Detecting XMRig CPU or GPU mining

XMRig is a Trojan Horse that hijacks a user’s computer and uses its resources to mine digital currency. It is high performance, open source, and cross platform. Attackers typically aim to hijack the resources of affected systems to validate transactions in cryptocurrency networks, earning the attackers virtual currency.

Transaction validation usually requires heavy system resource usage, and enough system resources can be consumed to negatively impact machines or cause them to become unresponsive. Servers and cloud-based systems are common targets because of the high potential for available resources, but user endpoint systems may also be compromised.

These searches allow you to detect and investigate unusual activities that might relate to XMRig, including looking for file writes associated with its payload, process command-line, defense evasion, and hacking tools including Telegram to download other files.

Required data

- Endpoint data
- Microsoft: Sysmon

Some commands, parameters, and field names in the searches below may need to be adjusted to match your environment. In addition, to optimize the searches shown below, you should specify an index and a time range when appropriate.

How to use Splunk software for this use case

► Deleting of net users

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search detects a suspicious net.exe or net1.exe command line used to delete a user on a system. This behavior is used by attackers to impair users or delete evidence created during the attack's lateral movement to additional systems.

If this search returns potentially suspicious results, review parallel processes for additional behavior and examine other user accounts created before or after the event.

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
False positives from this search might occur since system administrators or scripts may delete user accounts using this technique. Tune and filter as necessary.


Disable Windows app hotkeys

To complete this process, your deployment needs to ingest data that records registry activity from your hosts. This is typically populated via endpoint detection-and-response product, such as Carbon Black or endpoint data sources, such as Sysmon. The data used for this search is typically generated via logs that report reads and writes to the registry. You should also ensure you are ingesting normalized endpoint data, populating the Registry node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies suspicious registry modifications to disable hotkeys for native Windows applications. This technique is used by attackers to disable Windows applications like taskmgr.exe and cmd.exe. This technique is used to impair investigators in their analysis and removal of attacker implants in compromised systems.

False positives from this search might be high, based on legitimate scripted code in any environment. Tune and filter as necessary.

| tstats summariesonly=false allow_old_summaries=true count FROM datamodel=Endpoint.Registry WHERE ("Registry.registry_path"="*\Windows NT\CurrentVersion\Image File Execution Options\*" "Registry.registry_value_data"="HotKey Disabled" "Registry.registry_value_name"="Debugger") BY _time span=1h, "Registry.dest", "Registry.user", "Registry.registry_path", "Registry.registry_value_name", "Registry.registry_key_name", "Registry.process_guid", "Registry.registry_value_data" | rename "Registry.*" AS "*" | rename process_guid AS proc_guid | join _time,proc_guid [ | tstats summariesonly=false allow_old_summaries=true count FROM datamodel=Endpoint.Processes BY _time span=1h, "Processes.process_id", "Processes.process", "Processes.dest", "Processes.parent_process_name", "Processes.parent_process", "Processes.process_guid" | rename "Processes.*" AS "*" | rename process_guid AS proc_guid | fields + _time, dest, user, parent_process_name, parent_process, process_name, process_path, proc_guid, registry_path, registry_value_name, registry_value_data, registry_key_name] | table _time, dest, user, parent_process_name, parent_process, process_name, process_path, process, proc_guid, registry_path, registry_value_name, registry_value_data, registry_key_name | search
► Disable net user account

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies a suspicious command line that disables a user account using the net.exe utility in Windows. This technique is used by attackers to interrupt user account availability.

```bash
```

► Download file using Telegram

To complete this process, your deployment needs to ingest logs with the process name and TargetFilename from your endpoints, or information about events that occur when a process performs a download, like event code 15. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search identifies a suspicious download by the Telegram application on a Windows system. This technique is used by attackers who gain access, install Telegram, and then download different network scanners such as port, bruteforcer, or masscan to the system, which are used to map the network and move laterally.

```bash
| search (EventCode=15 TargetFilename="*:Zone.Identifier" process_name="telegram.exe" (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational)) | stats count min(_time) AS firstTime max(_time) AS lastTime BY Computer EventCode Image process_id TargetFilename Hash | convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime) | convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

► Enumerate user local group using Telegram

To complete this process, your deployment needs to ingest logs with the task schedule from your endpoints, for example security log event code 4798.

This search detects a suspicious Telegram process that enumerates all network users in a local group. This
technique is used by attackers who map all users on a compromised system. The search looks for event code 4798, which is generated when a process enumerates a user’s security-enabled local groups on a computer or device.

Tune and filter known instances of process like logonUI used in your environment.

