

Creating, Detecting, Defending Ransomware

Jackson McCullough

October 19, 2025

1 Creation, Setup, & Scenario

1.1 Scenario

Here, we are treating this scenario as it is a workplace environment, and this employee endpoint has sensitive data made vulnerable due to ineffective security controls. If a ransomware threat actor is able to execute a malicious code on the endpoint due to poor security/hardening, the initial set of files targeted may be encrypted before detection or preventive measures can respond. Mitigation and recovery measures can be taken to limit the spread to other endpoints, but often the first wave of encrypted files is unavoidable with poor security.





With this layout, a threat actor will launch a script to encrypt files, it will be logged, active response will take place, and access controls will be implemented to kill/delete the script from the endpoint, and prevent any other scripts from being ran.

1.2 Setup

To start, I made a very simple script that ran on open, that would take the files within a specific directory, and encrypt them. The point of this was to try

and simulate a real scenario where an employee, for example, would accidentally open what they believed was an unsuspecting file that would then encrypt "important documents"

In the Windows VM, dummy documents were made to simulate sensitive, yet vulnerable, data on some end device. This was the created list that was created, with useless and randomly generated data in them, but acted as passwords, usernames, etc.

 passwords.txt	10/17/2025 2:16 PM	Text Document	1 KB
 ssn.txt	10/17/2025 2:16 PM	Text Document	1 KB
 user_accounts.txt	10/17/2025 2:16 PM	Text Document	1 KB
 usernames.txt	10/17/2025 2:16 PM	Text Document	1 KB

The kinds of information in these files were placeholders like:

```
user_accounts.txt - Notepad
File Edit Format View Help
Full Name,Username,Email,SSN_token
Ava Patterson,ash_ketch,ava.patterson@example.test,TEST-SSN-0001
Brandon Lee,blueridge99,brandon.lee@example.test,TEST-SSN-0002
Celia Morgan,byte_monger,celia.morgan@example.test,TEST-SSN-0003
Derek Nash,cobalt_raven,derek.nash@example.test,TEST-SSN-0004
Eli Torres,delta_echo,eli.torres@example.test,TEST-SSN-0005
Fiona Park,echelon42,fiona.park@example.test,TEST-SSN-0006
Gavin Rose,falcon_six,gavin.rose@example.test,TEST-SSN-0007
Hanna Cole,ghostpacket,hanna.cole@example.test,TEST-SSN-0008
Ian Ford,hexsmith,ian.ford@example.test,TEST-SSN-0009
Jenna Cruz,ironclave,jenna.cruz@example.test,TEST-SSN-0010
Kurt Vale,jade_network,kurt.vale@example.test,TEST-SSN-0011
Lena Moss,kilo_kernel,lena.moss@example.test,TEST-SSN-0012
```

To make sure Wazuh had as wide of a range as possible for logging, **Sysmon** & **Syscheck** rules were added into the logging, so Wazuh can see as much as possible. Configurations were added to the XML policy files such as:

```

<syscheck>
  <disabled>no</disabled>

  <!-- changed syscheck frequency to 3600 -->
  <frequency>3600</frequency>
  <directories realtime="yes" check_all="yes" report_changes="yes">C:\Users\*\Documents</directories>
  <directories realtime="yes" check_all="yes" report_changes="yes">C:\Users\lxvert\Desktop\dummy</directories>

</localfile>
  <location>Microsoft-Windows-Sysmon/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>

<group name="local,sysmon,python_execution,">
  <!-- detecting the use of any scripts executed. Since python is on here
  , we need to monitor it-->
  <rule id="100501" level="12">
    <if_group>sysmon_event1</if_group>
    <regex type="pcre2">(?!C:\\Users\\lxvert\\Desktop\\dummy.*\\.py)s</regex>
  </rule>
</group>

```

1.3 Script

Next, a Python script was created that simply went through the files of the directory I directed it to, and encrypted all of the files within that directory. For this experiment, a folder called "important information" is being targeted. The script is using **AES-128**, symmetric encryption with the key generated once the script is ran, using the cryptography library **Fernet**.

