SysFlow Telemetry

Suyash Amatya, Nico Wang, Justin Kim, David Quintero, Felipe Alves, Sofia Carmona, Daien Miao, Reilly Prendergast

IBM and Rensselaer Polytechnic Institute

Introduction

SysFlow is a framework for monitoring cloud workloads and enhancing performance and security analytics. SysFlow revolutionizes system telemetry by capturing and analyzing a wide array of system events, providing invaluable insights into application behaviors and infrastructure dynamics within cloud environments.

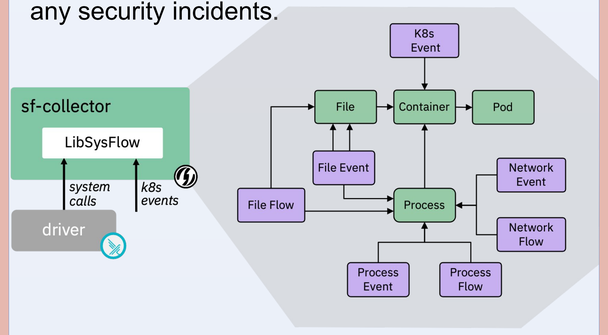
Our RCOS team tests its boundaries and leverages its capabilities to the fullest. We went from developing tutorials and conducting security analysis, to experimenting with exploit datasets and crafting visual representations of our findings, using tools such as Docker, Jupyter notebooks, and virtual machine softwares such as VirtualBox.

Red Team

The red team conducts data collection by simulating attacks on Linux VMs and gathering SysFlow traces, representing the offensive side of cybersecurity. Their aim is to mimic real-world cyber adversaries, identifying weaknesses in the organization's systems, networks, and applications using techniques like penetration testing.

Blue Team

The blue team is responsible for data analytics and visualization: developing analytics and visualization for threat hunting, representative of the defensive side of cybersecurity. The role of the Blue Team is to defend the organization's systems, networks, and data against cyber threats. Blue teams need to use various tools, technologies, and processes to detect, prevent, and respond to any security incidents.



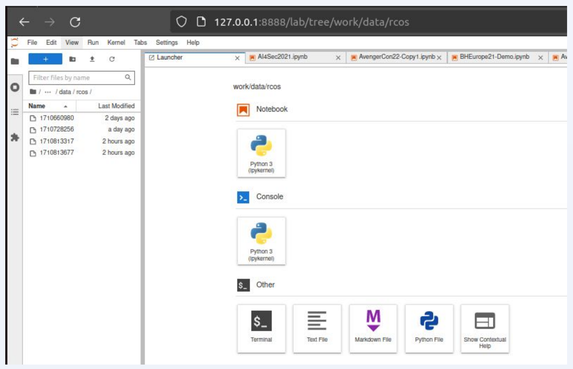
Setting Up

Setting up for this Sysflow project proved to be a difficult challenge, given our background in advanced operating systems concepts.

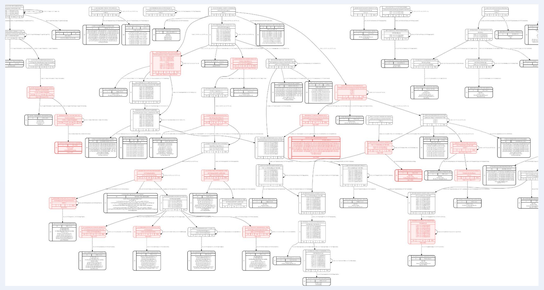
The red team started by setting up a Virtual Machine and used Docker containers to set up the environment for simulating attacks.

```
curl: (22) The requested URL returned error: 404
Unable to find a prebuilt falco eBPF probe
* Trying to compile the eBPF probe (falco_ubuntu-generic_6.5.0-21-generic_21~22.04.1.o)
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:
* eBPF probe located in /root/.falco/5.1.0+driver/x86_64/falco_ubuntu-generic_6.5.0-21-generic_21~22.04.1.o
* Success: eBPF probe symlinked to /root/.falco/falco-bpf.o
Configured filter: container.name=httpd
I20240319 03:02:10.230751     1 sysflowcontext.cpp:246] Opening ebpf driver in flow mode monitoring 92 system calls.
I20240319 03:02:11.104127     1 sysflowcontext.cpp:89] Starting dropping mode with sampling rate: 1
I20240319 03:02:11.104339     1 sysflowcontext.cpp:102] Enabled file only mode
I20240319 03:02:11.104377     1 sysflowcontext.cpp:113] Enabled process flow mode
I20240319 03:02:11.104382     1 sysflowcontext.cpp:130] File Read Mode was set to: 2 Modes are:  0 = enable all file r
, or 2 = disable file reads to certain directories
I20240319 03:02:11.104393     1 sysflowprocessor.cpp:43] File writer loaded.
I20240319 03:02:11.113209     1 main.cpp:276] Starting the SysFlow Collector...
```