```snplunk
| search (EventCode=4798 Process_Name="\telegarme.exe" eventtype=wineventlog_security)
| stats count min(_time) AS firstTime max(_time) AS lastTime BY ComputerName EventCode Process_Name Process_ID Account_Name Account_Domain Logon_ID Security_ID Message
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

► Excessive attempts to disable services

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search detects a suspicious series of command lines used to disable several services. This technique is used by attackers to disable security services or other malware services on a compromised system.

```snplunk
| tstats summariesonly=false allow_old_summaries=true values("Processes.process") AS process,
values("Processes.process_id") AS process_id, count, min(_time) AS firstTime, max(_time) AS lastTime
| where (count >= 5)
| rename "Processes.*" AS *"`
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)
```

► Excessive service stop attempts

To complete this process, your deployment needs to ingest information on process that include the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies a suspicious series of attempts to stop multiple services on a system using either net.exe or sc.exe. This technique is used by attackers to terminate security services or other related services on a compromised system.

```snplunk
| tstats summariesonly=false allow_old_summaries=true values("Processes.process") AS process,
values("Processes.process_id") AS process_id, count, min(_time) AS firstTime, max(_time) AS lastTime
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
FROM datamodel=Endpoint.Processes WHERE 
(("Processes.process_name"="net.exe" OR "Processes.original_file_name"="net.exe" OR "Processes.process_name"="net1.exe" OR "Processes.original_file_name"="net1.exe" OR "Processes.process_name"="sc.exe" OR "Processes.process_name"="net1.exe") 
("Processes.process"="*stop*" OR "Processes.process"="*delete*")) 
BY "Processes.process_name", "Processes.original_file_name", "Processes.parent_process_name", "Processes.dest", "Processes.user", _time span=1min
| where (count >= 5)
| rename "Processes."" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

► Excessive use of Cacls application

To complete this process, your deployment needs to ingest information on process that include the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies excessive usage of the cacls.exe, xcacls.exe, or icacls.exe applications to change file or folder permissions. This technique is used by attackers to impair users from deleting or accessing malware components or artifacts in a compromised system.

| tstats summariesonly=false allow_old_summaries=true values("Processes.process") AS process, values("Processes.process_id") AS process_id, values("Processes.process_name") AS process_name, count, min(_time) AS firstTime, max(_time) AS lastTime FROM datamodel=Endpoint.Processes WHERE 
("Processes.process_name"="cacls.exe" OR "Processes.process_name"="icacls.exe" OR "Processes.process_name"="XCACLS.exe") 
BY "Processes.parent_process_name", "Processes.parent_process", "Processes.dest", "Processes.user", _time span=1min
| where (count >= 10)
| rename "Processes."" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

► Excessive use of Net application

To complete this process, your deployment needs to ingest information on process that include the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies excessive usage of net.exe or net1.exe within a 1 minute bucket of time, although you can modify the time span as needed. This technique is used by attackers to create many users, or delete and disable users.

| tstats summariesonly=false allow_old_summaries=true values("Processes.process") AS process, values("Processes.process_id") AS process_id, count, min(_time) AS firstTime, max(_time) AS lastTime
From datamodel=Endpoint.Processes WHERE ("Processes.process_name"="net.exe" OR "Processes.original_file_name"="net.exe" OR "Processes.process_name"="net1.exe" OR "Processes.original_file_name"="net1.exe") BY "Processes.process_name", "Processes.parent_process_name", "Processes.original_file_name", "Processes.dest", "Processes.user", _time span=1min
| where (count >= 10)
| rename "Processes.*" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

► Excessive use of Taskkill

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies excessive usage of the taskkill.exe application. This technique is used by attackers to evade detections by stopping a security product or other processes to evade detection.

| tstats summariesonly=false allow_old_summaries=true values("Processes.process") AS process,
values("Processes.process_id") AS process_id, count, min(_time) AS firstTime, max(_time) AS lastTime
| where (count >= 10)
| rename "Processes.*" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

► Executables or script creation in suspicious file path

To complete this process, your deployment needs to ingest information on process that include the name of the Filesystem responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Filesystem node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies suspicious executables or scripts (known file extensions) in a list of suspicious file paths in Windows. This technique is used by attackers to evade detection. The suspicious file paths used in this search are known paths used in the wild and don't commonly have executables or scripts.

| tstats summariesonly=false allow_old_summaries=true values("Filesystem.file_path") AS file_path, count,
min(_time) AS firstTime, max(_time) AS lastTime FROM datamodel=Endpoint.Filesystem WHERE
("Filesystem.file_name"=*.exe OR "Filesystem.file_name"=*.dll OR "Filesystem.file_name"=*.sys OR
