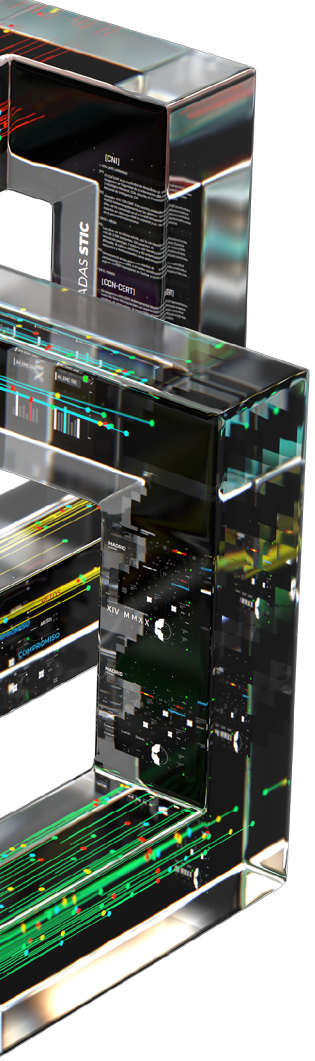
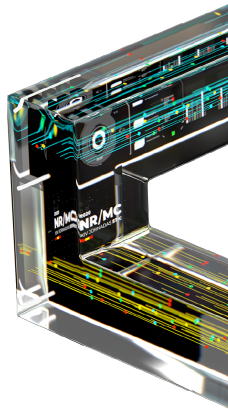


#XIVJORNADASCCNCERT

1472: la tormenta perfecta



e: rjrodriguez@unizar.es
w: http://www.ricardojrodriguez.es
@RicardoJRdez

Ricardo J. Rodríguez

- **Doctor en Informática e Ingeniería de Sistemas**
- **Líneas de investigación:**
  - Análisis binario de aplicaciones
  - Análisis forense de memoria
  - Seguridad en RFID/NFC
- **Miembro de RME-DisCo**, dedicados a investigación sobre seguridad software y de sistemas
  - https://reversea.me

