How Google YARA-L Works with Sigma and Other Community Standards

Dr. Anton Chuvakin

Outline - 15 MINUTES

- In brief - modern detection challenges
- Standards … what standards?
- YARA-L and Friends
  - YARA-L + Sigma
  - YARA-L + ATT&CK
- Future

Can Standards Help Detection?

So, Why Is Detection So Hard?

- Today's environments are complex and messy
- Detection needs **PEOPLE**
  - People are hard to scale
- Detection needs **DATA**
  - Data comes from many sources, owned by many people
  - **NEW**: Context is especially key
- Detection need **TRIAGE** (rules or ML)
- And, eh, the attackers don't want to be detected :-)
  - Well, some of them (ransomware doesn't, naturally)
- **NEW**: Detection may be about intent, not only activity

Anton's Why Is Threat Detection Hard?

Sure, but are standards the answer?

Remember CEE (cee.mitre.org) 

Why YARA-L, The Detection Language

- **Broad coverage** of security telemetry
- Efficient for **real-time and historical** analytics
- **Smart matching logic** to match to enriched data “automagically”
- **Focused language** for threat detection, not for querying the data
- **Declarative** approach, let the system “think”
- Relies on **YARA experience**

YARA-L Example

```
rule mitre_attack_T1548_002_windows_uac_bypass
{
 meta:
   author = "Google Cloud Security"
   description = "Net use commands for SMB/Windows admin shares"
   reference = "https://attack.mitre.org/techniques/T1021/002/"
   yara_version = "YL2.0"
   rule_version = "1.0"
   mitre = "T1548"

 events:
   (
     $e1.metadata.event_type = "PROCESS_LAUNCH" and
     re.regex($e1.principal.process.command_line, `reg\.exe add hkcu\\software\\classes\\mscfile\\shell\\open\\command /ve /d.* /f`) nocase
   )
   or
   (
     re.regex($e1.principal.process.command_line, `powershell.exe`) nocase and
     re.regex($e1.target.registry.registry_key, `\\software\\classes\\mscfile\\shell\\open\\command`) nocase
   )
 condition:
   $e1
}
```

YARA-L from Sigma Examples

```
rule the_gocgle_malicious_campaign {
meta:
   author = "Osman Demir"
   description = "Detects Web Skimming Attacks  License: https://github.com/Neo23x0/sigma/blob/master/LICENSE.Detection.Rules.md."
   reference = "https://tdm.socprime.com/tdm/info/LI0qbtGS0Jtv"
   version = "0.01"
   created = "2021-03-09"
   category = "proxy"
   mitre = "T1056, Collection"

 Events:
(($selection.src.hostname = "gocgle-analytics.net" or $selection.src.hostname = "googlo-analytics.com" or $selection.src.hostname = "gocgletagmanager.com" or
$selection.src.hostname = "googlc-analytics.com" or $selection.src.hostname = "gocgle-analytics.cm" or $selection.src.hostname = "gocgletagmanager.cm" or
$selection.src.hostname = "gocgle-analytics.com" or $selection.src.hostname = "analytic.is" or $selection.src.hostname = "qdtf54y6eu7i87t.ga") or
($selection.target.ip = "5.188.9.61" or $selection.target.ip = "5.188.9.33" or $selection.target.ip = "5.188.9.40" or $selection.target.ip = "194.180.224.112"))


 condition:
   $selection
}
```

YARA-L + ATT&CK

- YARA-L content maps to ATT&CK (duh!)
- ATT&CK used to guide detection development to cover detection gaps
- ATT&CK is used to develop cloud detections and telemetry to power them
- “Detection in depth” planning using ATT&CK
  - E.g. if we missed it in real time, what can help us detect it historically?

YARA-L + Sigma

- YARA-L content can be created from Sigma content
  - Examples: https://github.com/chronicle/detection-rules/tree/main/soc_prime_rules
- Reuse content from public repositories
- Simplify product migration (via Sigma conversion)

Future Plans

- More mapping
- More analytics of mapping and coverage
- More conversions from Sigma
- … and of course more fun detection content for Chronicle and other GCP Security capabilities

Resources and Contacts

- YARA-L Repo
- Chronicle SOC Prime rules repository
- YARA-L Whitepaper
- @anton_chuvakin
- www.chuvakin.org
- anton@chuvakin.org