Domain computers have accounts, too!

Owning machines through relaying and delegation

Thanks

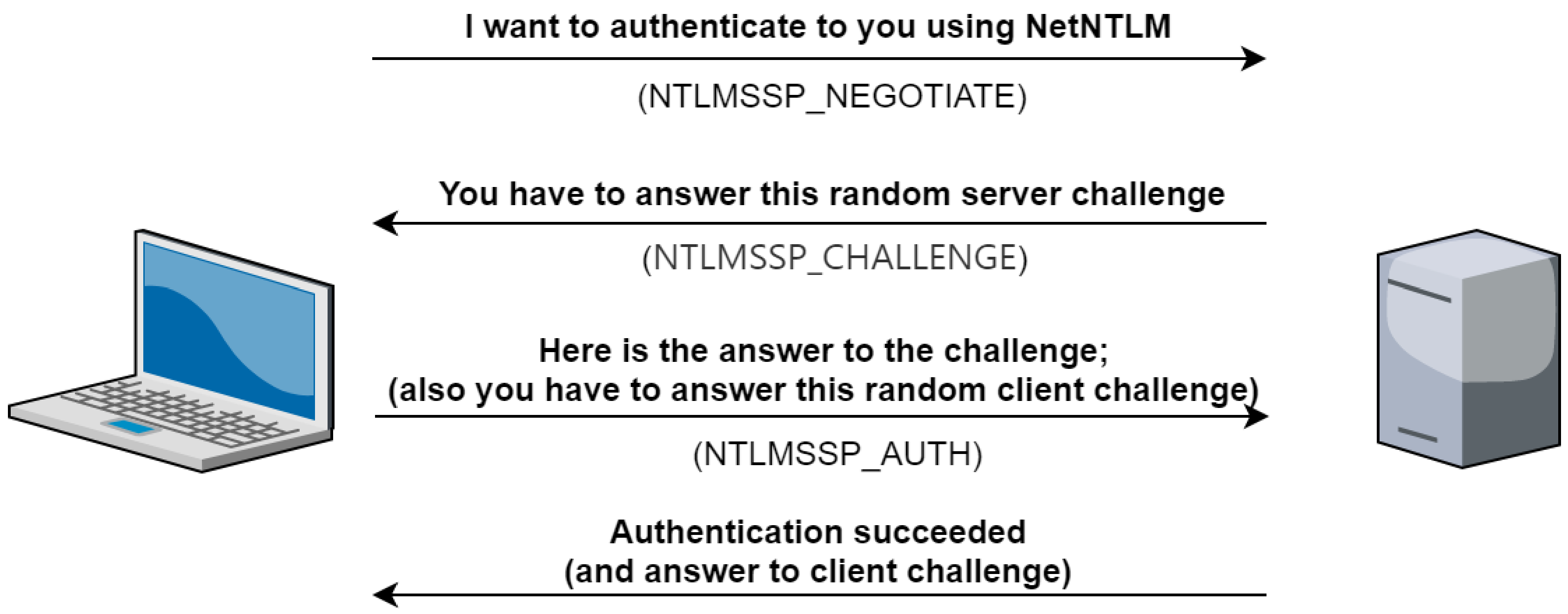
- @tifkin_ (Lee Christensen)
- @harmj0y (Will Schroeder)
- @enigma0x3 (Matt Nelson)
- @elad_shamir (Elad Shamir)
- @_dirkjan (Dirk-jan)
- everybody else who paved the way for this research
- my employer, DCIT

NTLM

- **NT (New Technology) LAN Manager** (**NTLM**)
- suite of protocols for security
- introduced in Windows NT4.0 (1996)
- best known for “NTLM hash” and related pass-the-hash attacks
- we will focus on the **challenge-response** protocols

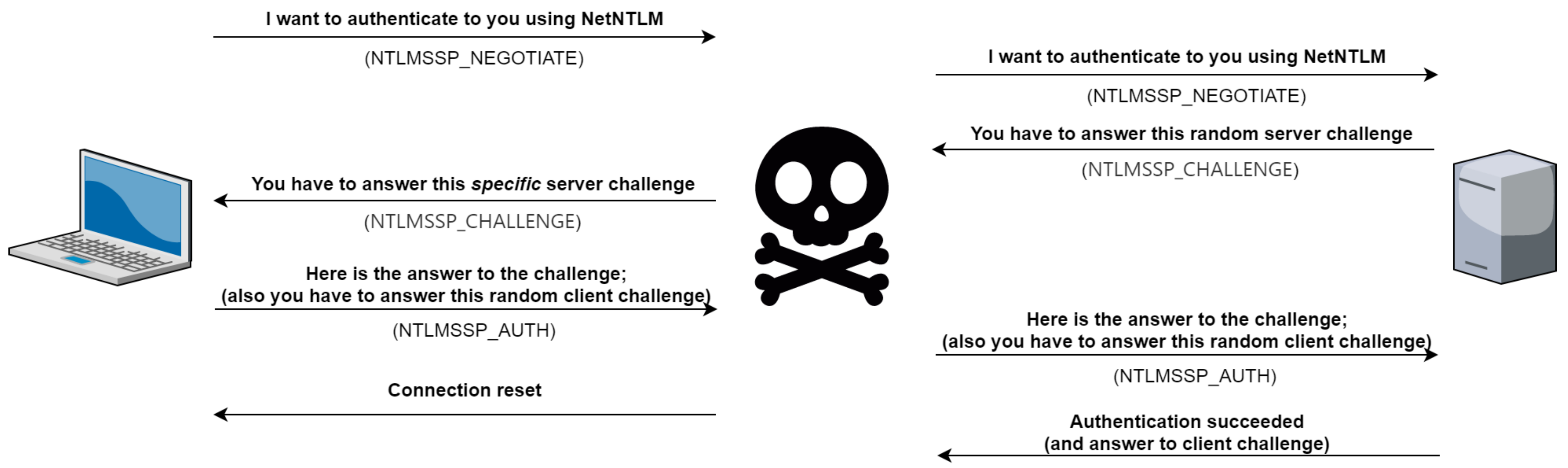
NetNTLM (v1 & v2)

Shared secret: NTLM hash of user‘s password


I want to authenticate to you using NetNTLM
(NTLMSSP_NEGOTIATE)
You have to answer this random server challenge
(NTLMSSP_CHALLENGE)
Here is the answer to the challenge;
(also you have to answer this random client challenge)
(NTLMSSP_AUTH)
Authentication succeeded
(and answer to client challenge)

* Actually, unless the server you are connecting to is the Domain Controller, it doesn‘t know the user‘s NTLM hash so it can‘t validate the response. Instead, it passes the authentication attempt through a secure “NETLOGON tunnel” to the DC. Incidentally, the NETLOGON security is derived from the NTLM hash of the machine account. And the NETLOGON tunnel has its own different challenge-response. It‘s a mess.

