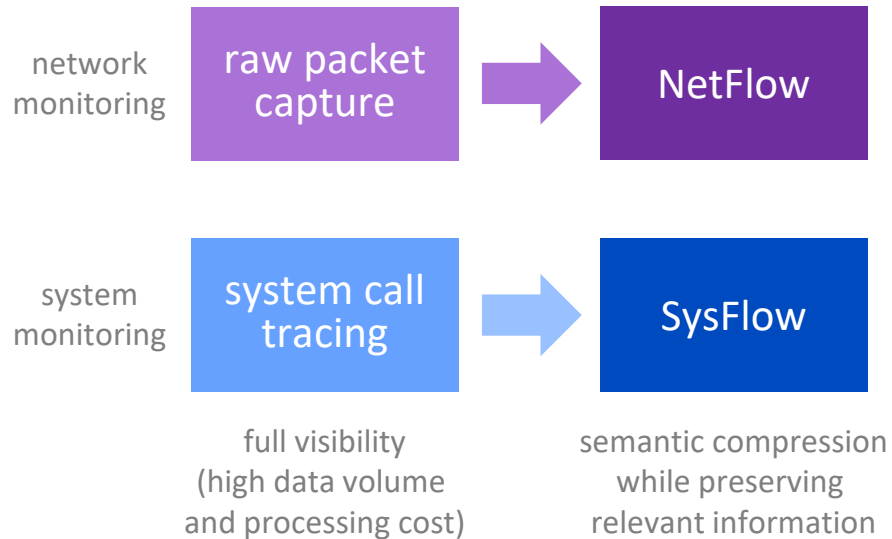
```

0000  8c 85 90 49 8f 92 d4 2c 44 f2 39 06 08 00 45 00  ...I..., D-9...E-
0010  00 34 5d 1f 40 00 ed 06 5e b4 0d 23 7d 68 09 4a  >4]@...^...#}h-J
0020  3e 1b 01 bb fe d7 00 84 fb a6 0c 8b 7a b4 80 10  >.....Z...
0030  00 7a 65 b2 00 00 01 01 08 0a 12 66 79 56 c1 0e  .ze.....fyV..
0040  6d d8                                             m.
    
```

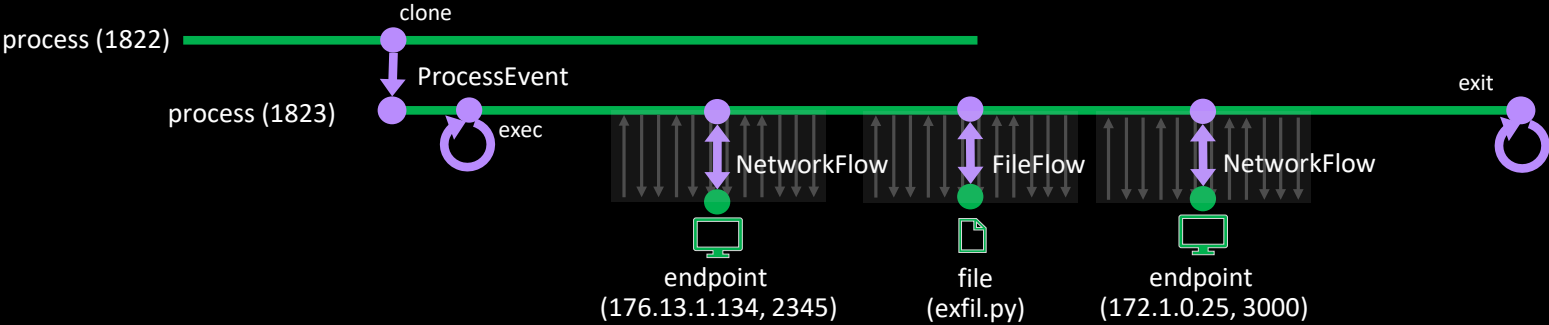
**NetFlow only
captures half
of the telemetry
picture.**