@reverseame

https://t.me/reverseame

Universidad Zaragoza
1542

Índice

1. Introducción

2. Conceptos previos

3. Análisis de la vulnerabilidad CVE-2020-1472

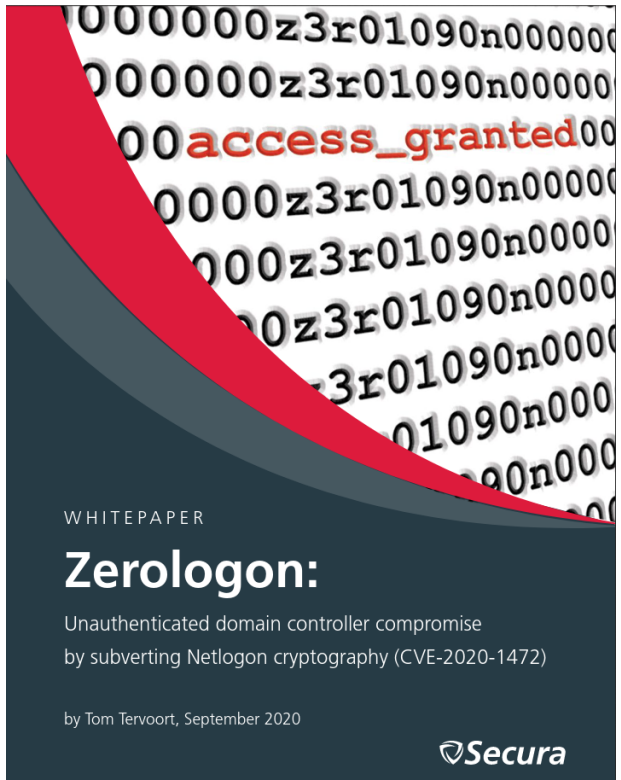
4. Soluciones

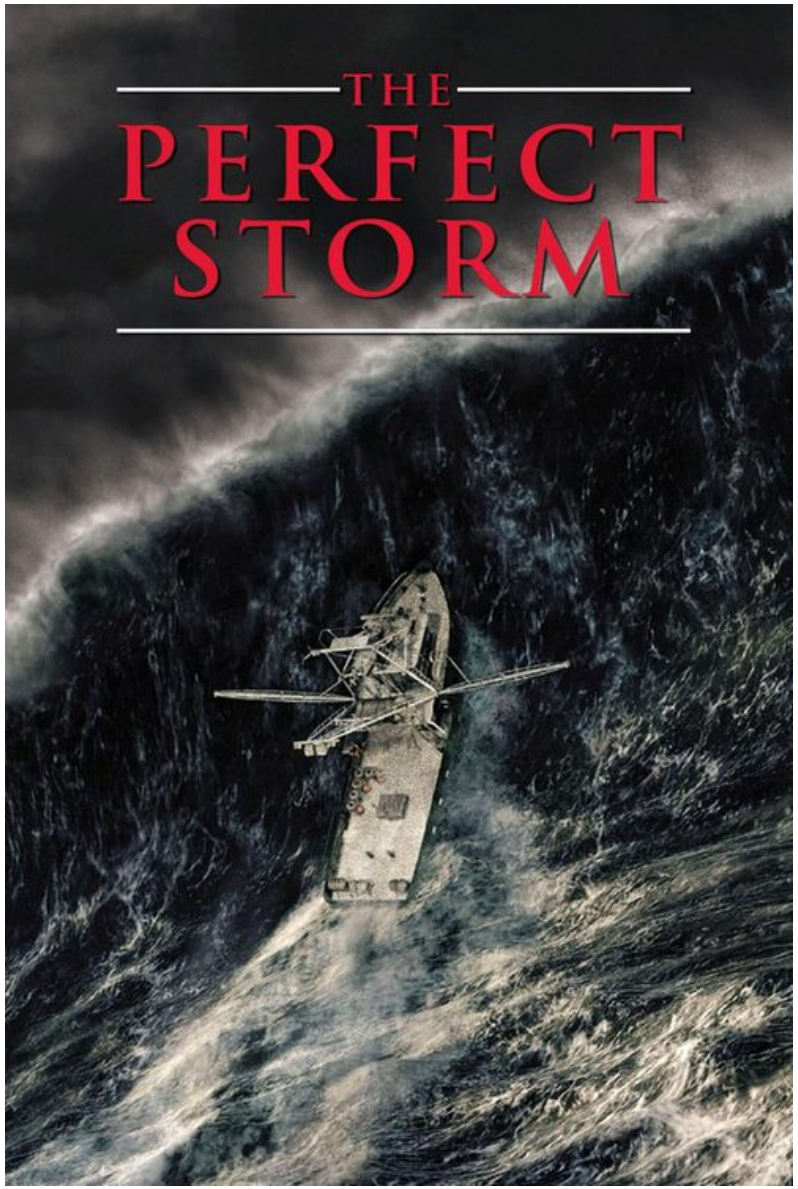
5. Conclusiones

1. Introducción

CVE-2020-1472

- **Vulnerabilidad publicada el 14 de septiembre de 2020**
- También conocida como **Zerologon**
  - Descubierta por Tom Tervoort (de Sectura)
  - Informe técnico en https://www.secura.com/pathtoimg.php?id=2055

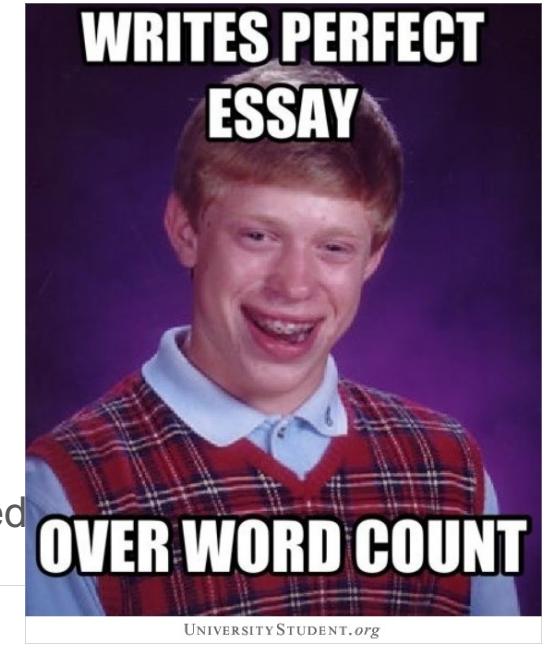




Créditos: eCartelera

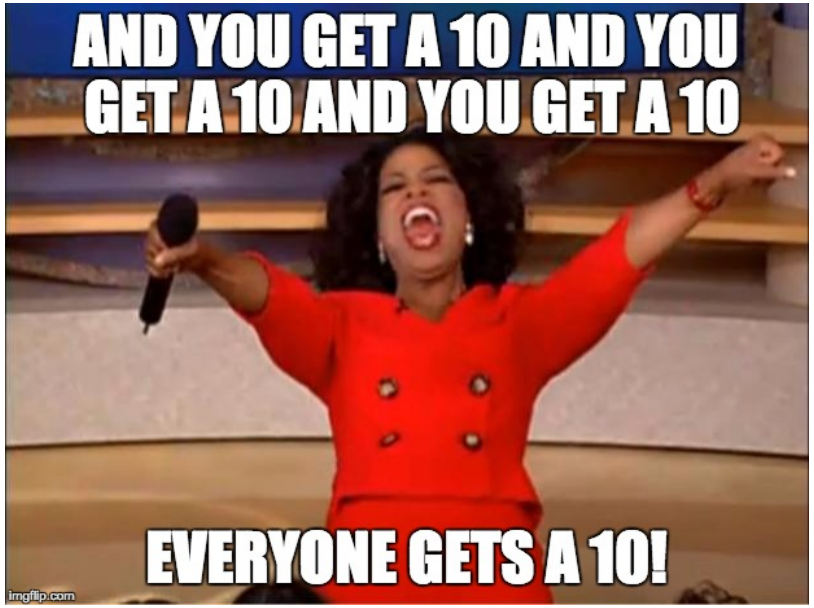
1. Introducción

Impacto de CVE-2020-1472



- **Permite realizar elevación de privilegios a través del protocolo Netlogon**
  - **EN UN ACTIVE DIRECTORY (AD) de Windows**
  - AD: implementación de Microsoft para su implementación de servicio de directorio en red
    Permite diferentes protocolos, como LDAP o Kerberos, entre otros

Los mejores del curso 2020!