NTLM Relaying


I want to authenticate to you using NetNTLM
(NTLMSSP_NEGOTIATE)
You have to answer this *specific* server challenge
(NTLMSSP_CHALLENGE)
Here is the answer to the challenge;
(also you have to answer this random client challenge)
(NTLMSSP_AUTH)
Connection reset
I want to authenticate to you using NetNTLM
(NTLMSSP_NEGOTIATE)
You have to answer this random server challenge
(NTLMSSP_CHALLENGE)
Here is the answer to the challenge;
(also you have to answer this random client challenge)
(NTLMSSP_AUTH)
Authentication succeeded
(and answer to client challenge)

Mitigation: Signing

- NetNTLM protocol can also derive a **session key** from the shared secret
- when signing is enforced, every **message** going through must be **signed** with that session key
- => you can relay authentication, but can‘t communicate after that
- however, not all protocols support it
  - SMB, LDAP(S) does
  - HTTP doesn‘t

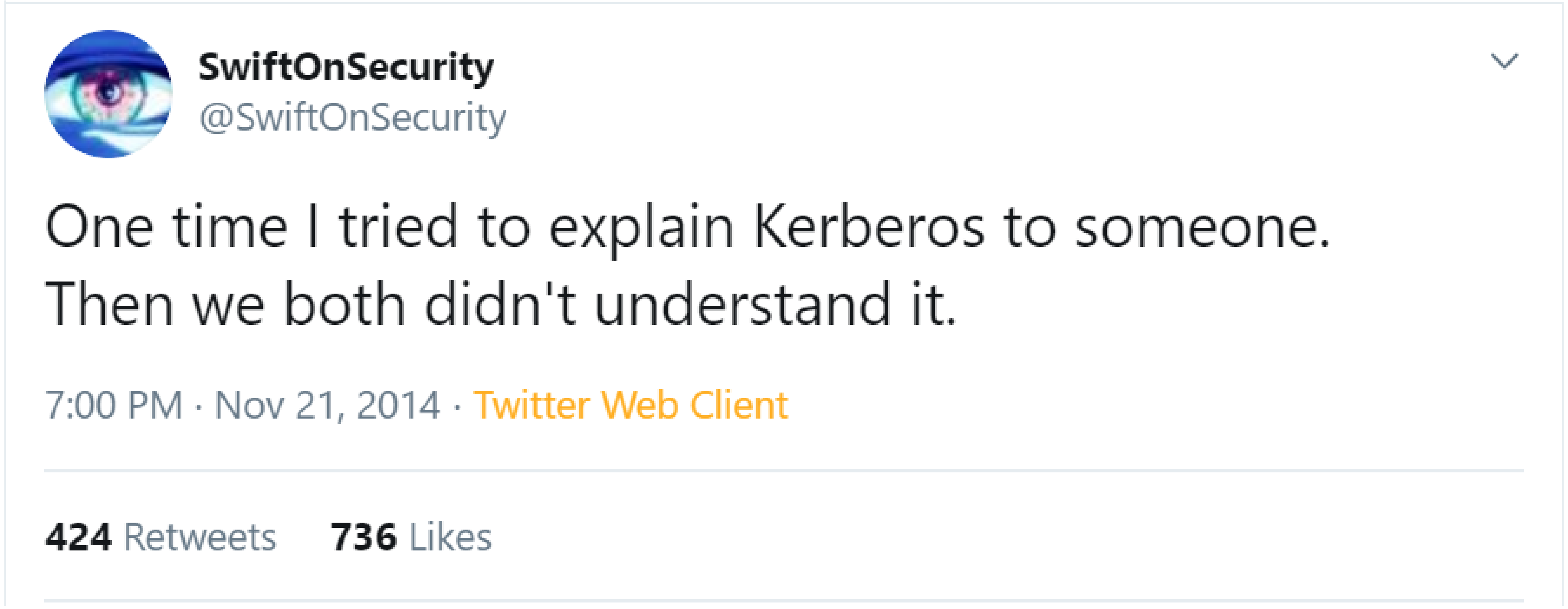
Negotiation

- the first message (NTLMSSP_NEGOTIATE) contains a bit mask of requested security features
  - no signing/sign if available/always sign
  - willingness to do NetNTLMv1
- the message contains a signature called MIC (Message Integrity Code), and cannot be tampered
  - save for implementation bugs *(Drop the MIC - CVE-2019-1040)*
- => you can‘t easily “downgrade” a request for signing

Domain computers have accounts, too!

- every machine that‘s joined to a domain has an account in AD
- this account name always ends with a dollar *(DESKTOP-8KCJBF6$)*
- there are not many important differences to regular users
  - has a (randomly-generated) password
  - can authenticate using NTLM
  - can get Kerberos tickets
  - **can be a victim of NTLM relaying**
- big question: **Can we own a computer by relaying its machine credentials?**