SysFlow

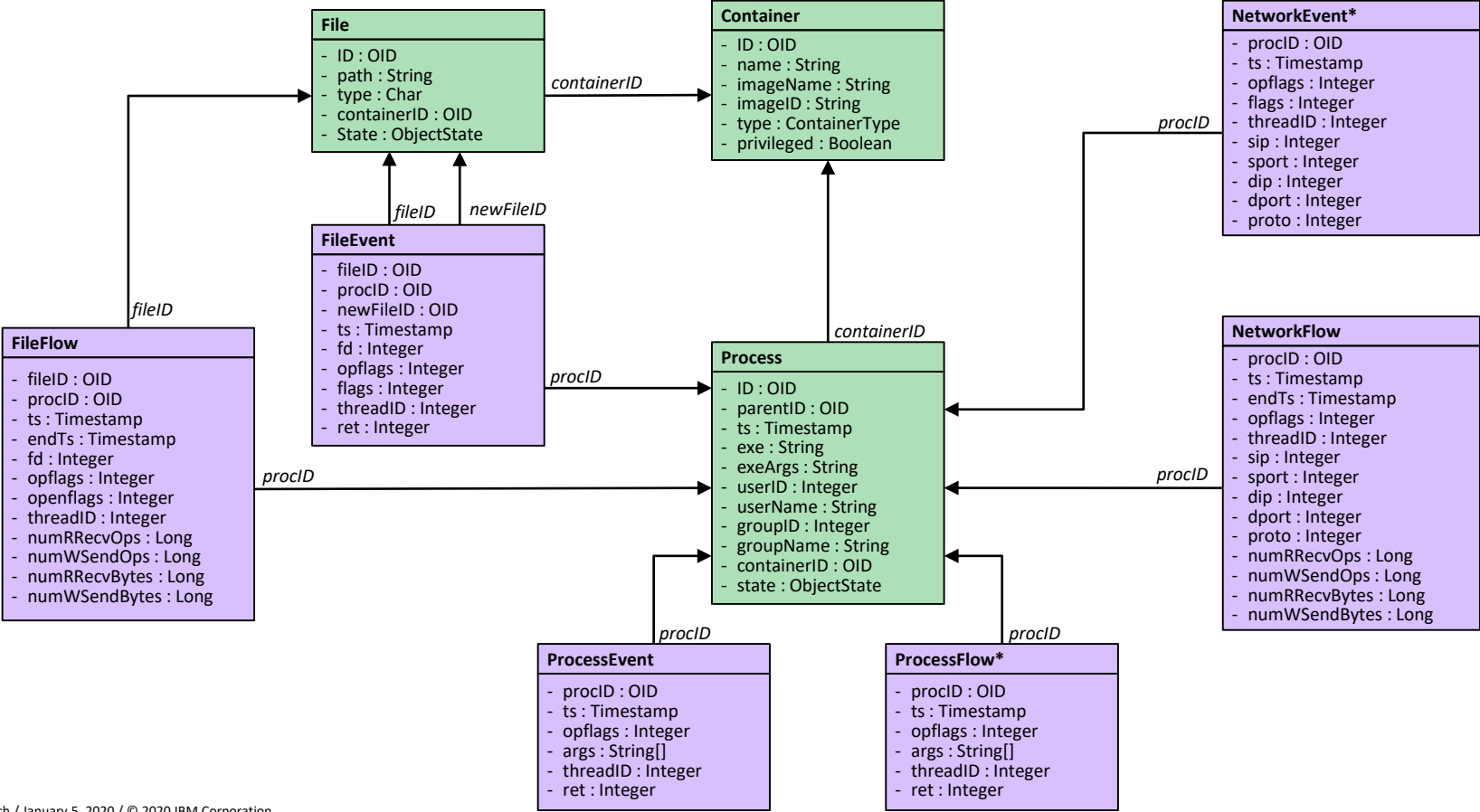
- “NetFlow” for system events
- Captures **process** control flows, **file** interactions, and **network** communications
- Container-aware, flow-centric semantics for system analytics



“Semantically compressed system events for scalable security, compliance, and performance analytics.”



