october 21, 2022

Playing Cat + Mouse with the Attacker

Item Set Mining in the Registry

Maeve Mulholland
Director of AI

Authors: Maeve Mulholland, Fred Frey, Tim Nary

Today, we'll discuss...

- The Trouble with Cyber Data
  - → Datasets for Machine Learning
  - → Using Data for Threat Detection
- Strategies for Using Cyber Data
  - → SnapAttack's Datasets
  - → Graph Analysis
- Catching the Mouse
  - → Scenario
  - → Hunting in the Registry
- Our Catch
  - → Results + Lessons Learned

Part 01.

The Trouble with Cyber Data

01. the trouble with cyber data

Datasets for Machine Learning

using ml for image recognition

Tesla's Self-Driving Vehicle - How it Works

01. collect all inputs
Collect many photos from every angle (including things like partial and obscure stop signs)

02. weed out variables
Represent and simulate all different lighting schemes

03. classify + label
Label all examples

04. test, tune, + improve
Monitor and collect failures to correct models

`ml for cyber → a different game`

But what if the stop sign had legs?

And there were like... four of them.

ML in cyber → **Stop signs with legs.**

01. collect all inputs → easier said than done
~~Collect many photos from every angle~~ → Sparse set of photographs that don't represent all conditions a stop sign may be found in. Oh, and the stop signs hide from the photographers.

02. weed out variables → hard to figure out which variables to look for
~~Represent and simulate all different lighting schemes~~ → It did a somersault—no one knew they could do that!

03. classify + label → hard to classify if you don't know what you're looking at
~~Label all examples~~ → Difficult to label the images, because no one really knows what they look like

04. test, tune, + improve → hard to measure and therefore, hard to improve
~~Monitor and collect failures to correct models~~ → Few opportunities for monitoring success and failure of a model

This is why ML for threat detection is hugely challenging.

Most ML efforts in the domain have drifted away from supervised methods.

`to build a classifier for a threat…`

Imbalanced Data: Voluminous data with threat examples being very rare

Examples are Context Dependent: Threats may look different in different environments

True Variance: True variance is hard to represent in datasets

Labeling: Labeling data requires expertise needed elsewhere

Measurement: Monitoring performance requires solving the problem some other way

Unknown Unknowns: Often, we do not know what we're looking for

SNAPATTACK

01. the trouble with cyber data

Using Data for Threat Detection

The Challenge

Many of the challenges faced by a SOC are the **same challenges blocking the construction of a dataset** for supervised ML in threat detection.

Process Is Adversarial

As detection methods are developed, hackers develop new methods of evasion.

Voluminous Unique Data

"Drinking from a firehose" / alert fatigue is a constant refrain, yet actual threats are rare. And, every environment is different and detecting a threat requires understanding the context it appears in.

Expertise Shortage

It's rare to find talent that can immediately analyze and understand threats and write heuristics for detecting them.

Are We Protected?

Monitoring and evaluating performance requires a way to understand what you're missing
Minimizing the unknown-unknowns and understanding the false negatives

part 02.

Strategies for Using Cyber Data

02. strategies for using cyber data

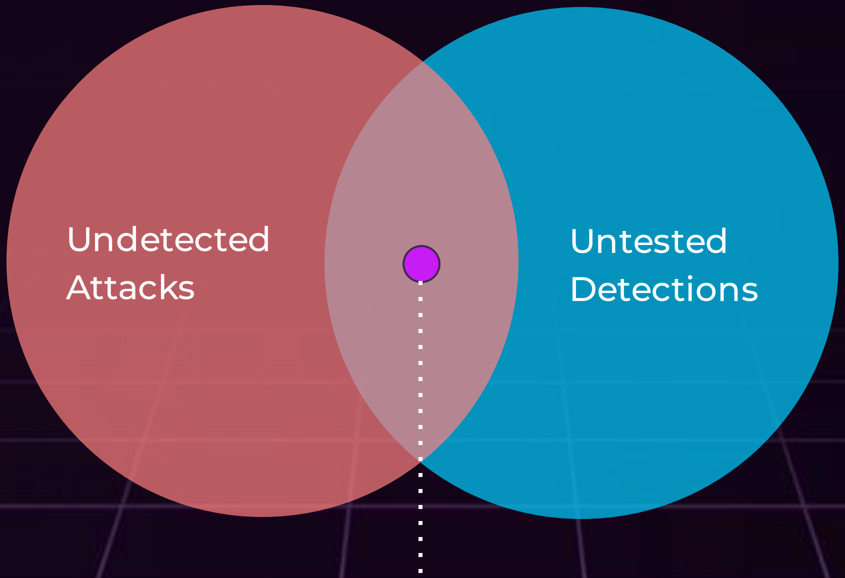
SnapAttack's Data Sets

SnapAttack’s Data Strategy

SnapAttack's goal is to give the advantage to the defender by building a dataset of contextualized and labeled attacks in a digestible format, and to collide them with detection analytics built by expert threat hunters.