Kerberos



SwiftOnSecurity
@SwiftOnSecurity

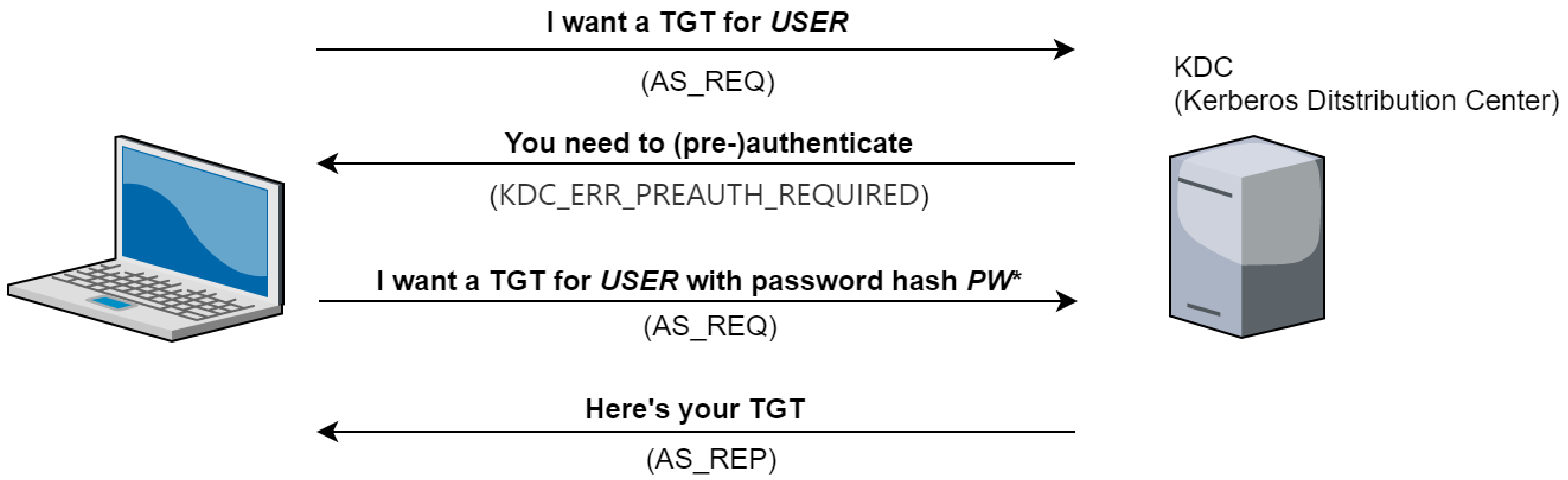
One time I tried to explain Kerberos to someone. Then we both didn't understand it.

7:00 PM · Nov 21, 2014 · Twitter Web Client

424 Retweets **736** Likes

Kerberos: Getting a TGT

- A TGT (Ticket-Granting-Ticket) is basically a “proof” that you authenticated that you can later use to request tickets for services

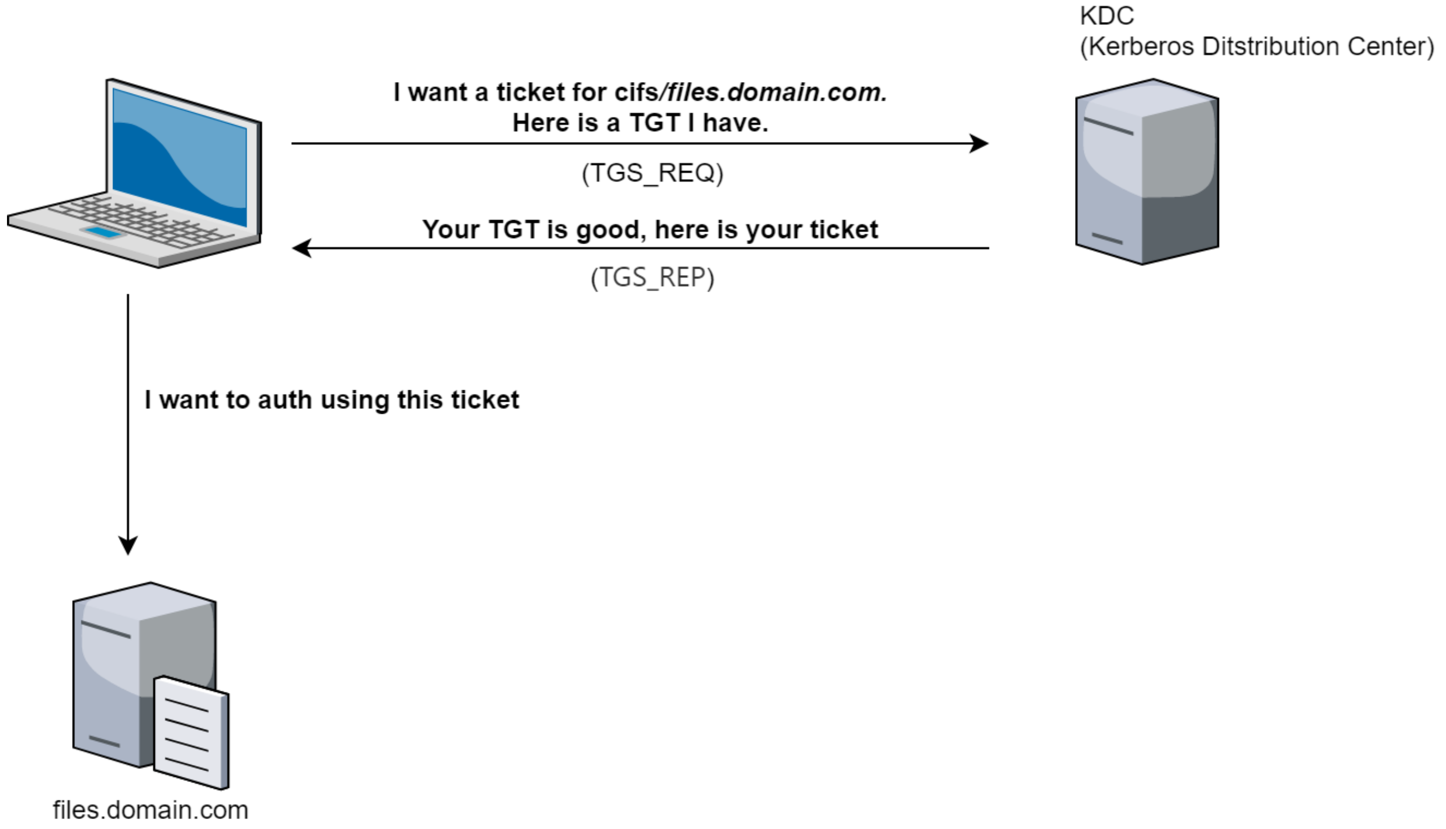

I want a TGT for *USER*
(AS_REQ)
KDC
(Kerberos Ditstribution Center)
You need to (pre-)authenticate
(KDC_ERR_PREAUTH_REQUIRED)
I want a TGT for *USER* with password hash *PW**
(AS_REQ)
Here's your TGT
(AS_REP)

* In original standard Kerberos, you proved yourself to KDC by knowledge of a DES key. Microsoft added in support for using NTLM hashes, which is by default enabled. Standard Kerberos later moved to AES keys, which MS-Kerberos supports, but doesn‘t enforce.