Object-Relational View



Operations

Process Events

CLONE (process/thread)	EXEC (new process)	EXIT (process/thread)	SETUID (change uid)
------------------------	--------------------	-----------------------	---------------------

File Events

MKDIR	RMDIR	LINK	UNLINK
-------	-------	------	--------

SYMLINK	RENAME		
---------	--------	--	--

File Flows

OPEN	SETNS (enter container)	READ	WRITE
------	-------------------------	------	-------

CLOSE	MMAP	CHOWN/CHMOD	MOUNT/UMOUNT
-------	------	-------------	--------------

Network Flows

ACCEPT	CONNECT	SEND	RECEIVE
--------	---------	------	---------

SHUTDOWN	CLOSE		
----------	-------	--	--

Implemented in current release Planned for next release
--

Pretty-printed SysFlow trace (selected attributes):

Process	PID	TID	Op	Flags	Start Time	End Time	Ret	Resource	NBRead	NBWrite	Cont
./server	13823	13823	EXEC		03/25/2019T19:48:00.704111		0				c1
./server	13823	13823	O	C	03/25/2019T19:48:00.704232	03/25/2019T19:48:00.704242		/etc/ld.so.cache	0	0	c1
./server	13823	13823	O	R C	03/25/2019T19:48:00.704263	03/25/2019T19:48:00.704310		/lib64/libc.so.6	832	0	c1
./client	13824	13824	EXEC		03/25/2019T19:48:02.831502		0				c1
./client	13824	13824	O	C	03/25/2019T19:48:02.831617	03/25/2019T19:48:02.831626		/etc/ld.so.cache	0	0	c1
./client	13824	13824	O	R C	03/25/2019T19:48:02.831647	03/25/2019T19:48:02.831692		/lib64/libc.so.6	832	0	c1
./client	13824	13824	CWR	T	03/25/2019T19:48:02.832226	03/25/2019T19:48:12.823003		127.0.0.1:40556-127.0.0.1:8080	80	80	c1
./client	13824	13824	EXIT		03/25/2019T19:48:12.823003		2				c1
./server	13823	13823	A	WR T	03/25/2019T19:48:02.832197	03/25/2019T19:48:13.422795		127.0.0.1:40556-127.0.0.1:8080	80	80	c1
./server	13823	13823	EXIT		03/25/2019T19:48:13.422795		2				c1

Network Flow

- Container/PID: <OID>
- Start time: 03/25/2019T19:48:02.832226
- End time: 03/25/2019T19:48:12.823003
- Flags: CWR T (create/write/read/truncated)
- Thread ID: 12824
- Source IP address: 127.0.0.1
- Source Port: 40556
- Dest IP address: 127.0.0.1
- Destination Port: 8080
- Protocol: TCP
- Bytes sent: 80
- Bytes rcvd: 80
- Send Operations: 2
- Recv Operations: 2

