Breach → ATT&CK → OSQUERY

Learning from Breach Reports to Improve Cross-platform Endpoint Monitoring
I'm Guillaume Ross

Uptycs - security analytics at scale using osquery

@gepeto42 on Twitter
OVERVIEW

- Too many « best practices » unrelated to real attacks
- MITRE ATT&CK is based in reality
- Review a true attack with it
- Improve detection with osquery
- Monitor defensive controls
PEW PEW MAPS?

Nope
Nahh...

Osquery 101
Cross-platform
Mac, Linux, Windows, Docker APIs

229 Tables
That's a lot of tables

100%
Amazing and open
OSQUERY: TRANSFORM YOUR OS INTO A VIRTUAL DB

- Uses SQLite for query execution
- Executes scheduled queries, real-time queries, and has an interactive mode
- Cross-platform
- Fast, reliable
- Flexible
- Originally from Facebook
LET’S REVIEW SOME CONCEPTS

Query

Query Pack

Outputs

Flags

Tables

Events
EXAMPLE ABSTRACTION

- ps ax | grep apache
- SELECT * FROM processes WHERE name LIKE ‘%apache%’;

No huge DB built over time on the agent. When queried, info is obtained (Plus: Streams of Events).
TWO DETAILED REPORTS

SingHealth
- Lots of details.
- Lateral movement.
- App servers, DBs, flat networks
- Reads like a novel

Equifax
- Vulns
- Management Structure
- We will use this one less.
SingHealth Full Report


Nahh it’s not malware*
ATT&CKING The SingHealth Breach


Great Article by @mattnotmax
Figure 7: Key events of the Cyber Attack

1. Initial entry (23 August 2017)
2. Lateral movement & privilege escalation (December 2017 – June 2018)
3. Compromised SCM (26 June 2018)
4. Queried SCM Database (27 June 2018 – 4 July 2018)
5a. Data transferred (27 June 2018 – 4 July 2018)
5b. Data transferred (27 June 2018 – 4 July 2018)
Initial Entry

- Two workstations - one decommissioned
- Second one makes C2 callback on August 24, 2017
- Powershell (T1086) / User Execution (T1204)
- Remnants of a “public RAT” found
- Password dumping capabilities
- Possibly T1192 or T1193 - spear phishing link or attachment - but indications this might be Ruler exploit
Initial Entry

- `Socket_events` or `Process_open_sockets` against threat intel feeds
- PowerShell logging with osquery (easy!)
- Monitor process / parents
- Office Hardening (Unclear - Possibly if phishing was the entry point)
Process_Open_Sockets

- `select * from process_open_sockets;`
- Match `remote_address` against threat lists in logs.
- Take specific attention to ports like 445, 3389, including laterally - useful to detect weak FW config against lateral movement.
PowerShell - T1086 - T1204

- Script block logging: ON
- `select * from powershell_events;`
- Sends back results over TLS.
- Consider using `cosine_similarity`
- Run queries for obvious bad keywords
  - Specific tools: gentilkiwi, mimikatz, Invoke-Shellcode (Powersploit)
  - Methods: SE_PRIVILEGE_ENABLED, TOKEN_PRIVILEGES, GetDelegateForFunctionPointer etc.
<table>
<thead>
<tr>
<th>COLUMN</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>BIGINT</td>
<td>Timestamp the event was received by the osquery event publisher</td>
</tr>
<tr>
<td>datetime</td>
<td>TEXT</td>
<td>System time at which the Powershell script event occurred</td>
</tr>
<tr>
<td>script_block_id</td>
<td>TEXT</td>
<td>The unique GUID of the powershell script to which this block belongs</td>
</tr>
<tr>
<td>script_block_count</td>
<td>INTEGER</td>
<td>The total number of script blocks for this script</td>
</tr>
<tr>
<td>script_text</td>
<td>TEXT</td>
<td>The text content of the Powershell script</td>
</tr>
<tr>
<td>script_name</td>
<td>TEXT</td>
<td>The name of the Powershell script</td>
</tr>
<tr>
<td>script_path</td>
<td>TEXT</td>
<td>The path for the Powershell script</td>
</tr>
<tr>
<td>cosine_similarity</td>
<td>DOUBLE</td>
<td>How similar the Powershell script is to a provided 'normal' character frequency</td>
</tr>
</tbody>
</table>
**PowerShell - T1086 - T1204**

```sql
SELECT datetime, script_name, script_path, script_block_count FROM powershell_events;
```