Kerberos: Authenticating to a service

- don‘t think Windows services (background executables), think protocols
- every service has an SPN (Service Principal Name)
  - *ldap/dc.domain.com*
  - *cifs/files.domain.com*: CIFS is SMB filesharing

Kerberos: Authenticating to a service

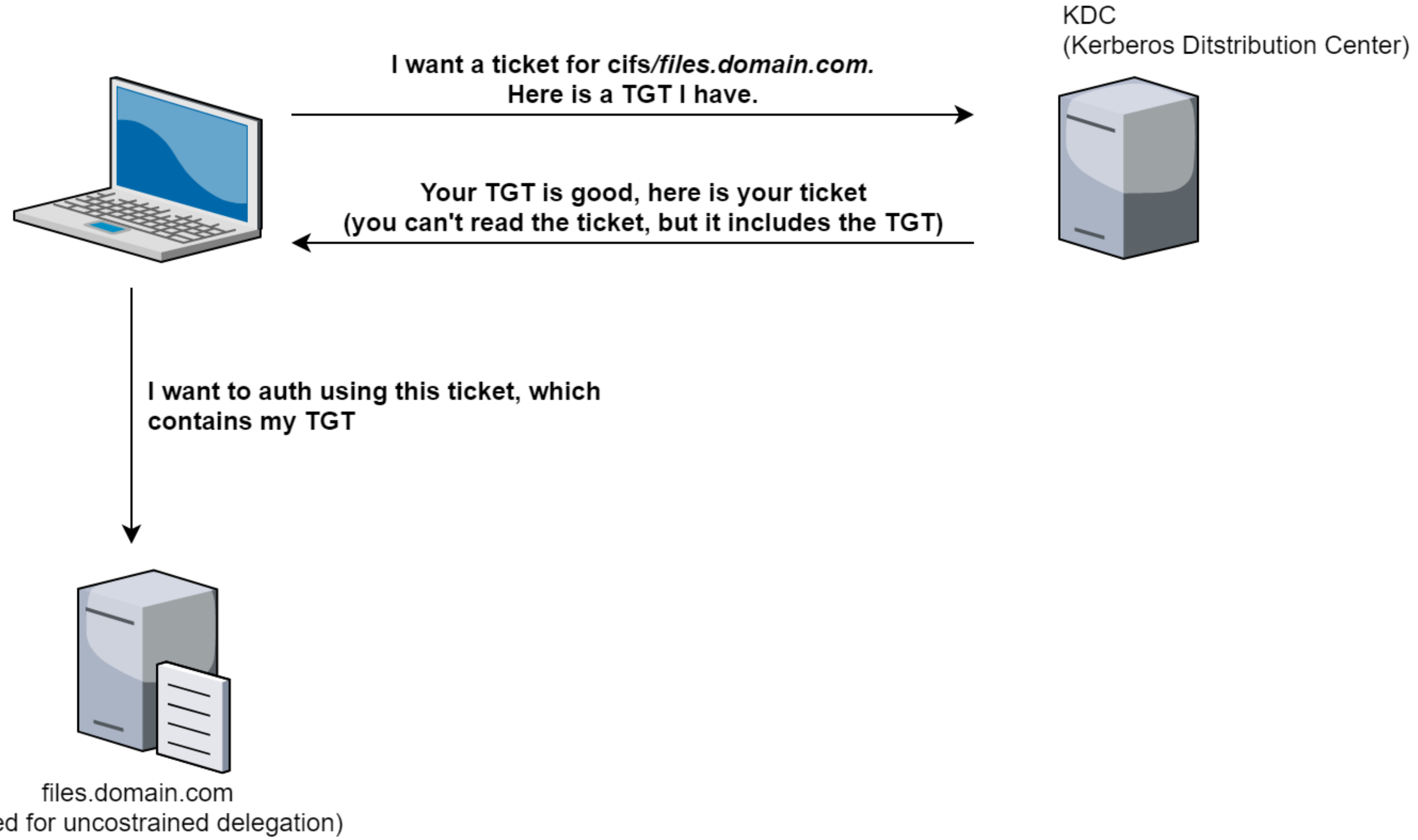

KDC
(Kerberos Ditstribution Center)
I want a ticket for cifs/*files.domain.com.*
Here is a TGT I have.
(TGS_REQ)
Your TGT is good, here is your ticket
(TGS_REP)
I want to auth using this ticket
files.domain.com

* One of the cool things about Kerberos is that files.domain.com doesn‘t have to check in with the KDC at all. The KDC encrypted & signed the ticket with a shared secret it has with files.domain.com, and therefore if the secret isn‘t compromised, the file server can trust the ticket without asking anybody. The shared secret is, once again, our machine account password. If the secret is compromised, you can create “silver tickets”.

Unconstrained Delegation

- if a service is allowed unconstrained delegation, tickets generated for it **contain the TGT decryptable by the target service**
- therefore the service can do everything it could do as if it authenticated with a password
- **however, by default** (and in all sensible configurations), **only domain admins can set up unconstrained delegation**