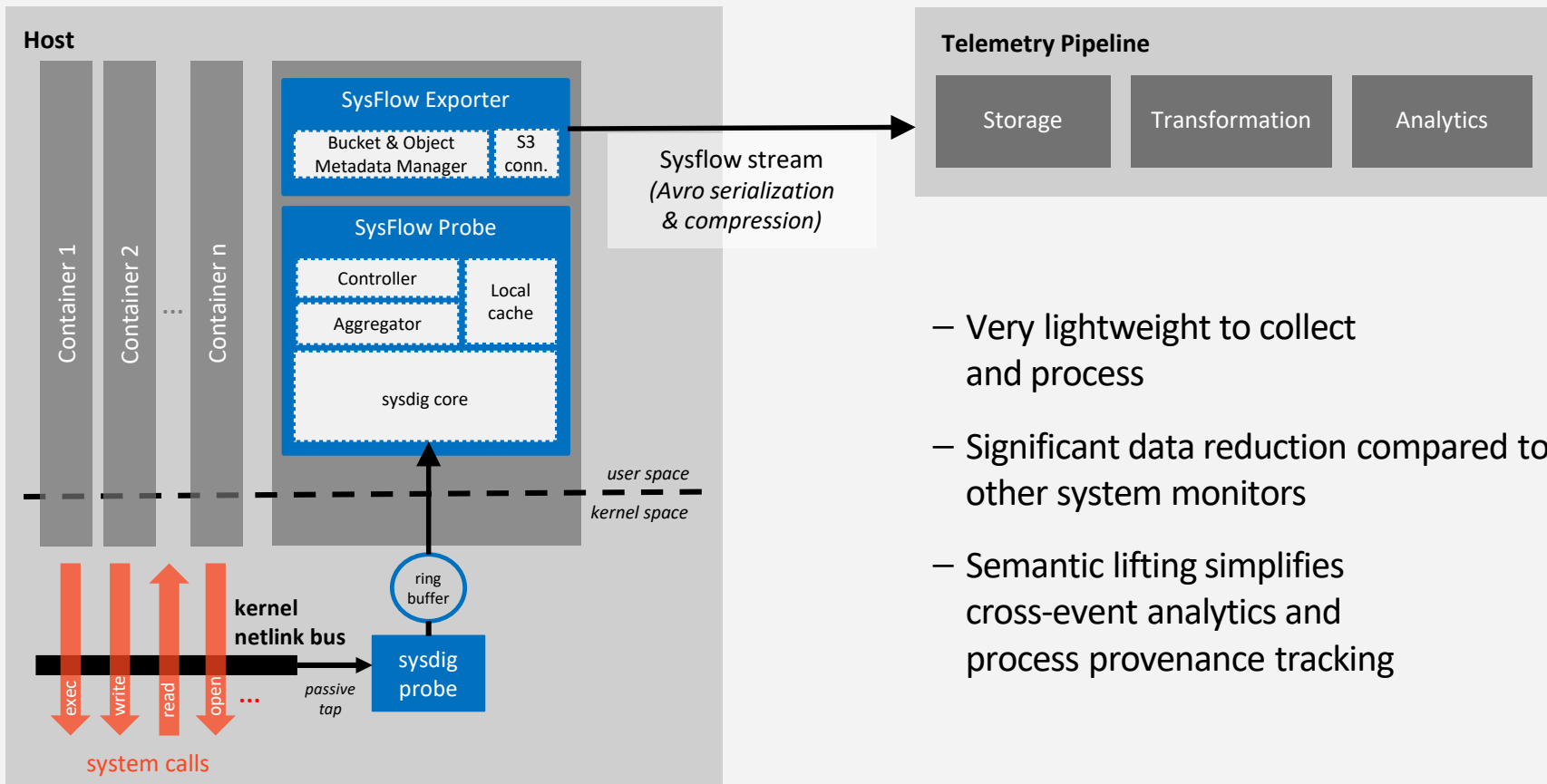
Process

- <OID>
- <createts>
- 12822
- 12824
- ./client
- 8080
- 1000
- ccsi
- 1000
- ccsi
- <CID>

Container

- <CID>
- c1
- <imageid>
- httpd
- DOCKER
- false

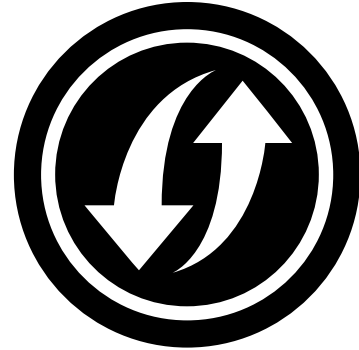
Architectural Overview



- Very lightweight to collect and process
- Significant data reduction compared to other system monitors
- Semantic lifting simplifies cross-event analytics and process provenance tracking