CVE-2020-1472 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

An elevation of privilege vulnerability exists when an attacker establishes a vulnerable Netlogon secure channel connection to a domain controller, using the Netlogon Remote Protocol (MS-NRPC), aka 'Netlogon Elevation of Privilege Vulnerability'.

+View Analysis Description

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD | Base Score: 10.0 CRITICAL | Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Créditos: https://nvd.nist.gov/vuln/detail/CVE-2020-1472

1. Introducción

¿Qué necesitamos saber?

- **Sistemas distribuidos**
  - Protocolos RPC: Netlogon

- **Criptografía simétrica**
  - Cifrado de bloque AES y modo de funcionamiento CFB8

- **Análisis de binarios mediante *binary diffing***

2. Conceptos previos

Protocolos RPC

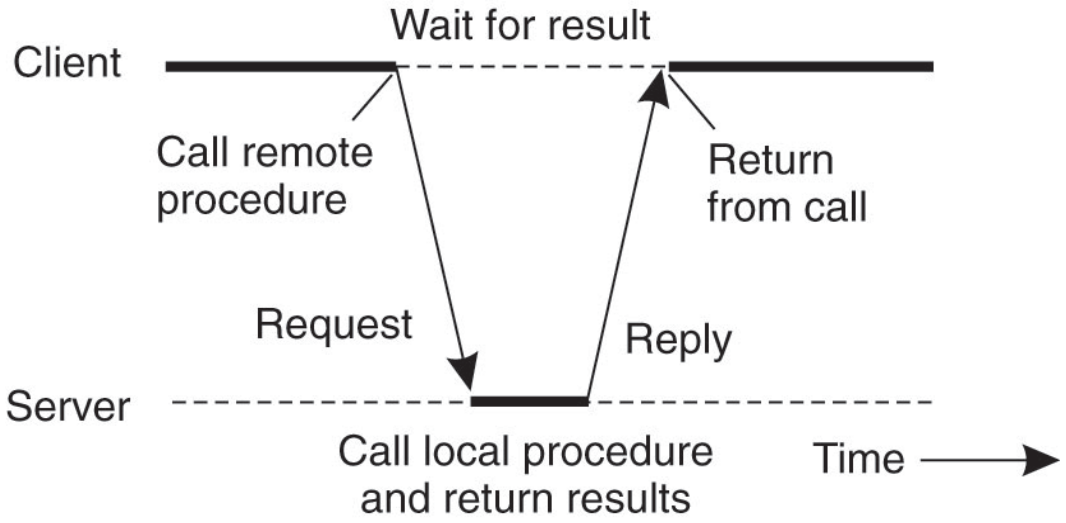
- Propuesto en 1984 (Birrell & Nelson)
- *Remote Procedure Call* (RPC)
  - **Invocación de un procedimiento remoto como si la invocación fuera local**
- **Ventajas**
  - *Simplifican el desarrollo aplicaciones cliente / servidor*
  - *Ocultan la capa de la comunicación de red*
  - *Proporcionan una capa de abstracción*
- Evolución:
  - Protocolos CORBA
  - Java RMI

2. Conceptos previos

Protocolos RPC: modelos

- **RPC síncrono**
  - **Cliente espera respuesta de servidor para continuar**

- **Otros modelos:**
  - RPC asíncrono
  - RPC síncrono diferido
  - RPC one-way (dirección única)



Créditos: A. S. Tanenbaum, M. van Steen; Distributed Systems: Principles and Paradigms, 2nd ed., Pearson Education, 2007