Unconstrained Delegation

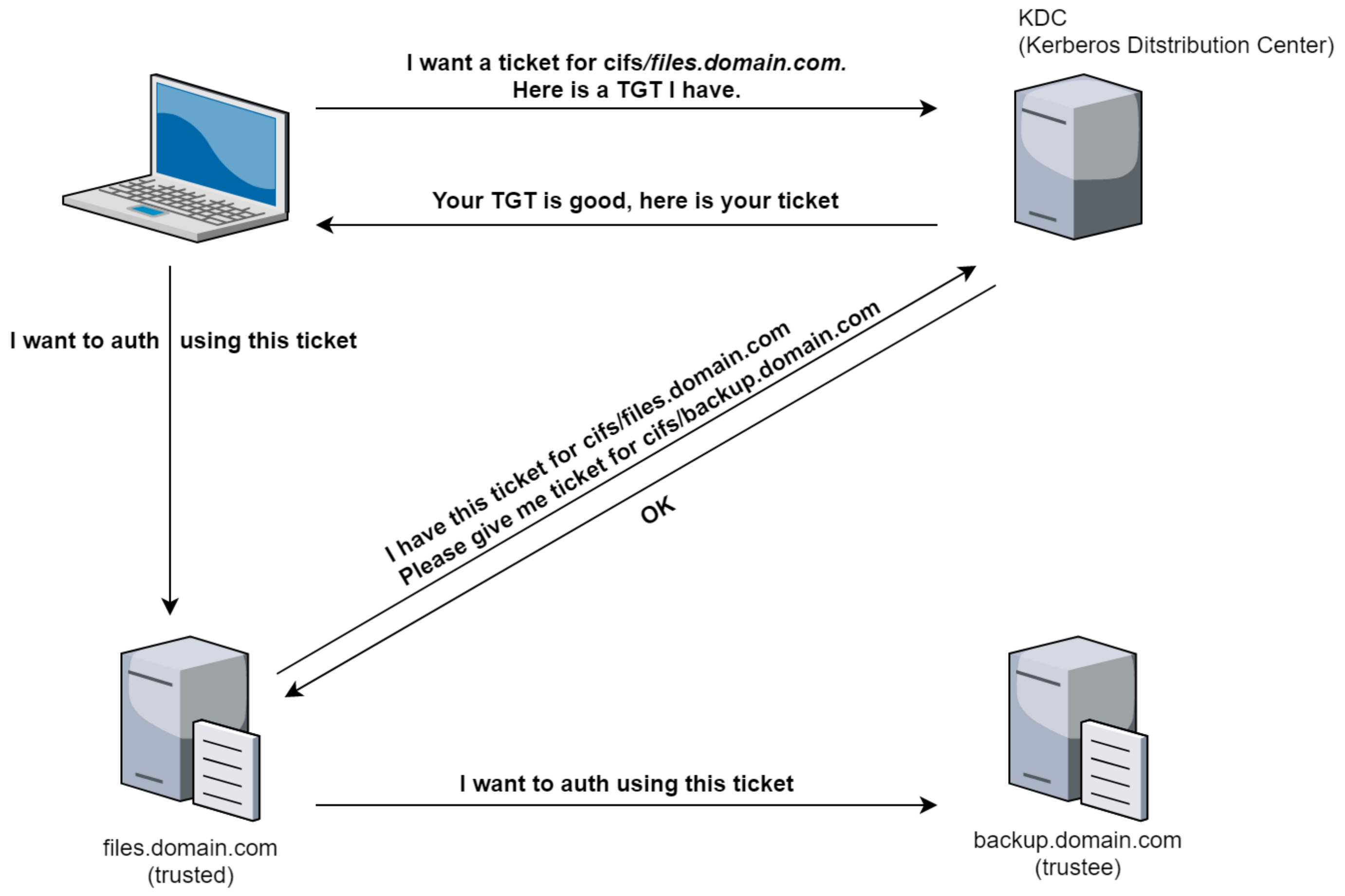

KDC
(Kerberos Ditstribution Center)
I want a ticket for cifs/*files.domain.com.*
Here is a TGT I have.
Your TGT is good, here is your ticket
(you can't read the ticket, but it includes the TGT)
I want to auth using this ticket, which contains my TGT
files.domain.com
(trusted for uncostrained delegation)

Constrained delegation (S4U2Proxy)

- the trust is actually between **two services**, but is most often set between **two machines**
  - let's say PC-A trusts PC-B; then every service on PC-A trusts every service on PC-B
- if the **trusted** service **gets a** forwardable* **ticket authenticating USER** for itself, it can **ask the KDC** to turn it into a **ticket authenticating USER** for the **trustee**
  - when user auths to cifs/PC-B, the service can use the ticket to get a ticket for ldap/PC-A

* By default, most tickets are forwardable. However, members of Protected user or other accounts with USER_NOT_DELEGATED never get a forwardable ticket.

Constrained delegation (S4U2Proxy)


KDC
(Kerberos Ditstribution Center)
I want a ticket for cifs/*files.domain.com.*
Here is a TGT I have.
Your TGT is good, here is your ticket
I want to auth using this ticket
I have this ticket for cifs/files.domain.com
Please give me ticket for cifs/backup.domain.com
OK
I want to auth using this ticket
files.domain.com
(trusted)
backup.domain.com
(trustee)

Constrained delegation (S4U2Proxy)

- originally, the trust was defined on the **trusted** machine, and only domain admins can set it
- however, Microsoft added **resource-based constrained delegation, in which the trustee can set constrained delegation for itself**

Protocol Transition (S4U2Self)

- every service can create tickets for itself out of thin air **for any user**
- what‘s more, **these tickets work as proof for S4U2Proxy**

Plan of attack

1. own any domain-joined machine (ATTACKER$)
2. get VICTIM$ to authenticate to us using NTLM
3. relay VICTIM$ credentials to LDAP, and use them to configure VICTIM$ to trust ATTACKER$
4. get a TGT for ATTACKER$
5. use S4U2Self to create an **Administrator** ticket for cifs/ATTACKER$
6. use S4U2Proxy to turn this into an **Administrator** ticket for cifs/VICTIM$

Own any domain-joined machine

- by default, all accounts (even machine accounts) can **add up to 5 machines to the domain**
  - every sensible admin disables this though
  - if it‘s enabled, you can even be clever and add the machine in step 3, using the relayed credentials