SysFlow Project

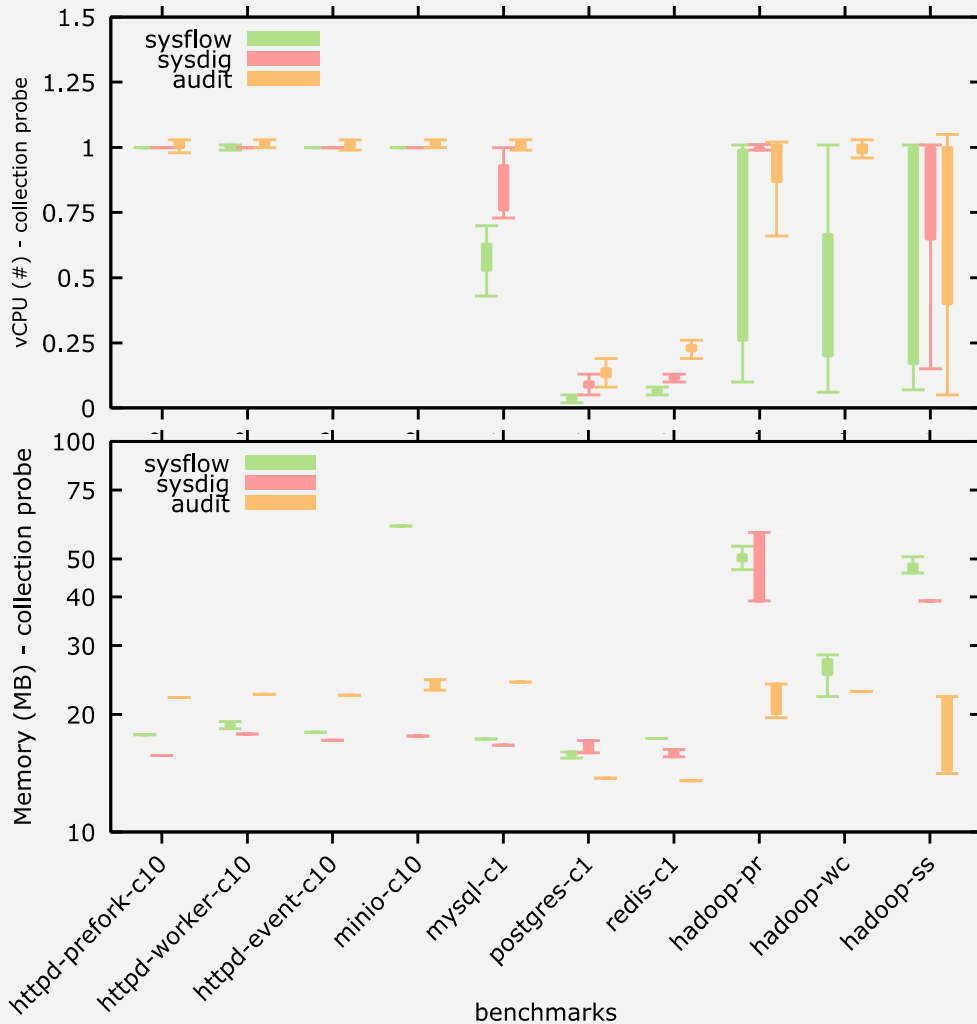
- Open source
github.com/sysflow-telemetry
- Growing set of APIs
Python, C/C++, Go, ...
- Non disruptive and easily deployable
helm and docker deployments



Collection Probe Performance

Benchmarks

HTTPD	Apache Benchmark (HTTP)
Minio	Wasabi S3 BM
Mysql	TPC-H HDB
Postgres	TPC-H HDB
Redis	TPC-H HDB
Hadoop	HiBench