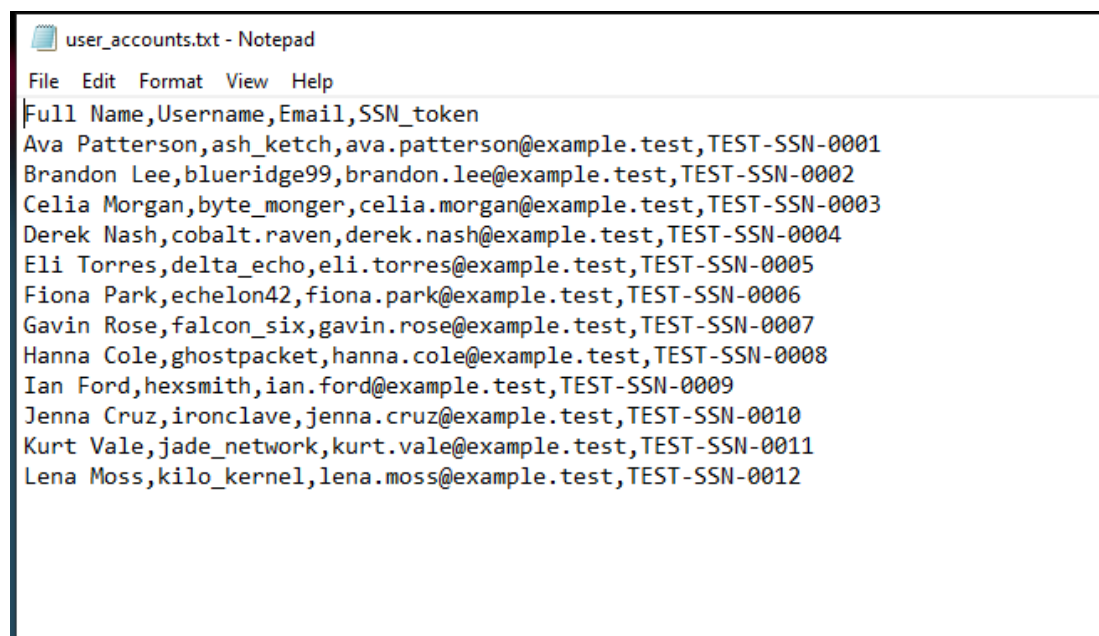
1.4 Monitoring and Access Control

For this project, **Wazuh** is the XDR we are using, so information can be logged while also using some EDR capabilities in this scenario. The monitoring will be from the Windows endpoint with an **Ubuntu machine** as the Wazuh manager. **Sysmon** is also being utilized to provide more information than what Windows Event Viewer would typically hold. This will be able to monitor the

installation or download of this script, and have **Windows Defender Application Control (WDAC)** to block this script from being ran after initial breach, and will use **Active Response** from Wazuh to kill/delete any python scripts that are on the endpoints, unless they are signed scripts.

2 Detection

The employee was sent this ransomware in someway, and opened the program. As this takes place, the files in the folder of sensitive information turned from this:







```
user_accounts.txt - Notepad
File Edit Format View Help
Full Name,Username,Email,SSN_token
Ava Patterson,ash_ketch,ava.patterson@example.test,TEST-SSN-0001
Brandon Lee,blueridge99,brandon.lee@example.test,TEST-SSN-0002
Celia Morgan,byte_monger,celia.morgan@example.test,TEST-SSN-0003
Derek Nash,cobalt.raven,derek.nash@example.test,TEST-SSN-0004
Eli Torres,delta_echo,eli.torres@example.test,TEST-SSN-0005
Fiona Park,echelon42,fiona.park@example.test,TEST-SSN-0006
Gavin Rose,falcon_six,gavin.rose@example.test,TEST-SSN-0007
Hanna Cole,ghostpacket,hanna.cole@example.test,TEST-SSN-0008
Ian Ford,hexsmith,ian.ford@example.test,TEST-SSN-0009
Jenna Cruz,ironclave,jenna.cruz@example.test,TEST-SSN-0010
Kurt Vale,jade_network,kurt.vale@example.test,TEST-SSN-0011
Lena Moss,kilo_kernel,lena.moss@example.test,TEST-SSN-0012
```

To a completely encrypted file such as this:

gAAAAABo9OM_91B52HTc74K3zyPw9Rgx5uEn19vS3WTLou-mE9C_5tdDZALyPatzbR8j3JJv9X0vfLW2zgmFb7u9bgJr1UHNCRb_MEwZryQYTGHuKnFjHkKAAIG9LNzyQ2qUGTX_xnekVNSecBJAYBsXTywj4m0XV5PxYjRKRNSL4S1XXK4We_kBqvb06f8uRYzKwm0ieE30BqyWTymVQ==

This would show suspicious on the manager side. There would be fast, repeated alerts of changes in these files, specifically saying "Integrity checksum changed", meaning that the data, or integrity, has been changed from what it previously was. In Wazuh, this alert is suspicious because it would be mass volume within a very small time period of integrity changes. Since this folder only has a few files, it is not as large at volume, but shows as this in Wazuh:

	timestamp	agent.name	syscheck.path	syscheck.event	rule.des...
	Oct 19, 2025 @ 13:10:21.736	flare_vm	c:\users\lxvert\desktop\dummy\user_accounts.txt	modified	Integrity ch...
	Oct 19, 2025 @ 13:10:21.647	flare_vm	c:\users\lxvert\desktop\dummy\usernames.txt	modified	Integrity ch...
	Oct 19, 2025 @ 13:10:21.563	flare_vm	c:\users\lxvert\desktop\dummy\ssn.txt	modified	Integrity ch...
	Oct 19, 2025 @ 13:10:21.479	flare_vm	c:\users\lxvert\desktop\dummy\passwords.txt	modified	Integrity ch...