"Filesystem.file_name"=*.com OR "Filesystem.file_name"=*.vbs OR "Filesystem.file_name"=*.vbe OR
"Filesystem.file_name"=*.js OR "Filesystem.file_name"=*.ps1 OR "Filesystem.file_name"=*.bat OR
"Filesystem.file_name"=*.cmd OR "Filesystem.file_name"=*.pif) ("Filesystem.file_path"="\windows\fonts\*")
OR "Filesystem.file_path"="\windows\temp\*" OR "Filesystem.file_path"="\users\public\*" OR "Filesystem.
file_path"="\windows\debug\*" OR "Filesystem.file_path"="\Users\Administrator\Music\*" OR "Filesystem.
file_path"="\Windows\servicing\*" OR "Filesystem.file_path"="\Users\Default\*" OR "Filesystem.file_
path"="\Recycle.bin\*" OR "Filesystem.file_path"="\Windows\Media\*" OR "Filesystem.file_
path"="\Windows\repair\*" OR "Filesystem.file_path"="\AppData\Local\Temp\*" OR "Filesystem.file_
path"="\PerfLogs\*\") BY "Filesystem.file_create_time", "Filesystem.process_id", "Filesystem.file_name",
"Filesystem.user"

| rename "Processes.*" AS "*"
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

► Hide user account from sign-in screen

To complete this process, your deployment needs to ingest data that records registry activity from your hosts. This is
typically populated via endpoint detection-and-response product, such as Carbon Black or endpoint data sources,
such as Sysmon. The data used for this search is typically generated via logs that report reads and writes to the
registry. You should also ensure you are ingesting normalized endpoint data, populating the Registry node of the
Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM,
see the Common Information Model documentation.

This search looks for a suspicious registry modification to hide a user account on the Windows login screen. When
attackers create user accounts with administrator privileges, they use this technique to hide the account from
the login screen to avoid this indicator of compromise being noticed by users.

| tstats summariesonly=false allow_old_summaries=true count, min(_time) AS firstTime, max(_time) AS
lastTime FROM datamodel=Endpoint.Registry WHERE ("Registry.registry_path"="\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist***" "Registry.registry_value_data"="0x00000000")
BY _time span=1h, "Registry.dest", "Registry.user", "Registry.registry_path", "Registry.registry_value_
name", "Registry.process_guid", "Registry.registry_key_name", "Registry.registry_value_data"
| rename "Registry.*" as "***"
| rename process_guid as proc_guid
| join _time,proc_guid

[ | tstats summariesonly=false allow_old_summaries=true count FROM datamodel=Endpoint.Processes BY
dest", "Processes.parent_process_name", "Processes.parent_process", "Processes.process_guid"
| rename "Processes.*" AS "***"
| rename process_guid as proc_guid
| fields + _time, dest, user, parent_process_name, parent_process, process_name, process_path, process,
proc_guid, registry_path, registry_value_name, registry_value_data, registry_key_name]
| table _time, dest, user, parent_process_name, parent_process, process_name, process_path, process,
proc_guid, registry_path, registry_value_name, registry_value_data, registry_key_name
| search

► Icacls grant or deny command

To complete this process, your deployment needs to ingest logs with the process name, parent process, and
command line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the
Sysmon TA. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the
Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM,
This search looks for the modification of the security permissions of a specific file or directory in the `icacls.exe`, `cacls.exe` or `xcacls.exe` applications. This technique is commonly seen in miner scripts to evade detection and restrict access to their component files.

| rename "Processes.*" AS *
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime)
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime)

Make the following adjustments to the search above to search for grant or deny commands.

<table>
<thead>
<tr>
<th>Command to search for</th>
<th>Replacement search lines</th>
</tr>
</thead>
</table>
| Grant                 | "Processes.process"="*/grant*"
| Deny                  | "Processes.process"="*/deny*"

Modify ACL permission to files or folders

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command line executions from your endpoints. If you are using `Sysmon`, you must have at least version 6.0.4 of the Sysmon TA. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the `Endpoint data model` in the Common Information Model (CIM). For information on installing and using the CIM, see the [Common Information Model documentation](https://www.splunk.com). This search looks for suspicious modifications of ACL permissions to files or folders to make them available to everyone. This technique is used by attackers to evade ACL or protected files access.

This change is usually configured by a file or directory owner with appropriate permissions. This behavior is potentially suspicious if this command is seen on a machine used by an account without permissions to do so.

False positives from this search may occur since some administrators might use this command. Tune and filter where necessary.

| rename "Processes.*" AS **

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
► Wmic.exe using delete to remove executable path

To complete this process, your deployment needs to ingest information on process that includes the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Process node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies wmic.exe using delete to remove a executable path. Attackers typically perform this action via a batch file during the beginning stages of setting up an endpoint for mining.

| rename "Processes.*" AS "*" |