Compression Factors

Trace sizes (# records) for different benchmarks

Benchmark	SysFlow		Sysdig		Audit	
	# records	Size	# records	Size	# records	Size
httpd_prefork	8.19E+05	11	7.45E+06	62	1.94E+05	78
httpd_worker	6.29E+05	7.9	6.40E+06	58	1.93E+05	80
httpd_event	6.42E+05	7.8	5.90E+06	53	1.85E+05	75
minio	7.52E+05	19.5	2.62E+07	552	2.19E+06	966
mysql	1.89E+02	0.09	1.56E+08	2592	7.86E+05	327
postgres	7.08E+03	0.22	9.31E+06	169.2	4.29E+06	2000
redis	9.10E+03	0.15	1.52E+07	68	8.31E+06	4100
hadoop	6.27E+05	16.4	1.09E+07	234	2.47E+06	1700

NetworkFlow (SysFlow)

- Operates at the transport layer
 - monitors system calls (e.g., accept, recv, send)
 - no concept of packet; no remote scan detection*
- Process-centric
 - links network activity to process thread

Network Flow

```
- Container/PID:      <OID>
- Start time:        03/25/2019T19:48:02.832226
- End time:          03/25/2019T19:48:12.823003
- Syscall Flags:     CWR T
(create/write/read/truncated)
- Thread ID:         12824
- Source IP address: 127.0.0.1
- Source Port:       40556
- Dest IP address:   127.0.0.1
- Destination Port:  8080
- Protocol:          TCP
- Bytes sent:        80
- Bytes rcvd:        80
- Send Operations:   2
- Recv Operations:   2
```

NetFlow

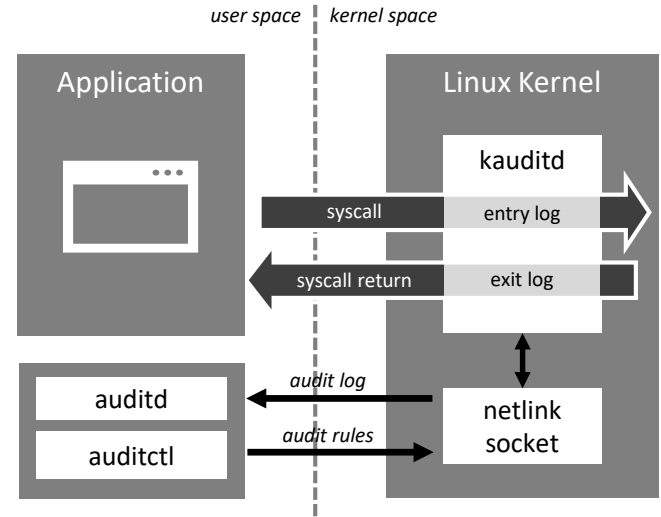
- Operates at the network layer
 - can monitor passive network traffic (to host ports not listening)
- Network-centric
 - no process/workload correlation
 - centralized collection points

NetFlow

```
- Start time:        03/25/2019T19:48:02.832226
- End time:          03/25/2019T19:48:12.823003
- TCP Flags          SA F
- Source IP address: 127.0.0.1
- Source Port:       40556
- Dest IP address:   127.0.0.1
- Destination Port:  8080
- Protocol:          TCP
- Bytes sent:        80
- Bytes rcvd:        80
- Packets sent:      2
- Packets rcvd:      2
```

How about Linux Audit?

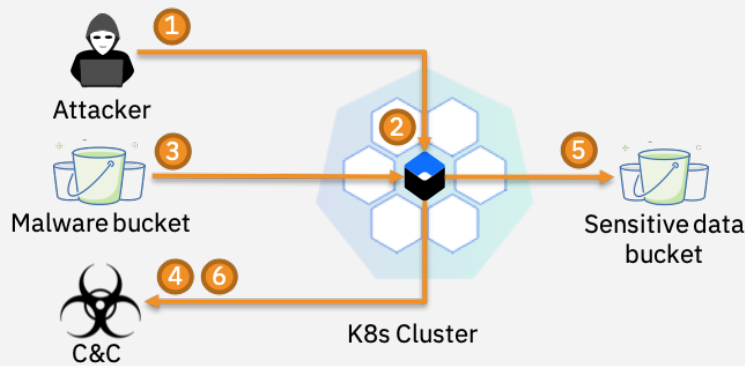
- Uses pre-configured rules to track system events
 - can be coupled with LSMs for **runtime monitoring**
- Lacks container-awareness
 - containers are user space constructs; kernel cannot track container provenance and actions
 - *nsID* proposal discarded; *container ID* RFE
- Does not support binary output formats
- Can suffer from log spills due to backlog queue limits
 - kernel backlog queue can be increased, but takes up kernel memory; difficult to monitor large process trees