Attack Library

Logged VM sessions with attack labeled by process or timestamp



Validated Detections of Attacks

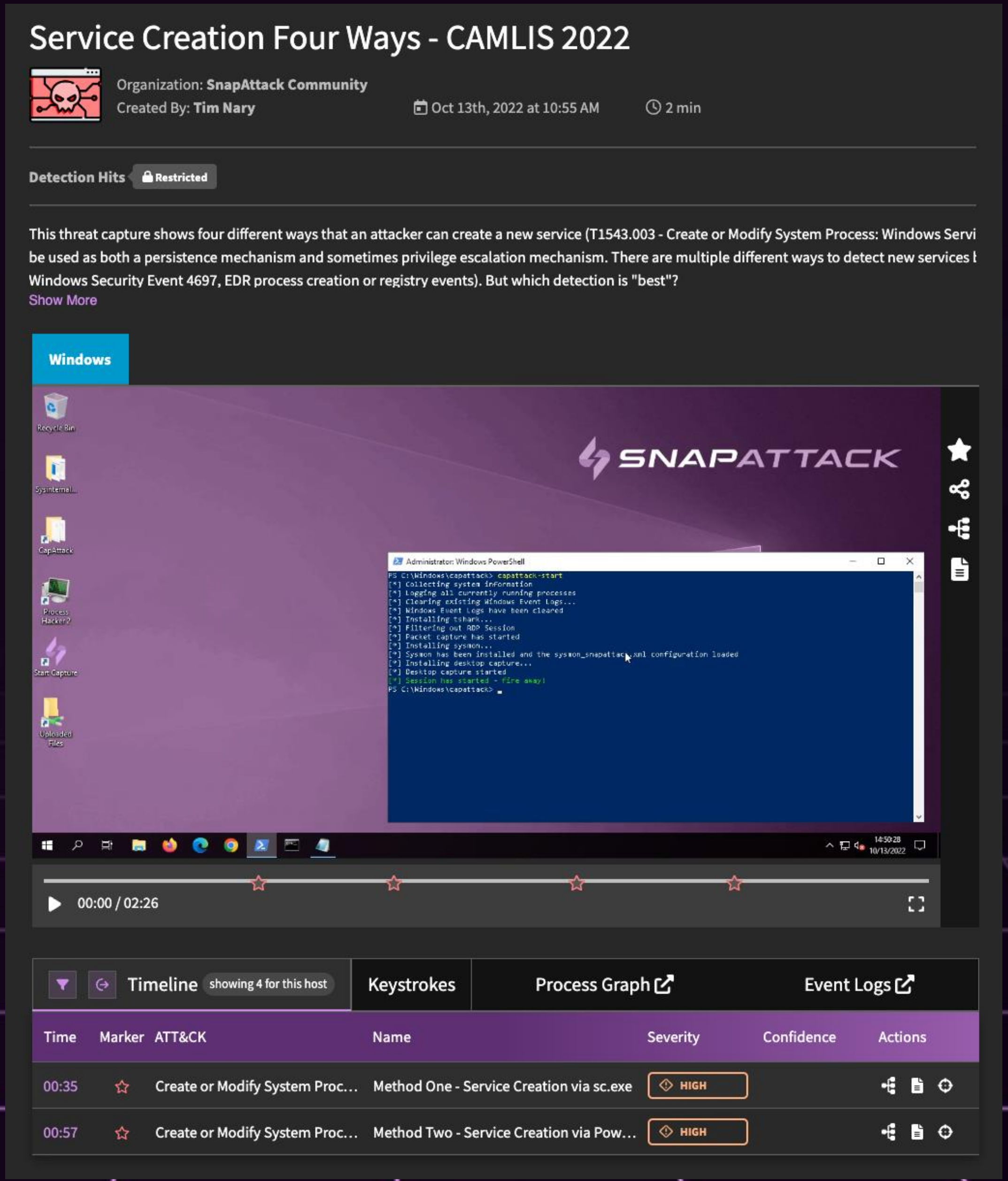
Detection Repo

Behavioral detection queries written in Sigma

SnapAttack's Threat Capture

Attack Library

Logged VM sessions with attack labeled by process or timestamp




```
Free Community Edition
www.snapattack.com/community

Service Creation Four Ways
app.snapattack.com/threat/WVDbr
```