2. Conceptos previos

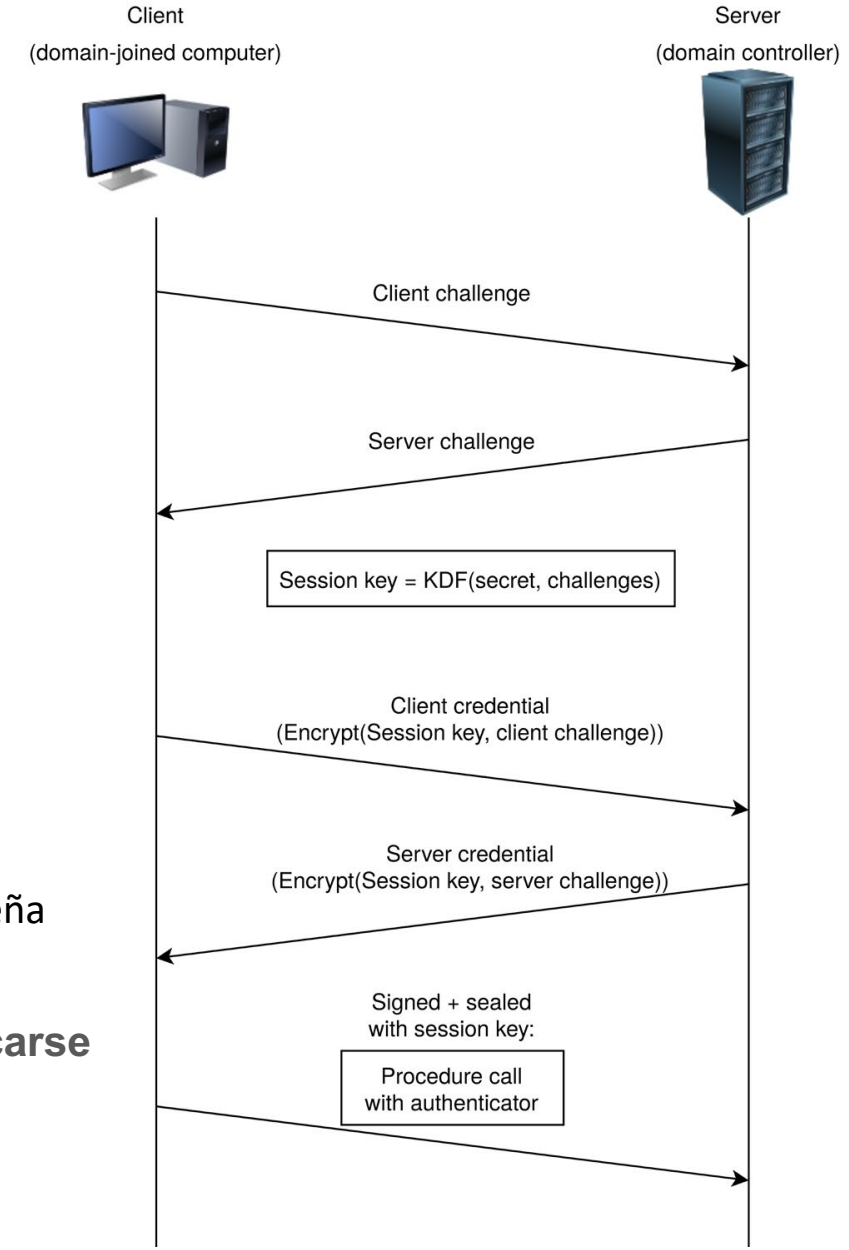
Protocolos RPC: Netlogon

- **MS-NRPC Netlogon Remote Protocol**
  - Especificación abierta: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/ff8f970f-3e37-40f7-bd4b-af7336e4792f
  - RPC síncrono
  - *"… an RPC interface that is used for user and machine authentication on domain-based networks; to replicate the user account database for operating systems earlier than Windows 2000 backup domain controllers; to maintain domain relationships from the members of a domain to the domain controller, among domain controllers for a domain, and between domain controllers across domains; and to discover and manage these relationships"*
  - **Autentica la identidad de un usuario (u otros servicios) ante servicios de red cuando se intenta acceder**
    - Facilita el logueo de usuarios usando el protocolo NTLM de Microsoft
    - Y también actualizar la contraseña del usuario en el dominio
  - Servicio de Windows (se ejecuta en segundo plano)
  - Usa un protocolo criptográfico propio para autentificar a un cliente (usuario del dominio) con un servidor (controlador de dominio) y probar que ambos conocen un mismo secreto
    - **El secreto (compartido) es un hash de la contraseña del usuario**

2. Conceptos previos

Protocolos RPC: Netlogon

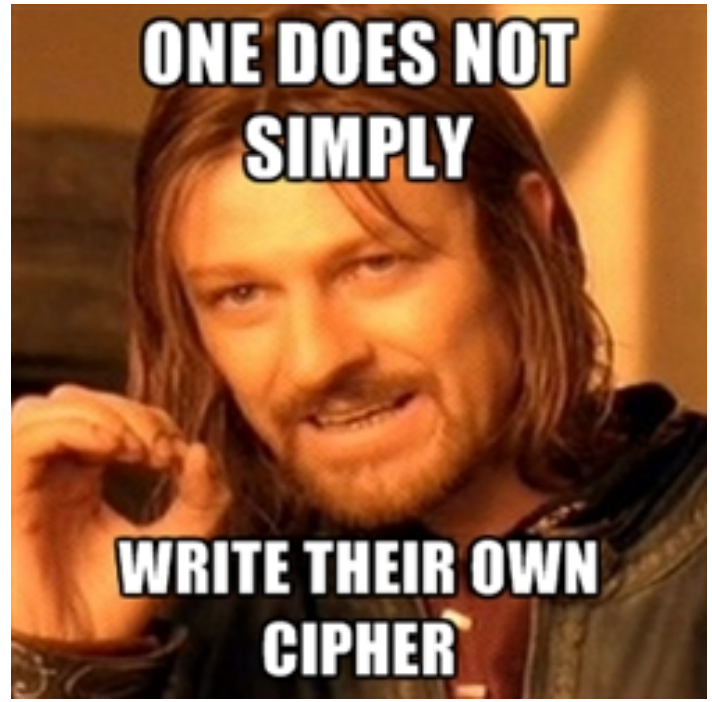
- **Intercambio de claves (hand-shake inicial)**
  1. El cliente inicia la sesión, enviando un challenge
  2. El servidor responde con su challenge
     - Llamados nonces (números únicos), de 8 bytes
  3. Se deriva una clave de sesión (KDF, *key derivation function*)
     - Usando los challenges y el número secreto (propio)
- **Establecimiento de conexión (autenticación de la conexión)**
  1. Usando la clave de sesión, el cliente genera sus credenciales
  2. El servidor calcula de nuevo esas credenciales
     - Si coinciden, se asume que el cliente conoce la clave de sesión y la contraseña
- **Netlogon permite cifrar y firmar todos los mensajes después de autenticarse**
  - Evita ataques basados en red
  - Si no está activo, y la operación es sensible, se necesita un elemento autenticador (que se calcula con la clave de sesión)