Demo

Simplified attack kill chain

1. Perform reconnaissance on the cluster's public services and look for vulnerabilities
2. Exploit identified vulnerabilities to drop and run malicious code in one of the containers
3. The malicious payload downloads malware, installs and bootstraps it
4. The malware connects to the C&C and get instructions
5. The malware connects to a data store and retrieves sensitive data
6. The data is exfiltrated through the C&C



Thank you

github.com/sysflow-telemetry

—

sysflow-telemetry.slack.com

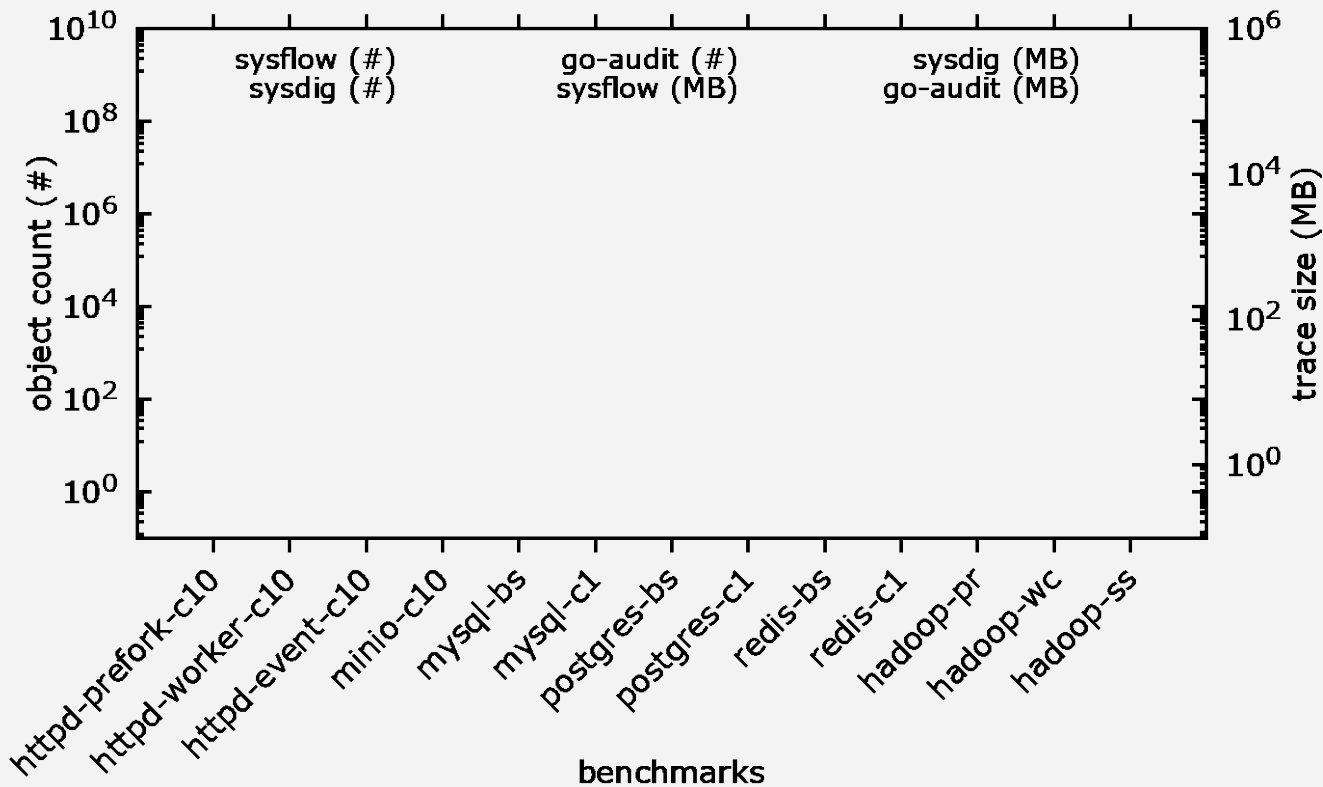
sysflow@us.ibm.com

SysFlow is an open source research project and not an IBM proprietary product. We hope to establish an open-source community around the project.

© Copyright IBM Corporation 2019. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and trademark information](#).



Compression Factors



Pretty-printed SysFlow trace (selected attributes):

Evt #	T	Process	PPID	PID	TID	Op	Flags	Start Time	End Time	FD	Ret	Resource	NOBRead	NOBWrite
0	PE	./filer	1887	21847	21847		EXEC	04/10/2019T16:47:14.717700				0		
1	FF	./filer	1887	21847	21847	O		04/10/2019T16:47:14.717796	04/10/2019T16:47:14.717804			/etc/ld.so.cache	0:0	0:0
2	FF	./filer	1887	21847	21847	O R	C	04/10/2019T16:47:14.717816	04/10/2019T16:47:14.717858		3	/lib/x86_64-linux-gnu/libc.so.6	1:832	0:0
3	FF	./filer	1887	21847	21847	O W	C	04/10/2019T16:47:14.718098	04/10/2019T16:47:14.718128		3	/tmp/tested_file.txt	0:0	1:31
4	FF	./filer	1887	21847	21847	O W	C	04/10/2019T16:47:14.718142	04/10/2019T16:47:14.718150		3	/tmp/tested_file2.txt	0:0	1:37
