Data Log Management for Cyber-Security Programmability of Cloud Services and Applications

ALESSANDRO CARREGA, MATTEO REPETTO
CNIT, NATIONAL LABORATORY OF SMART AND SECURE NETWORKS (S2N), GENOA (ITALY)
ALESSANDRO.CARREGA@CNIT.IT, MATTEO.REPETTO@CNIT.IT
Outline

- Cyber-security challenges
- New computing paradigms and software orchestration
- Concept and framework
- ASTRID architecture
- The Context Fabric and Context Abstraction
- Preliminary results
- Conclusions
Cyber-security challenges

Visibility
- dynamic and evolving systems
- partially unknown topologies

Efficiency
- limited computing resources
- overlapping appliances

Humans in the loop
- slow or ineffective reaction
- lack of skills/expertise/care
- manual and error-prone processes

Targets
- data
- service providers

Weak links in the chain
- social engineering, IoT
- outdated sw, weak configurations

Attacks
- stealthy and persistent
- multi-vector
- AI
New computing paradigms

Proliferation of cloud providers and services
New computing paradigms

Heterogeneity of infrastructures and services
- no common and uniform Security-as-a-Service APIs in CMS
- low cross-cloud interoperability

Emergence of software orchestration paradigms
- more agility for security management
Service orchestration

Elasticity

Redundancy

Resilience
Concept

Context fabric

Adaptation through programmability

Programmatic access to security context

Detection and analysis
The Context Fabric
The Context Fabric - Implementation

**Elastic Stack** for collecting and storing data
- Beats for **capturing the context**
  - standard beats: FileBeat, MetricBeat, PacketBeat
  - enhanced beats (work in progress): CubeBeat, SysBeat, NetdataBeat
- Logstash for **local processing pipelines**
  - aggregation, filtering, timestamp insertion, geolocation, anonymization, removal of private and sensitive data, fingerprinting, serialization, name resolution, classification, encryption, parsing, and creation of structured data, ...
- Logstash for **central processing pipelines**
  - timestamp insertion, fingerprinting, aggregation, ...
- Elasticsearch for **metadata**
- Time series DB for **historical data**
The Context Fabric - Implementation

Kafka for **delivering real-time data** to multiple consumers
- detection algorithms, forensics DB, ElasticTree, ...

Context Manager for **programming local agents**

Polycube as **Local control framework**
- changes the configuration of inspection tools
  - beats, scripts,
- injects code
  - eBPF programs, Logstash pipelines
Context Abstraction

Data consumers
- Firewall
- Antivirus
- IPS

API

Virtual Function i

Security Context Abstraction

Virtual Function k

Execution Environment
- Logs
- Metrics
- Mgmt actions

Applications

System libraries, daemons

Local Security Agent
- Logs
- Mgmt actions
- System calls
- DPI
- Packet filtering

OS Kernel
Context Abstraction – API

REST API to access the context:
- service topology
- VMs, links, OS, libraries
- local agents
- configurations, commands
- data
  - origin, timestamp, location

```
/rest_api
  /exec-env <id>
  /exec-env-type <id>
  /network-link <id>
  /connection <id>
  /agent <id>
  /config <id>
  /network-link-type <id>
  /catalog <id>
  /data <id>
```

15/11/2019
CYSARM - LONDON, UK
Preliminary results

Overhead on service execution:
- CPU, memory, network

Parameters:
- workload
  - 10, 100, 500, 1000 1/s
- polling time
  - 1, 50, 500, 1000 s

<table>
<thead>
<tr>
<th>Function</th>
<th>Operating System</th>
<th>vCore</th>
<th>vRAM</th>
<th>Beat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Debian 4.9</td>
<td>1</td>
<td>1 GB</td>
<td>Filebeat</td>
</tr>
<tr>
<td>MySQL</td>
<td>Debian 4.9</td>
<td>1</td>
<td>1 GB</td>
<td>Metricbeat</td>
</tr>
<tr>
<td>mini_httpd</td>
<td>Ubuntu 18.04.2 LTS</td>
<td>2</td>
<td>2 GB</td>
<td>Cubebeat</td>
</tr>
<tr>
<td>CB</td>
<td>Debian 4.9</td>
<td>4</td>
<td>4 GB</td>
<td>—</td>
</tr>
</tbody>
</table>
Preliminary results – CPU usage

Total CPU usage by local agents (Beat + Logstash)
Preliminary results – CPU usage

Breakdown of CPU usage by local agents (Beat, Logstash)
Preliminary results – Memory usage

Breakdown of memory usage by main process (Apache, MySQL, mini_httpd), Logstash and Beats
Preliminary results – Network

Bandwidth usage (Context collection/total)
Preliminary results – Network

Breakdown of latency

Source

Beat ➔ Logstash ➔ Kafka ➔ Logstash ➔ Elasticsearch

- File to Beat
- Beat to Logstash
- Logstash to CB

Latency [s]

Poll: 1 s  Poll: 5 s  Poll: 10 s  Poll: 20 s
Number of requests [s⁻¹]

a) Apache

Poll: 1 s  Poll: 5 s  Poll: 10 s  Poll: 20 s
Number of requests [s⁻¹]

b) MySQL

c) mini_httpd

15/11/2019
CYSARM - LONDON, UK
Conclusions

Large workloads may lead to excessive overhead

- File collection is the most critical operation (frequent disk access)
- Logstash operation should be optimized (especially memory)
- eBPF filters have very low overhead

Low dependency on the polling interval

Next steps:

- development of the control part
- evaluate the delay of control commands
- integration in the ASTRID Use Cases
**Project acronym** | ASTRID  
---|---  
**Project ID** | 786922  
**Starting Date** | 1st May 2018  
**Ending Date** | 30th April 2021  
**Call** | H2020-DS-2016-2017  
**Total Cost** | EUR 2,932,297.50  
**EC Contribution** | EUR 2,932,297.50  
**Funding scheme** | RIA
Thank you!

M. Repetto: matteo.repetto@cnit.it