- Monitor script_block_count trends
Processes and Process_events include parent PID.
Quick script to create process tree on Uptycs Blog
Query and join the table with itself to get the process name.
Then, query for weirdness...
Parent Process

- Select * from processes;
- Subqueries are your friends
- Select the processes table, for specific process names, look at the parent, see what the parent is, log when unexpected. For example:

```sql
osquery> select pid, name, path, parent from processes where name='services.exe'
```

```
<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>path</th>
<th>parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>568</td>
<td>services.exe</td>
<td>476</td>
<td></td>
</tr>
</tbody>
</table>
```
Parent Process

What's 476? Wininit. Expected!

```
osquery> select pid, name, path, parent from processes where pid=476;

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>path</th>
<th>parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>476</td>
<td>wininit.exe</td>
<td>400</td>
<td></td>
</tr>
</tbody>
</table>
```
Parent Process

```
SELECT name as bad_parent_child_name, pid bad_parent_child_pid
FROM processes WHERE
pid=(SELECT parent FROM processes WHERE parent!=
(SELECT pid from processes where name='wininit.exe')
AND LOWER(name)='services.exe')
OR pid=(SELECT pid FROM processes WHERE parent!=
(SELECT pid from processes where name='wininit.exe')
AND LOWER(name)='services.exe');
```

This example + MANY more can be found on Filippo Mottini’s GitHub:

Office Hardening - Potential T1192/1193

- The registry table is our friend.
- \texttt{User\software\policies\microsoft\office\16.0\App}

- WILDCARDS!
SELECT * FROM registry WHERE key LIKE 'HKEY_USERS\%\Software\Policies\Microsoft\office\16.0\%\security';

% = wildcard
Office Hardening - Potential T1192/1193

- Not easy to see what user that is...
- But.. the SID is in the registry path!
- Join some subqueries.
- Split data on backslashes.
  - (It is Windows, after all!)
Office Hardening - Potential T1192/1193

```sql
SELECT username, data, split(path, '\', 1) AS sid
FROM
(SELECT data, path FROM registry
WHERE key LIKE 'HKEY_USERS\\SOFTWARE\Policies\Microsoft\office\16.0\\security' AND name='blockcontentexecutionfrominternet')
JOIN users ON users.uuid = sid;
```

<table>
<thead>
<tr>
<th>username</th>
<th>data</th>
<th>sid</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>1</td>
<td>S-1-5-21-2940226973-2973024380-1756164060-1001</td>
</tr>
</tbody>
</table>
Office Hardening - Potential
T1192/1193

● If it was Ruler:
  ○ Use same technique to check that EnableUnsafeClientMailRules is not equal to 1 (disables the patch against malicious rules/code exec from rules)
  ○ Use programs table to ensure Outlook is 16.0.4534.1001 or above
Lateral movement & privilege escalation

- Over a period of +6 months
- Spread malware over shares (T1077 and T1105)
- Weak Local Admin credentials + stored in clear (T1078 + T1081)
- Lateral movement + malware spread using shares, remote desktop/Citrix (T1076)
- Attempts to access DB from Citrix with Domain Admin failed
- Exploit SCM app to get service account
Lateral movement & privilege escalation

- Monitor lateral movement
  - Windows: 445, 3389
  - Mac/Linux: 22
- Ensure LAPS is deployed and effective
- Identify accounts logging in “wrong”
- SQL Audit logs via osquery?
SELECT *
FROM process_open_sockets
WHERE local_port=445
AND remote_address NOT LIKE '10.0.99%'
AND remote_address NOT LIKE '0.0.0.0'
AND remote_address NOT LIKE '::';
235. As explained above, the password to the L.A. account was ‘P@ssw0rd’, which is easily cracked, and it is possible that the attacker gained control over the account by cracking the password. The L.A. account was also considered a ‘dormant’ account, which meant that it was an account that has been used before, but has not been logged into for the last 183 days.²³
Local Admin - T1078/1081

Start with this..

```sql
SELECT data, path FROM registry
WHERE key = 'HKEY_LOCAL_MACHINE\Software\Policies\Microsoft Services\AdmPwd';
```
Or we can track a specific value as a maximum:

```sql
SELECT data, path FROM registry
WHERE key LIKE 'HKEY_LOCAL_MACHINE\Software\Policies\Microsoft Services\AdmPwd'
AND name='PasswordLength'
AND data < 31;
```
27.2 Detecting an active login to Citrix Server 2 and disabling the S.A. account on the morning of 5 July 2018

523. Midway through the meeting on the morning of 5 July 2018, Joanne noticed that there was a ‘live’ active session by the S.A. account connecting to Citrix Server 2 via RDP from VM 1. Lum observed that:

(a) VM 1 was the same hostname that was discovered to have accessed Citrix Server 1 using the L.A. account as early as 8 June 2018; and

(b) the S.A. account ought not to have the privileges to login to the server following its removal from the administrator group on 26 June 2018.
Lateral Movement / Account Abuse

Use `logon_sessions` to track powerful sessions.
Lateral Movement / Account Abuse

Leverage naming conventions...