5	FF	./filer	1887	21847	21847	O W	C	04/10/2019T16:47:14.718163	04/10/2019T16:47:14.718170		3	/tmp/tested_file3.txt	0:0	1:37
6	FF	./filer	1887	21847	21847	O W	C	04/10/2019T16:47:14.718188	04/10/2019T16:47:14.718195		3	tested_file_test.txt	0:0	1:41
7	FE	./filer	1887	21847	21847		MKDIR	04/10/2019T16:47:14.718230				0 /tmp/testing_dir		
8	FE	./filer	1887	21847	21847		MKDIR	04/10/2019T16:47:14.718397				0 ./testing_dir		
9	FE	./filer	1887	21847	21847		RMDIR	04/10/2019T16:47:14.718616				0 /tmp/testing_dir		
10	FE	./filer	1887	21847	21847		RMDIR	04/10/2019T16:47:14.718775				0 ./testing_dir		
11	FE	./filer	1887	21847	21847		LINK	04/10/2019T16:47:14.719299				0 /tmp/tested_file.txt,./tested_file.txt		
12	FE	./filer	1887	21847	21847		SYMLINK	04/10/2019T16:47:14.719407				0 /tmp/tested_file2.txt,./tested_file2.txt		
13	FE	./filer	1887	21847	21847		RENAME	04/10/2019T16:47:14.719518				0 /tmp/tested_file3.txt,/tmp/tested_file4.txt		
14	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.719623				0 /tmp/tested_file.txt		
15	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.719738				0 ./tested_file.txt		
16	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.719845				0 ./tested_file2.txt		
17	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.719956				0 /tmp/tested_file2.txt		
18	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.720073				0 ./tested_file_test.txt		
19	FE	./filer	1887	21847	21847		UNLINK	04/10/2019T16:47:14.720186				0 /tmp/tested_file4.txt		
20	PE	./filer	1887	21847	21847		EXIT	04/10/2019T16:47:14.720320				0		