Créditos: T. Tervoort, “Zerologon:Unauthenticated domain controller compromise by subverting Netlogon cryptography (CVE-2020-1472)”, Secura, Septiembre 2020

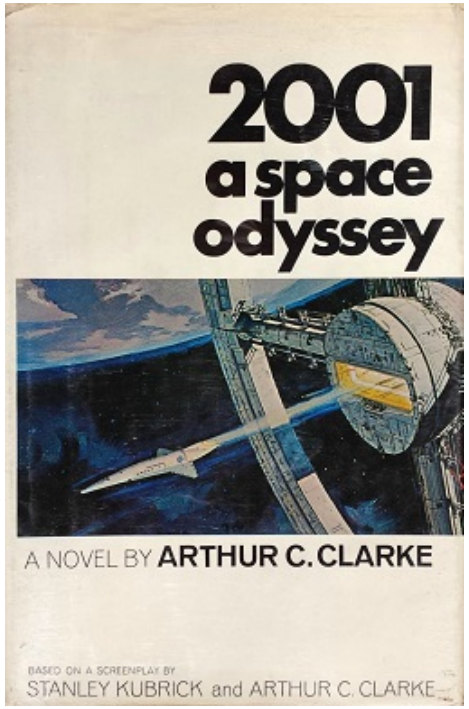
2. Conceptos previos

Protocolos RPC: Netlogon



- **Don't roll your own**: **Primera regla de la criptografía**

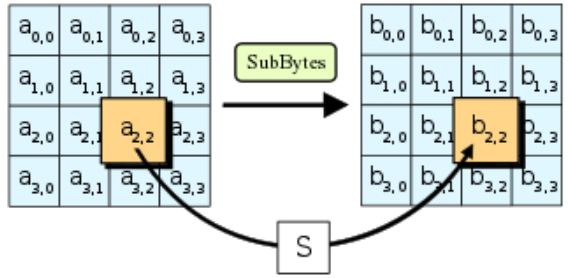
  "Anyone can invent an encryption algorithm they themselves can't break; it's much harder to invent one that no one else can break" - Bruce Schneier

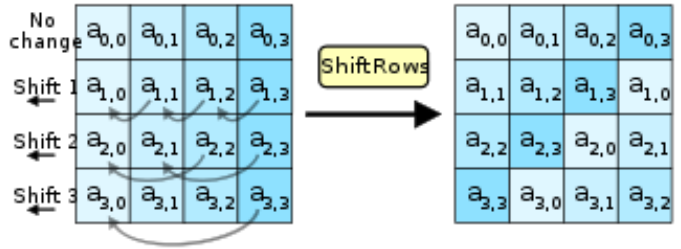


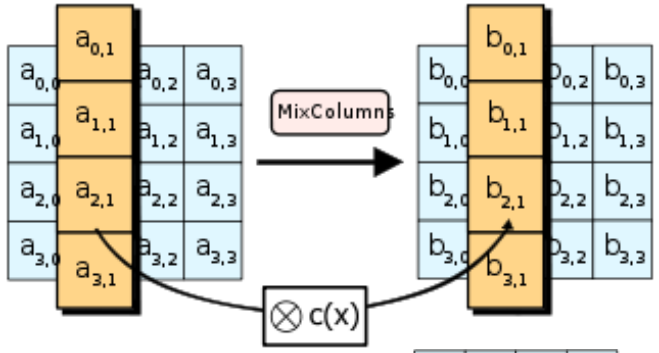
- **CVE-2019-1424**
  - *Notable alto*: **8.1**
    - https://nvd.nist.gov/vuln/detail/CVE-2019-1424
  - Descubierta también por T. Tervoort
  - Si un ataque *person-in-the-middle* es posible (puede ver y modificar tráfico), se puede conseguir acceso de administrador local