- EternalBlue any old machine, boot kali on a provided workstation, …

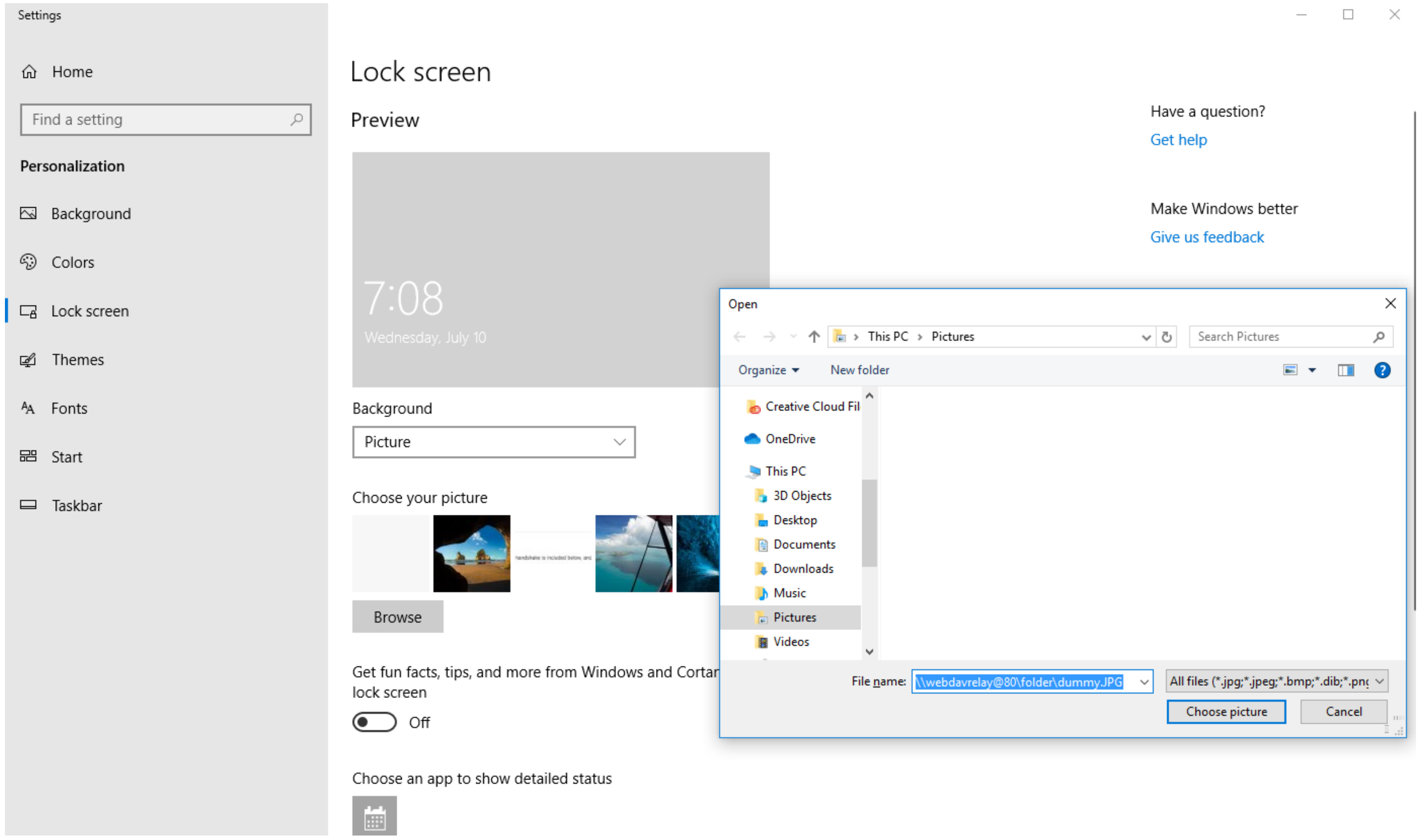
Getting authentication from machines

- physical MITM
  - reconnect the machine to your computer, tell it you‘re the DNS server and resolve everything to yourself
  - reboot the machine
  - eventually, Windows will (hopefully) try to download something (updates, revoked certs, …) **using HTTP** and auth using machine credentials

- MITM6
  - if the network doesn‘t have IPv6 set up, you can promote yourself to a DHCP6 server and confuse the victim

DAV

- change your lockscreen or profile picture



Settings

Home

Find a setting

Personalization

Background

Colors

Lock screen

Themes

Fonts

Start

Taskbar

Lock screen

Preview

7:08
Wednesday, July 10

Background

Picture

Choose your picture

Browse

Get fun facts, tips, and more from Windows and Cortan lock screen

Off

Choose an app to show detailed status

Have a question?
Get help

Make Windows better
Give us feedback

Open

This PC › Pictures

Search Pictures

Organize ▾ New folder

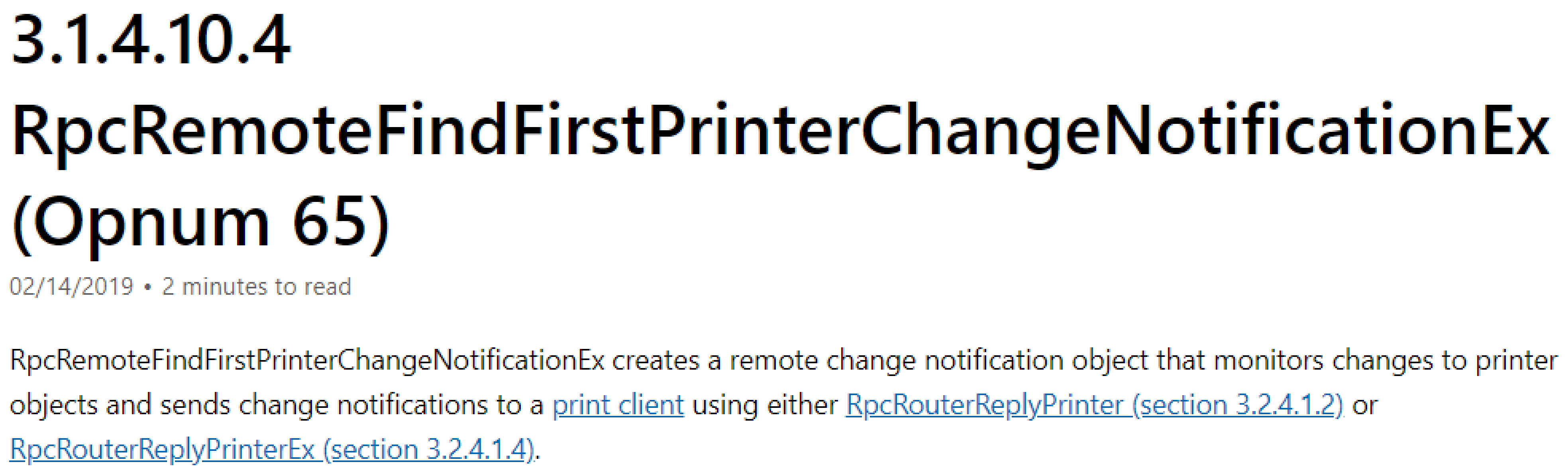
Creative Cloud Fil
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos

File name: \\webdavrelay@80\folder\dummy.JPG

All files (*.jpg;*.jpeg;*.bmp;*.dib;*.pn

Choose picture Cancel

Printer “bug”



3.1.4.10.4 RpcRemoteFindFirstPrinterChangeNotificationEx (Opnum 65)

02/14/2019 • 2 minutes to read

RpcRemoteFindFirstPrinterChangeNotificationEx creates a remote change notification object that monitors changes to printer objects and sends change notifications to a print client using either RpcRouterReplyPrinter (section 3.2.4.1.2) or RpcRouterReplyPrinterEx (section 3.2.4.1.4).

Printer “bug”

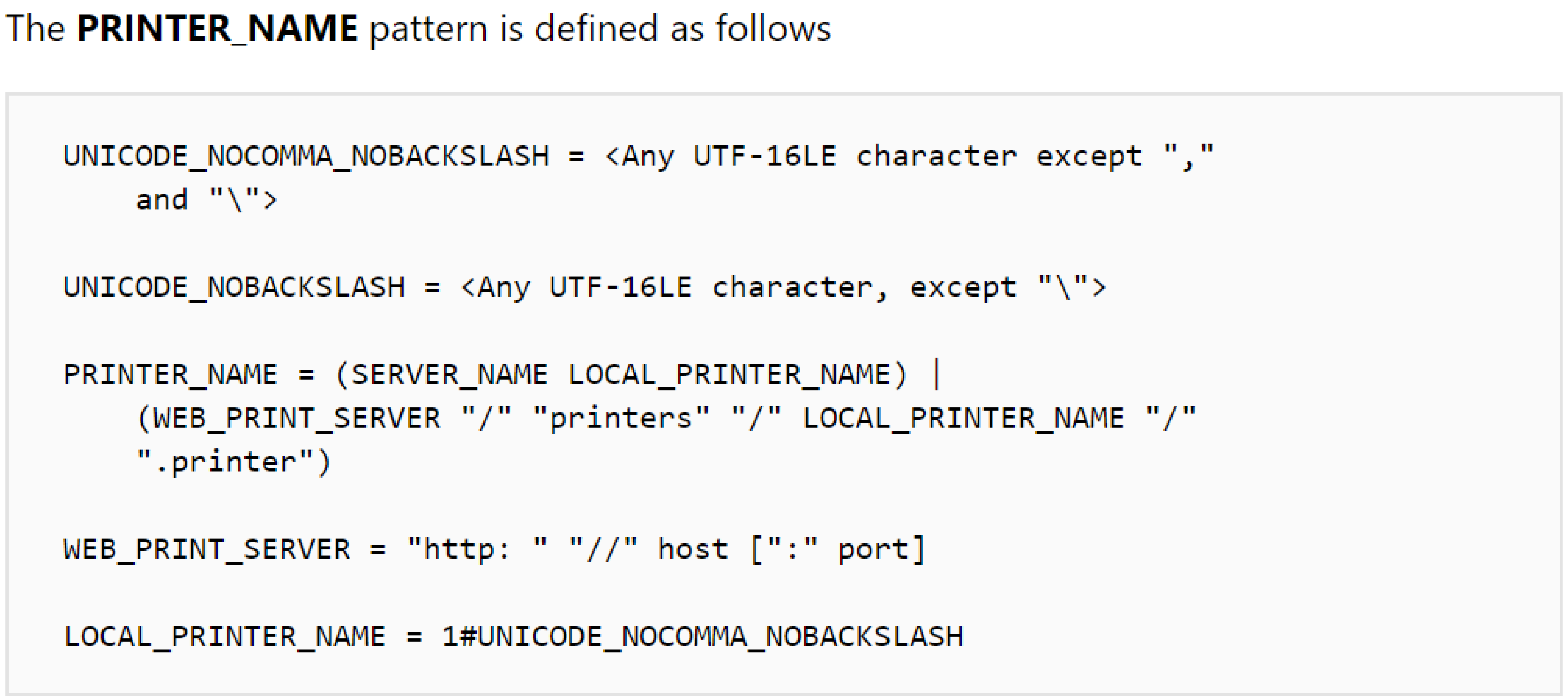
- connect with unprivileged account to SMB
- register \\attacker\ as a notification receiver
- get a SMB connection using the machine account

- most configurations at least **ask for** signing in the negotiation, so it‘s usually not relayable to LDAP, which supports signing

Printer “bug” reloaded

- the available exploits do two calls: OpenPrinter and FindFirstPrinterChangeNotificationEx
- I‘ve read the RPRN protocol docs

Printer "bug" reloaded

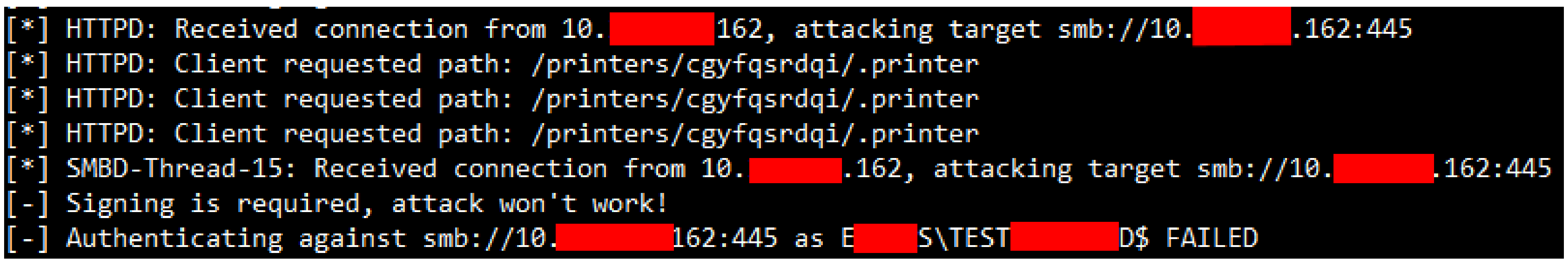


The **PRINTER_NAME** pattern is defined as follows

```
UNICODE_NOCOMMA_NOBACKSLASH = <Any UTF-16LE character except ","
    and "\">

UNICODE_NOBACKSLASH = <Any UTF-16LE character, except "\">

PRINTER_NAME = (SERVER_NAME LOCAL_PRINTER_NAME) |
    (WEB_PRINT_SERVER "/" "printers" "/" LOCAL_PRINTER_NAME "/"
    ".printer")

WEB_PRINT_SERVER = "http: " "//" host [":" port]

LOCAL_PRINTER_NAME = 1#UNICODE_NOCOMMA_NOBACKSLASH
```