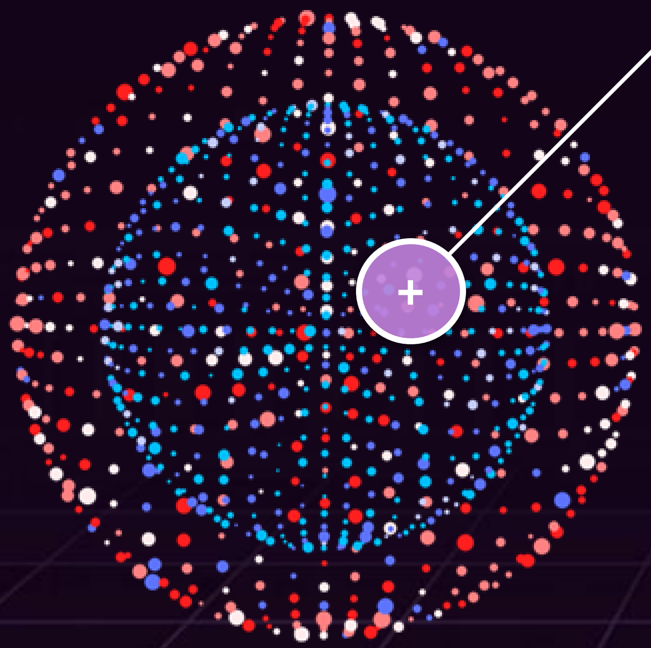
02. strategies for using cyber data

Graph Analysis

SNAPATTACK

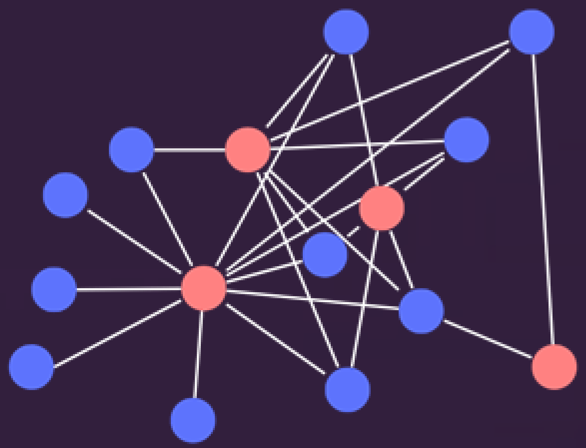
Data Structure

SNAPATTACK DATASET



STICKY KEYS SUBGRAPH

ATT&CK T1546.008



Subgraph of "Sticky Keys" exploits and related detection analytics. Extracted from main graph structure via community detection.

Detection Analytics (4,802)

Attack Instances (2,266)

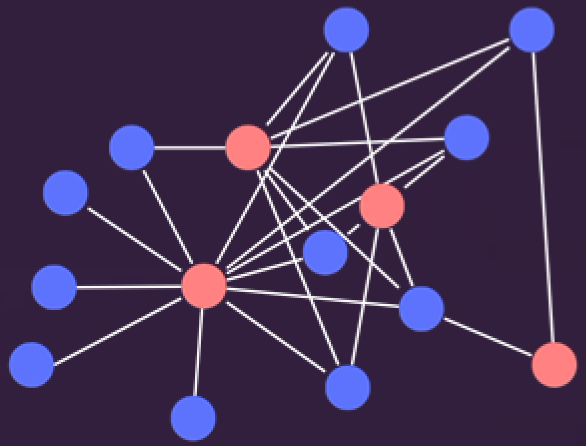
Validated Hits

Data Structure

- Detection suite extraction with community detection
- Coverage calculations for ATT&CK cells
- Similarity calculations
- Graph features for ML

STICKY KEYS SUBGRAPH

ATT&CK T1546.008



Subgraph of "Sticky Keys" exploits and related detection analytics. Extracted from main graph structure via community detection.

Detection Analytics (4,802)

Attack Instances (2,266)

Validated Hits

part 03.