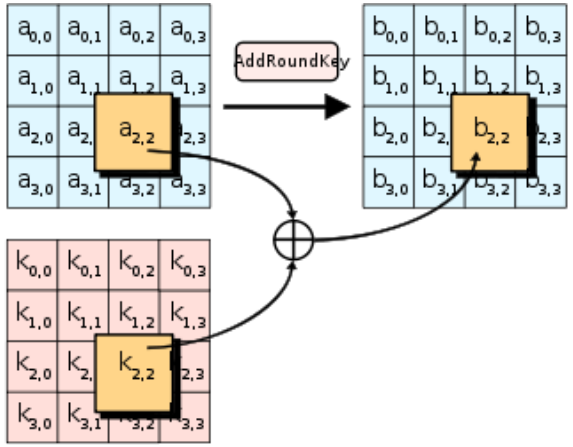
2. Conceptos previos

Criptografía: AES









- **AES (*Advanced Encryption Standard*)**
  - **Algoritmo de criptografía simétrica**
  - **Cifrado por bloque**
  - Desarrollado por Joan Daemen y Vincent Rijmen (Bélgica)
- **Funcionamiento**:
  - Tamaño de bloque de 128 bits
  - Clave variable (128, 192 ó 256)
  - Rondas: (10, 12 ó 14; depende del tamaño de la clave)
    - Entrada: bloque de 128 bits
    - Salida: bloque de 128 bits
    - Diferentes etapas
      - SubBytes, ShiftRows, MixColumn, AddRoundKey
  - Matriz de estado:
    - 16 bytes puestos en columnas

Créditos: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

2. Conceptos previos

Criptografía: Cifrado por bloque

- **Un bloque es un conjunto de bits, de longitud fija**
  - El modo de operación define cómo se aplica la operación de (des)cifrado a tamaño del bloque
- **Vector de inicialización (IV)**
  - Bits usados para aleatorizar el cifrado y garantizar que el texto cifrado es d texto plano varias veces
- **Diferentes modos de operación**
  - OJO: algunos modos tienen problemas (e.g., AES-ECB, AES-CBC)

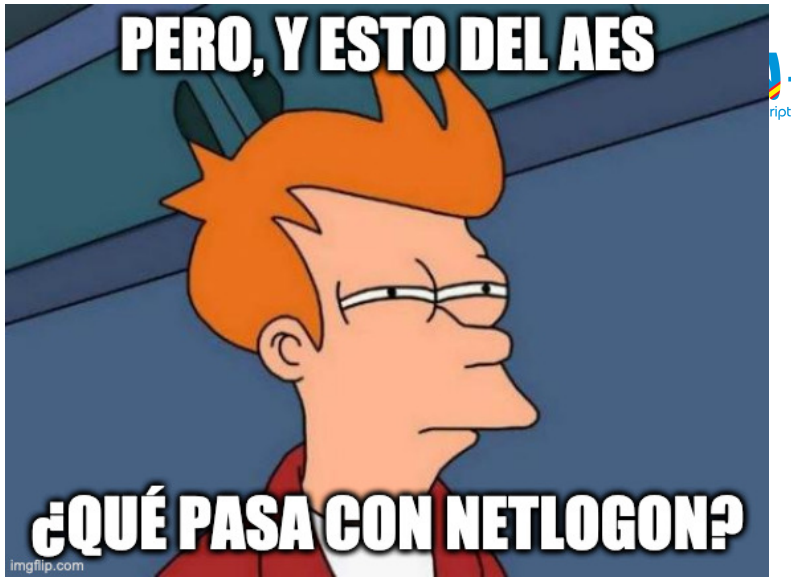
Summary of modes

Mode		Formulas	
Electronic codebook	(ECB)	$Y_i = F(\text{PlainText}_i, \text{Key})$	Y_i
Cipher block chaining	(CBC)	$Y_i = \text{PlainText}_i \text{ XOR } \text{Ciphertext}_{i-1}$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Propagating CBC	(PCBC)	$Y_i = \text{PlainText}_i \text{ XOR } (\text{Ciphertext}_{i-1} \text{ XOR } \text{PlainText}_{i-1})$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Cipher feedback	(CFB)	$Y_i = \text{Ciphertext}_{i-1}$	Plaintext XOR $F(Y, \text{Key})$; $\text{Ciphertext}_0 = IV$
Output feedback	(OFB)	$Y_i = F(Y_{i-1}, \text{Key})$; $Y_0 = F(IV, \text{Key})$	Plaintext XOR Y_i
Counter	(CTR)	$Y_i = F(IV + g(i), \text{Key})$; $IV = \text{token}()$	Plaintext XOR Y_i