As you can see, all of these files were modified all within the same second, which would be near impossible for a regular employee to modify so many files in under one second.

Wazuh also picked up many packages that were on this endpoint that would not be used for general python programming, which sparks the suspicion on top of what has already been observed of this endpoint.

218 hits					
△	Export Formatted	🔄	Reset view	📄	22 available fields
≡	Columns	📄	Density	⬆	1 fields sorted
🖥	Full screen				
	agent.name	package.vendor	↑ package.name	package.version	package.type
🔍	flare_vm	-	coloredlogs	15.0.1	pypi
🔍	flare_vm	-	cryptography	45.0.6	pypi
🔍	flare_vm	-	cxxfilt	0.3.0	pypi
🔍	flare_vm	-	decorator	5.2.1	pypi
🔍	flare_vm	-	defusedxml	0.7.1	pypi
🔍	flare_vm	-	dissect	3.19	pypi
🔍	flare_vm	-	dissect.archive	1.6	pypi
🔍	flare_vm	-	dissect.btrfs	1.7	pypi
🔍	flare_vm	-	dissect.cim	3.12	pypi
🔍	flare_vm	-	dissect.clfs	1.10	pypi
🔍	flare_vm	-	dissect.cstruct	4.5	pypi
🔍	flare_vm	-	dissect.esedb	3.16	pypi
🔍	flare_vm	-	dissect.etl	3.11	pypi
🔍	flare_vm	-	dissect.eventlog	3.10	pypi
🔍	flare_vm	-	dissect.evidence	3.11	pypi

3 Eradication & Recovery

As this was investigated, we could see with Sysmon what script was executed. In order to block this script from being ran again, we will use **Windows Defender Application Control**. This will allow us to make a rule that will prevent this script from being ran anywhere from the hash signature.

Since we know what script was ran, going into powershell and looking for this hash of the exact script would not be hard to do. This is the best way to prevent this exact script from running, since it would block the file itself, no matter what directory it was in on the endpoint.

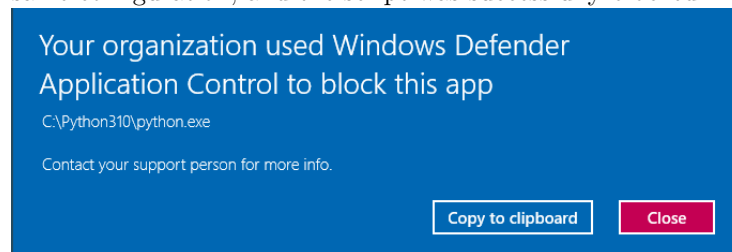
```
Algorithm      Hash
-----
SHA256         220AE54ACE62FD7E23A4908345415FF6D479A65E72A6B13D5C5D5DF42700AECB
```

Once the hash has been gathered, the WDAC rule needs to be created for

the endpoint to be restricted on this file execution. The rule created was as follows:

```
1 $policyXML = "C:\Windows\schemas\CodeIntegrity\ExamplePolicies\DefaultWindows_Enforced.xml"
2
3 $blockedHash = "220AE54ACE62FD7E23A4908345415FF6D479A65E72A6813D5C5D5DF42700AECB"
4
5 Add-CIPolicyRule -FilePath $policyXML -Deny -Hash $blockedHash
6
7 ConvertFrom-CIPolicy -XmlFilePath $policyXML -BinaryFilePath "C:\Windows\System32\CodeIntegrity\SIPolicy.p7b"
```

After restart, the configuration has to be tested to know it works. Once the script was found in file, an attempt of access was made in a sandbox with the same configuration, and the script was successfully blocked.



4 Next Steps & Large-Scale Work

After successful eradication and recovery. The next steps would be to apply this to all endpoints that are in the company, and focus system hardening. For this project, there was only one endpoint. Therefore, large scale application control and system hardening would not take place at great volume. If there were more than just one, Group Policy, Microsoft Intune, or Configuration Manager (SCCM) to enforce controls enterprise-wide would be utilized. Finally, integrate automated patching, vulnerability scanning, network access control, code signing, and user awareness training to reduce the attack surface and improve system hardening and resilience across all endpoints.