Catching the Mouse

03. catching the mouse

The Scenario

How do we use this data to automate threat detection development?

scenario

Malicious Service Creation

"If your detection's goal is to identify malicious scheduled task creation, then you must first be able to identify **ALL** scheduled task creation."

Jared Atkinson
Playing Detection with a Full Deck
source

scenario

Malicious Service Creation

Creation of a schedule service requires the creation of this key:

`HKLM\System\CurrentControlSet\Services\<ServiceName>`

Robust Against Environment Changes

Registry events are common to all windows environments

Must Handle Variance Introduced by the Adversary (mouse can't hide)

Registry events are fundamental to the OS; many actions are inextricable from their associated key

Age old problem, but the detections are brittle.

Training Does not Require Additional Labeling

We have time stamps and sysmon registry events for all attacks

scenario

Malicious Service Creation

Creation of a schedule service requires the creation of this key:

`HKLM\System\CurrentControlSet\Services\<ServiceName>`

Service Creation Variations

- `sc.exe create`
- `PowerShell New-Service cmdlt`
- `SharePersist.exe`
- `WMI`

The Big Question

Can we create a system that will learn a "base condition" given the data we have available?

Service Creation Four Ways
app.snapattack.com/threat/WVDbr

03. catching the mouse

Hunting in the Registry

Creating Registry Data

Service Creation Method

- sc.exe create
- PowerShell New-Service cmdlt
- SharPersist.exe
- WMI

Variance

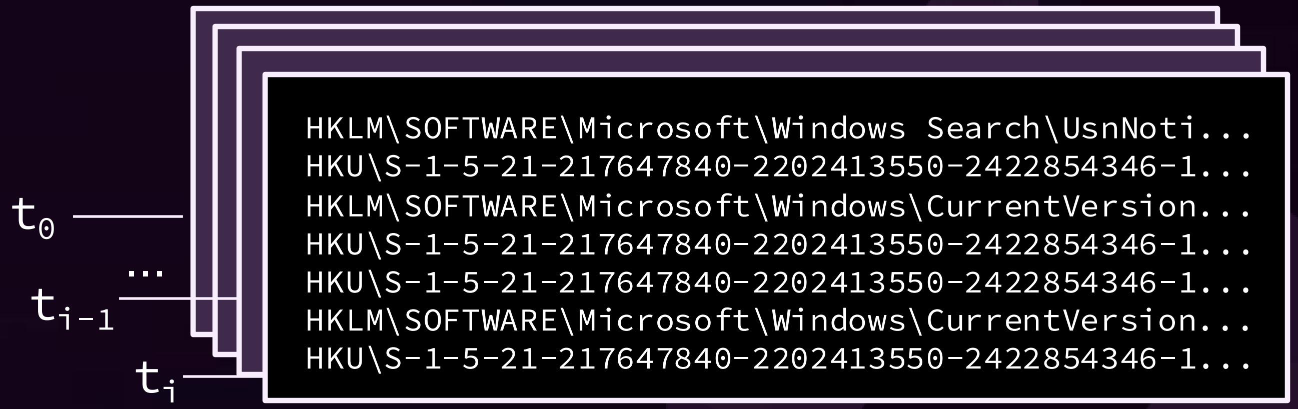
- Timing
- Service Names
- Service Creation Method

Noise

- MS Office Suite
- MS Paint
- Notepad
- Web Browsing
- Remote Desktop Activity

Registry Keys From Log Entries Within Time Window

`(Sysmon Events 12, 13, 14)`



Additional Dataset Statistics

- Event Generation Ran for 17 hours
- 364,675 registry events were logged
- 89,248 unique keys

Processing Registry Data

Key Normalization

Keys exhibit common patterns with small changes. There are common keys structures that are common except for service names, GUIDs, identifier strings, etc.

Replacements

- ServiceIDs
- Process GUIDs
- ProgIDs
- Common Service IDs
- ComponentFamily strings
- Misc. others

Initial Set of Keys Example

```
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_159ebf9
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_159ebf9\ImagePath
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_159ebf9\FailureActions
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_15d736f\Description
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_3b34c9e
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_3b34c9e\ImagePath
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_3b34c9e\FailureActions
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_3b34c9e\Description
```

SNAPATTACK

Processing Registry Data

Key Normalization

Keys exhibit common patterns with small changes. There are common keys structures that are common except for service names, GUIDs, identifier strings, etc.