Printer “bug” reloaded

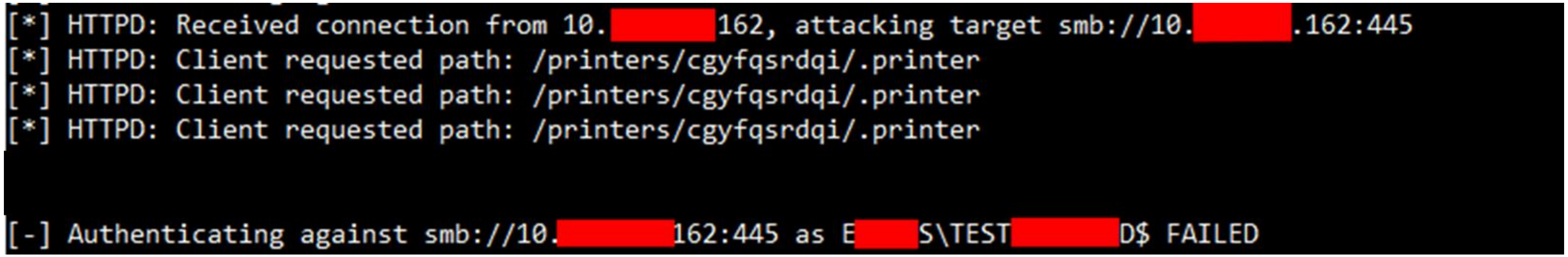
- HTTP > SMB
- so I changed the OpenPrinter from \\self\ to http://attacker/printers/random/.printer



```
[*] HTTPD: Received connection from 10.[redacted]162, attacking target smb://10.[redacted].162:445
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer
[*] SMBD-Thread-15: Received connection from 10.[redacted].162, attacking target smb://10.[redacted].162:445
[-] Signing is required, attack won't work!
[-] Authenticating against smb://10.[redacted]162:445 as E[redacted]S\TEST[redacted]D$ FAILED
```

Printer “bug” reloaded

- what I saw

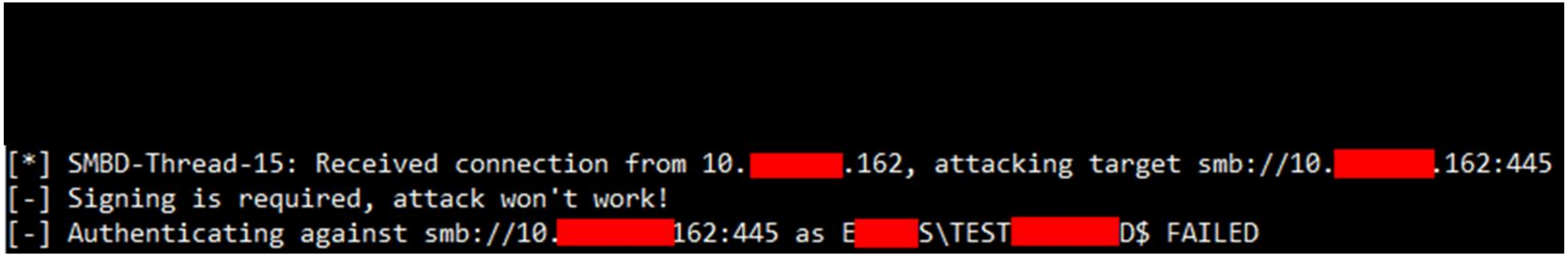


```
[*] HTTPD: Received connection from 10.[redacted]162, attacking target smb://10.[redacted].162:445
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer
[*] HTTPD: Client requested path: /printers/cgyfqsrdqi/.printer


[-] Authenticating against smb://10.[redacted]162:445 as E[redacted]S\TEST[redacted]D$ FAILED
```

- “Hey! It‘s authenticating over HTTP as the machine account! The relay failed for some reason, but this is HUGE!”
- so I registered a talk at CCC

Printer “bug” reloaded

- I tried to reproduce over and over, but couldn‘t
- turns out, what really happened



```
[*] SMBD-Thread-15: Received connection from 10.[redacted].162, attacking target smb://10.[redacted].162:445
[-] Signing is required, attack won't work!
[-] Authenticating against smb://10.[redacted]162:445 as E[redacted]S\TEST[redacted]D$ FAILED
```

Printer “bug” reloaded

- I left the call to FindFirstPrinterChangeNotificationEx unmodified
- the remote machine really connected over HTTP, but it didn‘t send any meaningful credentials
- but after that, the call to PrinterChangeNotification did the regular SMB call

- ¯_(ツ)_/¯

Demo

1. add a machine using unpriv user
2. physical MITM VICTIM$ to authenticate to us using NTLM
3. relay VICTIM$ credentials to LDAP, and use them to configure VICTIM$ to trust ATTACKER$
4. get a TGT for ATTACKER$
5. use S4U2Self to create an **Administrator** ticket for cifs/ATTACKER$
6. use S4U2Proxy to turn this into an **Administrator** ticket for cifs/VICTIM$

Mitigations

- most of the chain is “by design”
- Microsoft would have to change that
- the only real mitigation is preventing relaying
    - force signing for all protocols where available; ban NTLM auth where not available
- since last week, MS recommends LDAP signing
- defense-in-depth:
    - Add critical users to Protected Users, or at least give them USER_NOT_DELEGATED

The end

- epic fails happen, but such is life
- I‘d be more than happy to talk to you later
    - here at CCCamp
    - Twitter @jagotu
- Any questions?