# Week 9 Ass 2 Attack

## Task Overview

This task is expected to take approximately 10 hours to complete to a high standard: 5 hours to develop a collection of attacks to emulate a known Threat Actor (TA) and 5 hours to implement detections (or preventions) and test those controls on an implementation system.

## Groupwork

You may work alone or in groups of 2 or 3 people for this task and for your final Assignment 3 presentation.

All group members must be identified in the groupwork artefacts. Evidence must be provided that all group members contributed adequately to the final submissions. All group members must submit via the unit website. The moderation process might allocate group members different marks. Sharing of artefacts, for example, code or virtual machines, between groups is not permitted.

## Code Reuse

You are encouraged to search for and reuse good quality code in this unit. All sources of code should be referenced. For websites and web services, the inclusion of a commented URL in the reused code is a sufficient reference. Deprecated code, for example, PowerShell scripts that use ping, should not be reused.

## Repository

Create a private Git repository for yourself or for your group, for example, on GitHub. One code repository is to be used by all group members. Invite your tutor, the unit coordinator and your group members to the private repository.

## Due

The Attack task is due in Week 9.

## Return

Feedback for will be provided within 2 weeks of the due date.

## Submission Overview

Submit artefacts to both your private code (Git) repository and to the unit website. Submit a link to your private repository to the unit website. All group members must submit.

## Criteria Overview

You will be marked on aspects such as your scripts' functionality, modularity, style, documentation, resemblance of actual attack code, lack of deprecated features, level of automation, ATT&CK categorisation of attack code, use of code repository tools, the quality of your detection rules and testing.

## Scenario

You work at the Monto Caravan and Cabin Park. Unfortunately, the company which you outsource your bookings to has been attacked by BlackCat ransomware. Your boss has asked you to improve the cybersecurity of our systems to prevent or mitigate further attacks.

For this task, the information system is comprised of the following nodes:

- DC VM
- Wazuh VM

You will:

- Emulate an attack
- Implement and test detections (or preventions)

**Emulate an Attack**

Do not download untrusted resources even in a virtual machine.

Develop four attacks. Investigate BlackCat to identify actual commands run by BlackCat. Reference your sources. Categorise the ATT&CK for each command. Appendix A shows an example for BumbleBee.

Your attacks should benignly emulate BlackCat using PowerShell on DC as close as possible to real attack commands. You are encouraged to use resources such as the Atomic Red Team library and ChatGPT to help you develop your attacks. Include references.

Create PowerPoint slides detailing your attacks including screenshots.

Document and clean up your code using PSScriptAnalzyer. Submit your code to your Git repository.

**Implement and Test Detections and Preventions**

Develop at least two detections for your attacks. Add rules to Wazuh that generate alerts when your attacks are executed. To save yourself time, you should first determine if each attack can be detected using Sysmon events. The ATT&CK website also includes detection advice which you might find useful.

Ensure any rules you add are not too specific and not too noisy: your detections should include at least one negate that stops a benign log event from escalating to an alert.

Submit your Wazuh rules file to your Git repository.

Add slides to your presentation for the detections that you have implemented.

Alternatively, instead of implementing detections, you can implement preventions to stop the attacks. You do not need to develop code for any preventions that you implement. Include slides and screenshots showing how you implemented the prevention and the difference in behaviour of the attack.

## Task Submission

Include a link to your private repository in your PowerPoint slideshow. Commit your PowerPoint slideshow to your private Git repository. Submit your PowerPoint slideshow to the unit website. All group members must submit.

## Task Criteria

Each of the following marking criteria have equal weighting.

| Criteria | Indicative of 100% | 75% | 50% | 25% | 0% |
|---|---|---|---|---|---|
| Attacks | Resembles actual attack code ←<br>Correct categorisation of ATT&CK codes for attacks ←<br>Attacks are scripted ←<br>Code consistent, reasonable layout ←<br>Functions documented appropriately ←<br>Git commits showing script & function development ← | | → Only 2 attacks implemented | →Lack of modularity, e.g. poor or no functions<br>→Uses deprecated functionality | → Attacks do not work in VMs<br>→ Lack of Git evidence of script(s) development<br>→ Poor referencing |
| Detections or preventions | Excellent detections or preventions, e.g. not too specific yet not noisy ← | | | → Noisy alerts<br>→ Overly specific rules | → No preventions or detections |
| Testing | Excellent testing of detections showing alerts & preventions showing difference in attack behaviour ← | | | | |

## Appendix A Emulated BumbleBee Attack

BumbleBee runs the following command (Proofpoint.com 2022):

```
cmd.exe /c start /wait "" "C:\Users\[removed]\AppData\Local\Temp\ATTACHME.LNK"
```

This command can be categorised as T1059.003 *Command and Scripting Interpreter: Windows Command Shell*. The command could be benignly emulated using the following PowerShell commands which create a symbolic link (like a file pointer) to notepad.exe and then runs notepad:

```
DC PS> New-Item -Path "$env:Temp\ATTACHME.LNK" `
           -Value "$env:Windir\notepad.exe" -ItemType SymbolicLink

DC PS> cmd.exe '/c start /wait "" "C:\Users\vagrant\AppData\Local\Temp\1\ATTACHME.LNK"'
```

References: (reddit.com 2022)