Créditos: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

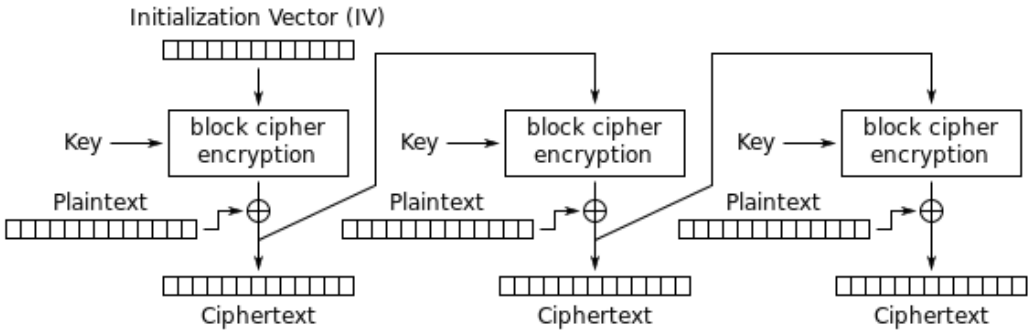
2. Conceptos previos

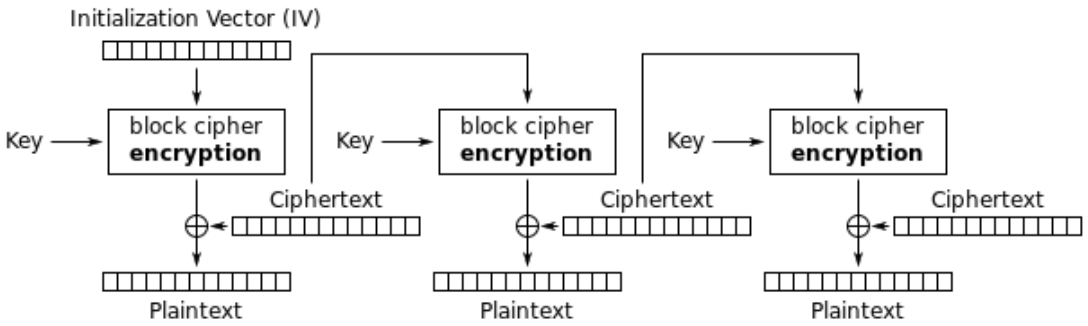
Criptografía: AES-CFB



- **Las credenciales de Netlogon las genera la función *NlComputeCredentials***
  - Recibe un parámetro de 8 bytes y realiza la derivación de la clave de sesión
  - Dos versiones, depende de la configuración del cliente:
    - 2DES (por defecto, se rechaza por el servidor)
    - AES: con modo CFB
- **Modo CFB (*cipher feedback*)**

$$C_i = \begin{cases} \text{IV}, & i = 0 \\ E_K(C_{i-1}) \oplus P_i, & \text{otherwise} \end{cases}$$
$$P_i = E_K(C_{i-1}) \oplus C_i,$$

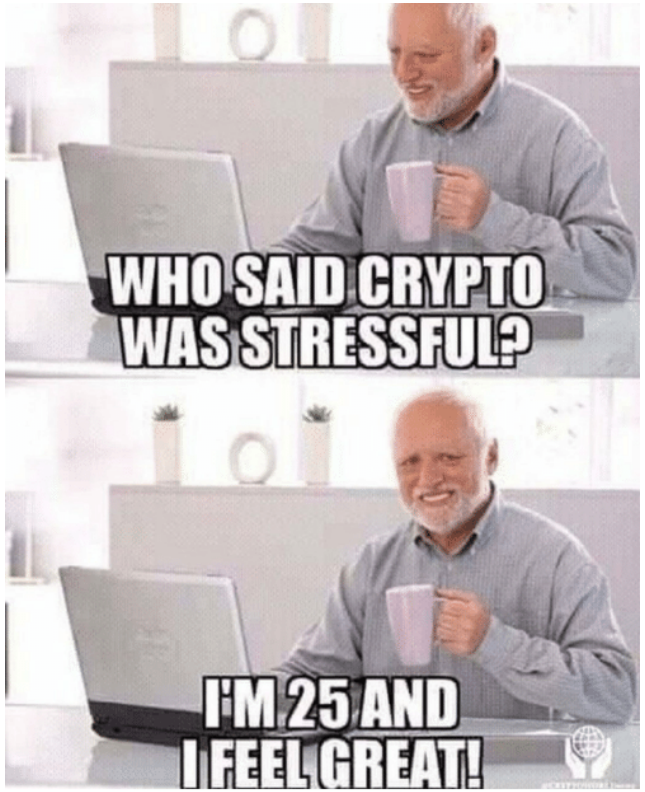




Créditos: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

2. Conceptos previos

Criptografía: AES-CFB



- En NIST SP800-38A (https://csrc.nist.gov/publications/detail/sp/800-38a/final) se define una anchura $1 \leq s \leq b$ para el modo CFB
  - Indica el tamaño de cada trozo de texto plano/cifrado
  - Indirectamente, se trunca la salida del cifrador de bloque