<table>
<thead>
<tr>
<th>user</th>
<th>upn</th>
<th>logon_domain</th>
<th>logon_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td><a href="mailto:bob@robotnik.labs">bob@robotnik.labs</a></td>
<td>ROBOTNIK</td>
<td>Interactive</td>
</tr>
<tr>
<td>DWM-4</td>
<td></td>
<td>Window Manager</td>
<td>Interactive</td>
</tr>
<tr>
<td>DWM-4</td>
<td></td>
<td>Window Manager</td>
<td>Interactive</td>
</tr>
<tr>
<td>UMFD-4</td>
<td></td>
<td>Font Driver Host</td>
<td>Interactive</td>
</tr>
<tr>
<td>g</td>
<td>WIN10-HARDEN$@robotnik.labs</td>
<td>WIN10-HARDEN</td>
<td>Interactive</td>
</tr>
<tr>
<td>g</td>
<td></td>
<td>WIN10-HARDEN</td>
<td>Interactive</td>
</tr>
<tr>
<td>DWM-3</td>
<td></td>
<td>Window Manager</td>
<td>Interactive</td>
</tr>
<tr>
<td>DWM-3</td>
<td></td>
<td>Window Manager</td>
<td>Interactive</td>
</tr>
<tr>
<td>UMFD-3</td>
<td></td>
<td>Font Driver Host</td>
<td>Interactive</td>
</tr>
<tr>
<td>bob</td>
<td>WIN10-HARDEN$@robotnik.labs</td>
<td>ROBOTNIK</td>
<td>Interactive</td>
</tr>
<tr>
<td>g</td>
<td>WIN10-HARDEN$@robotnik.labs</td>
<td>WIN10-HARDEN</td>
<td>Interactive</td>
</tr>
<tr>
<td>g</td>
<td></td>
<td>NT AUTHORITY</td>
<td>Service</td>
</tr>
<tr>
<td>LOCAL SERVICE</td>
<td></td>
<td>ROBOTNIK</td>
<td>Service</td>
</tr>
<tr>
<td>WIN10-HARDEN$</td>
<td></td>
<td>Font Driver Host</td>
<td>Interactive</td>
</tr>
<tr>
<td>UMFD-0</td>
<td>WIN10-HARDEN$@robotnik.labs</td>
<td>Undefined Logon Type</td>
<td>Interactive</td>
</tr>
<tr>
<td>WIN10-HARDEN$</td>
<td>WIN10-HARDEN$@robotnik.labs</td>
<td>ROBOTNIK</td>
<td>Undefined Logon Type</td>
</tr>
</tbody>
</table>
Lateral Movement / Account Abuse

So if your service accounts were say: *svc_servicename* - you could query. Do the opposite for non-svc accounts!

```sql
SELECT upn, logon_domain, logon_type FROM logon_sessions
WHERE upn LIKE 'svc_%'
AND logon_type = "Interactive";
```
SQL Auditing

- Ensure SQL can write to Event Logs (Application or Security, or anything else)
- Look for login related event IDs such as 18456.
- Can be done for all Windows logs too
  - Selectively centralize workstation logs over TLS!

```sql
SELECT * FROM windows_events
WHERE eventid='XXXXXX';
```
Logging in general

- Apple Logging System (ASL) also supported.
- Syslog
- Selinux events
- Great when used with sysmon
- One of my favorite ways of selectively picking logs to centralize on workstations via TLS
MAP ATT&CK TO YOUR OSQUERY CONFIG

As shown, a few queries can help match entire techniques from ATT&CK.
READ IR REPORTS

There's always a tidbit you can use to improve your osquery deployment.
NOTHING BEATS OPEN SECURITY DATA FOR FLEXIBILITY AND MULTI-PLATFORM SUPPORT
THANKS!

Any questions?

You can find me at:
@gepeto42
gross@uptycs.com (I KNOW)