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime) |
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime) |

► Schtasks task run on-demand

To complete this process, your deployment needs to ingest logs with the process name, parent process, and command-line executions from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA. You should also ensure you are ingesting normalized endpoint data, populating the Process node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search identifies a Windows scheduled task run on-demand through the shell or command line. This technique is used by attackers to run a scheduled task they have created as a persistence mechanism, or for lateral movement as part of an attack on a compromised machine.

False positives from this search may occur since administrators might use this technique to debug scheduled task entries. Tune and filter where necessary.

| rename "Processes.*" AS "*" |
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime) |
► Suspicious driver loaded path

To complete this process, your deployment needs to ingest logs with the driver loaded and signature from your endpoints. If you are using Sysmon, you must have at least version 6.0.4 of the Sysmon TA.

This search detects suspicious driver loaded paths. This technique is used by coin miners like XMRig to register malicious drivers from notable directories where executable or drivers do not commonly exist.

If you return potentially suspicious results from this search, validate whether the driver seen is for legitimate business use by reviewing the metadata and certificate information. Unsigned drivers from non-standard paths are not normal but can occur in limited legitimate scenarios. You should also review driver loads into ntoskrnl.exe for possible other drivers of interest. Perform long tail analysis of drivers by path (outside of default, and in default) for further review.

| search (EventCode=6 ImageLoaded="*.sys" NOT ImageLoaded="\\Windows\System32\DriverStore\FileRepository\*" NOT ImageLoaded="\\Windows\System32\drivers\*" NOT ImageLoaded="\\Windows\inf" (source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR source=type=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational)) |
| stats min(_time) AS firstTime max(_time) AS lastTime count BY Computer ImageLoaded Hashes IMPHASH Signature Signed |
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(firstTime) |
| convert timeformat="%Y-%m-%dT%H:%M:%S" ctime(lastTime) |

► Suspicious process file path

To complete this process, your deployment needs to ingest information on process that include the name of the process responsible for the changes from your endpoints. You should also ensure you are ingesting normalized endpoint data, populating the Processes node of the Endpoint data model in the Common Information Model (CIM). For information on installing and using the CIM, see the Common Information Model documentation.

This search detects a suspicious process running in a file path where a process is not commonly seen. This behavior is used by attackers where they drop and run an .exe file in a path that is accessible without admin privileges.

| tstats summariesonly=false allow_old_summaries=true count, values("Processes.process_name") AS process_name, values("Processes.process") AS process, min(_time) AS firstTime, max(_time) AS lastTime FROM datamodel=Endpoint.Processes WHERE ("Processes.process_path"="*\windows\fonts\*" OR "Processes.process_path"="*\windows\temp\*" OR "Processes.process_path"="*\users\public\*" OR "Processes.process_path"="*\windows\debug\*" OR "Processes.process_path.file_path"="*\Users\Administrator\Music\*" OR "Processes.process_path.file_path"="*\Windows\servicing\*" OR "Processes.process_path_file_path"="*\Users\Default\*" OR "Processes.process_path.file_path"="*\Recycle.bin\*" OR "Processes.process_path"="*\Windows\Media\*" OR "Processes.process_path"="*\Repair\*" OR "Processes.process_path"="*\PerfLogs\*" ) BY "Processes.parent_process_name", "Processes.parent_process", "Processes.process_path", "Processes.dest", "Processes.user" |
| rename "Processes.*" AS "*" |
XMRig driver loaded

To complete this process, your deployment needs to ingest logs with the driver loaded and signature from your endpoints. If you are using **Sysmon**, you must have at least version 6.0.4 of the Sysmon TA.

This search identifies XMRig miner drivers installed on a system. The XMRig driver name by default is WinRing0x64.sys.

```
| search (EventCode=6 (ImageLoaded="*\WinRing0x64.sys" OR Signature="Noriyuki MIYAZAKI")
(source=Syslog:Linux-Sysmon/Operational OR source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational OR sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational))
| stats min(_time) AS firstTime max(_time) AS lastTime count BY Computer ImageLoaded Hashes IMPHASH Signature Signed
```

Next steps

The content in this article comes from **Splunk Enterprise Security (ES)**. As a Splunk premium security solution, ES solves a wide range of security analytics and operations use cases including continuous security monitoring, advanced threat detection, compliance, incident investigation, forensics and incident response. Splunk ES delivers an end-to-end view of an organization’s security posture with flexible investigations, unmatched performance, and the most flexible deployment options offered in the cloud, on-premises, or hybrid deployment models. If you have questions about this use case, see the Security Research team’s [support options on GitHub](#).

In addition, these Splunk resources might help you understand and implement this use case:

- **E-book**: [Ransomware, malware and cyberthreats](#)
- **Blog**: [Hunting with Splunk: The basics](#)