The blue team started by setting up Docker and the sfnb Jupyter environment, used for threat hunting and data analysis. After red team sends the blue team a trace file, they created visualizations in python notebooks.

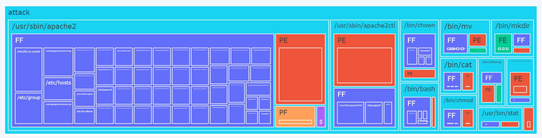


Shell Shock Trace

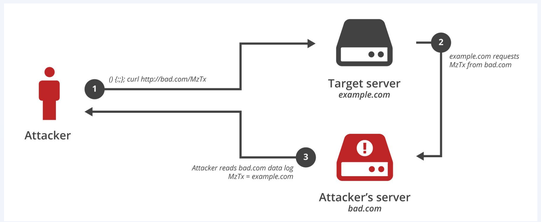


The team used the Shellshock reverse shell exploitation (CVE-2014-6271), which uses Remote Code Execution in the BASH shell. SysFlow is used to trace its execution path, using the mitre database to look for and highlight in red any adversary tactics, techniques, and procedures (TTPs), detecting malicious activity.

Shell Shock Trace (cont.)



These are data streamed to Jupyter from the SysFlow generated file. SysFlow monitors different components of the system, such as processes, files, network connections, and system calls and aggregates them into a single file (trace).



Methods

To run the Shellshock attack, first we had to set up the Jupyter Notebook by running the Sysflow Notebook command, so we can send the Blue team the collected traces from the attack. Then, we ran a command to start the vulnerable environment in a specified docker container, which is where we are going to run our attack in. Next, we run the Sysflow Collector to start scanning the container, and in another terminal we type in the command to execute the attack. After waiting for a couple of seconds, we can go back and stop the Sysflow collector to finish gathering traces. Finally, if we view the docker logs for the Sysflow Notebook, we can get the link to open and share the Jupyter notebook with the Blue Team, so they can see and analyze the attack traces in the linked notebook.

Next Steps

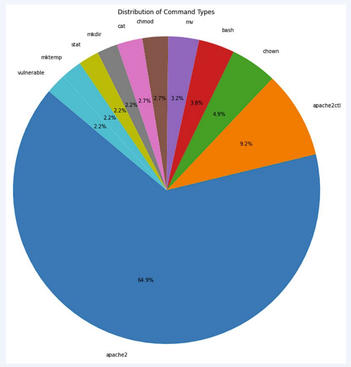
We plan to create traces for these attacks:

- More advanced Shellshock attacks
- CVE-2016-6515 (CPU consumption Denial of Service)
- CVE-2017-7494 (SambaCry)

Results (Progress)

At the start of the semester both teams focused on going through the Jupyter Notebooks and understanding how Sysflow works. We then documented our set up process on the Sysflow github. Throughout the semester we were able to generate new attacks and test Sysflow's ability to trace them. The Red Team successfully executed the Shell Shock attack and the Blue team was able to create visuals from it.

Conclusion



Our RCOS project is a step towards creating a safer and more secure future in cybersecurity. SysFlow's robust monitoring system can aid in creating signatures for Intrusion Detection Systems to more easily track common vulnerabilities with no need for complex behavioral analysis. Through this project, we hope to contribute to the ongoing effort of fortifying digital infrastructures against emerging threats, paving the way for a more resilient and secure digital ecosystem.

References

https://colab.research.google.com/github/sysflow-telemetry/sf-lab/blob/main/pynb/LFOSSNA23.ipynb
https://blog.cloudflare.com/inside-shellshock
https://hub.docker.com/u/vulnerables

Acknowledgements

Frederico Araujo, and Teryl Taylor