Replacements

- ServiceIDs
- Process GUIDs
- ProgIDs
- Common Service IDs
- ComponentFamily strings
- Misc. others

After Substitution

```
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_<CSID>
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_<CSID>\ImagePath
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_<CSID>\FailureActions
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_<CSID>\Description
```

Also truncate long key paths to N=7 elements

Additional Dataset Statistics

- Event Generation Ran for 17 hours
- 364,675 registry events
- 89,248 unique keys
- 28,438 unique processed keys
- 65% Reduction in unique key count

Clustering Registry Data

Tokenize keys by path element after substitutions

Perform Agglomerative Clustering Using Jaccard Distance

key 1

```
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_159ebf9\Description
```

Token Set { HKLM, System, CurrentControlSet, Services, ConsentUxUserSvc_<CSID>, Description }

key 2

```
HKLM\System\CurrentControlSet\Services\ConsentUxUserSvc_3b34c9e\FailureActions
```

Token Set { HKLM, System, CurrentControlSet, Services, ConsentUxUserSvc_<CSID>, FailureActions }

$$D(K1, K2) = 1 – J(K_1 \text{ Token Set}, K_2 \text{ Token Set}) = 1- 5/7 = 0.286$$

Clustering Registry Data

Example Key Clusters

Cluster Statistics

- 7148 clusters (92% reduction)
- 148 clusters with N > 10
- 95% of clusters N < 6
- Largest cluster N = 2502

Example Cluster 1 (N=14)

```
hklm\components\deriveddata\components\amd64_microsoft-windows-appx-dep\f!appxapplicabilityblob.dll
hklm\components\deriveddata\components\amd64_microsoft-windows-appx-dep\f!appxupgrademigrationplugi_90cf
hklm\components\deriveddata\components\amd64_microsoft-windows-appx-dep
hklm\components\deriveddata\components\amd64_microsoft-windows-appx-dep\s256h
hklm\components\deriveddata\components\amd64_microsoft-windows-appx-dep\f!settings.dat
...
```

Example Cluster 2 (N=93)

```
hku\.default\software\microsoft\systemcertificates\root
hku\.default\software\microsoft\systemcertificates\trust\ctls
hku\.default\software\microsoft\systemcertificates\disallowed\certificates
hku\<sid>\software\microsoft\systemcertificates\trust
hku\<sid>\software\microsoft\systemcertificates\trustedpeople
...
```

Example Cluster 3 (N=4)

```
hku\<sid>\software\microsoft\input\typinginsights
hku\<sid>\software\microsoft\input\tipc
hku\<sid>\software\microsoft\input\ec
hku\<sid>\software\microsoft\input\typinginsights\insights
```

part 04.

Our Catch

04. our catch

Results + Lessons Learned

SNAPATTACK

Results

Pattern Mining Results

SUPPORT	ITEM SETS
0.6209	{Clust 0001}
0.3946	{Clust 0027}
0.3919	{Clust 0027, Clust 0001}
0.3603	{Clust 2599}

Negative Baseline

0.8969	{Clust 0027}
0.5954	{Clust 2599}
0.3511	{Clust 0064}

28,438 Unique Normalized Keys

7,148 Clusters

1,747 Transactions for FIM to analyze

Cluster 0001 N=6045

hklm\system\currentcontrolset\services\lqdvlqzy7szi4 5c3jb1_poy6xtnmhmd\start
hklm\system\currentcontrolset\services\ejx-+agc07wfl3ae8rwhm65fyn9ptduo
hklm\system\currentcontrolset\services\abveylieks6bml5a+obgdc21kvhqfjz7\start
...

Cluster 0027 N=98

hku\<sid>\software\microsoft\onedrive\accounts\lastupdate
hku\<sid>\software\microsoft\windows\shell\bags
hku\<sid>\software\miscrosoft\edge\extensions
...

Cluster 2599 N=1

hklm\system\currentcontrolset\services\bam\state\usersettings

Cluster 0064 N=6

hku\<sid>\software\microsoft\gamebarapi\input\inputredirected
hku\<sid>\software\microsoft\input\tpic
hku\<sid>\software\miscrosoft\gamebarapi\input
...

SNAPATTACK

Lessons Learned

01.
"Attack as code" was critical for this project.

02.
Registry data per processing technique seems very promising. Clustering merits further evaluation.

03.
Frequent Item Set Mining may be unnecessary.

The pilot study is a success!
Explore this methodology with other types of attacks.