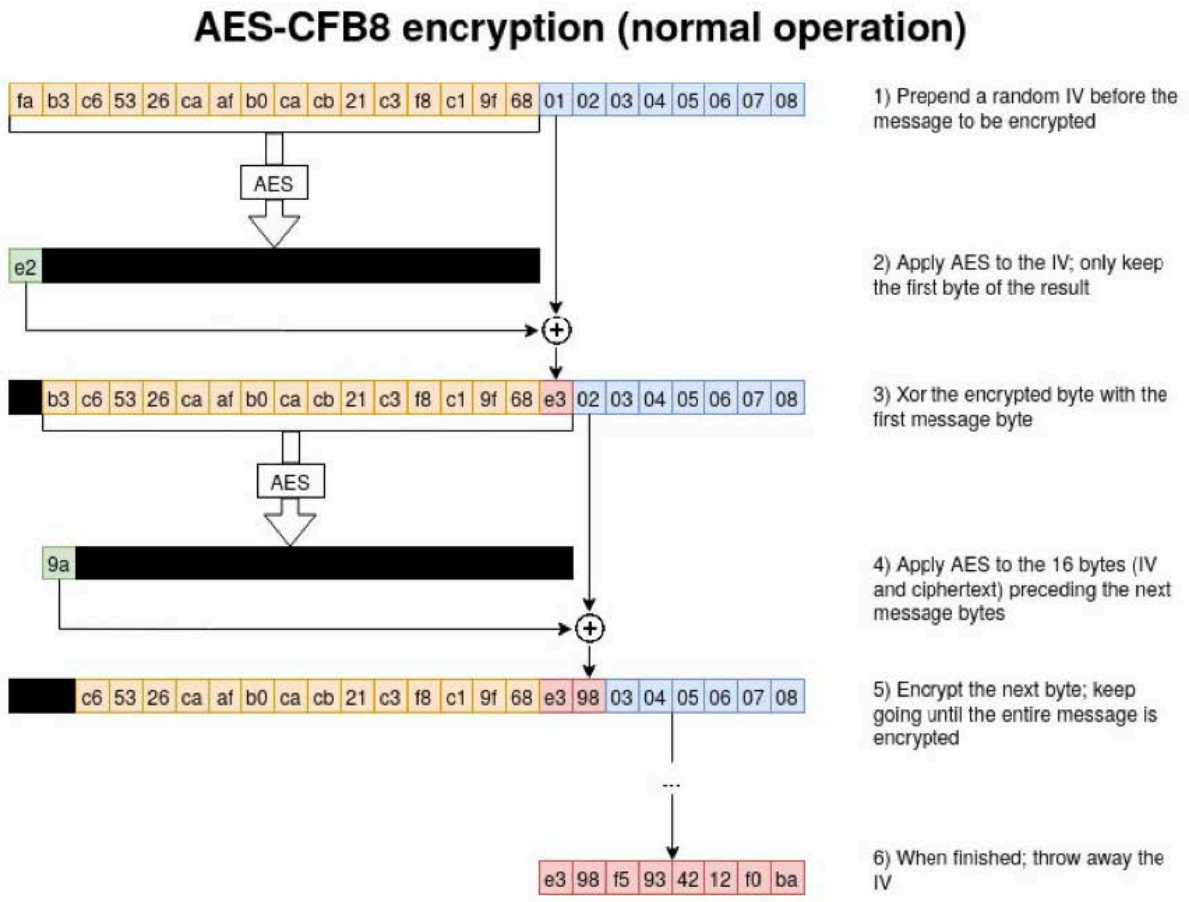
$$
\begin{aligned}
I_0 &= \mathrm{IV}. \\
I_i &= \left((I_{i-1} \ll s) + C_i\right) \bmod 2^b, \\
C_i &= \mathrm{MSB}_s\left(E_K(I_{i-1})\right) \oplus P_i, \\
P_i &= \mathrm{MSB}_s\left(E_K(I_{i-1})\right) \oplus C_i,
\end{aligned}
$$

  - El tamaño se suele acompañar al nombre: CFB1 (1 bit), CFB8 (8 bits), CFB64 (64 bits), CFB128 (128 bits)
- **CFB8**
  - Creado para minimizar la propagación de errores en un canal con ruido (mejor usar CRCs u otros…)
  - **Es lento**: concretamente, 16 veces más lento
    - **Requiere una operación de cifrado de bloque PARA CADA BYTE**

Créditos: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

2. Conceptos previos

Criptografía: AES-CFB8



Créditos: T. Tervoort, “Zerologon:Unauthenticated domain controller compromise by subverting Netlogon cryptography (CVE-2020-1472)”, Secura, Septiembre 2020

2. Conceptos previos

Criptografía: AES-CFB8

- **Función *NIComputeCredentials***
  - Define un IV fijo, de 16 bytes, de todo ceros
- **Uso inseguro de AES-CFB8**
  - Las propiedades de seguridad se mantienen sólo cuando los IV son aleatorios
- Dado un mensaje $m = 0^N$ (es decir, todo ceros) y dado un IV de todo ceros, existe una probabilidad $P = \frac{1}{256}$ de obtener como mensaje cifrado un mensaje con todo ceros
- Sin pérdida de generalidad, si IV tiene todo ceros, $\exists x \in \mathbb{N}, 0 \leq x \leq 255 : P_i = x \therefore C_i = 0, 1 \leq i \leq N$, donde x está distribuido uniformemente de manera aleatoria
  - **Es decir, no es necesario que sean ceros los bytes del mensaje, si no <u>que sean idénticos</u>** (esto lo recordaremos más adelante…)

3. Vulnerabilidad CVE-2020-1472

Análisis

- **NetrServerAuthenticate3 autentifica a un cliente después del hand-shake**
  - Usa las credenciales que recibe de la función NlComputeCredentials, que usa el challenge enviado anteriormente por el cliente
  - El valor de challenge puede ser arbitrariamente puesto por el cliente (e.g., todo ceros)
  - 1 de cada 256 claves de sesión será todo ceros ☺

3. Vulnerabilidad CVE-2020-1472

Análisis

Echemos un vistazo a otras funciones dentro de netlogon.dll….

- **NlComputeCredentials**:
  - Llamadas a BCryptGetProperty y BCryptGenerateSymmetricKey para gestión de cifrado AES
- **NtlnitializeCNG**:
  - Uso de AES + CFB
- **NlComputeCredentials**:
  - Llamada a BCryptEncrypt con IV cero

3. Vulnerabilidad CVE-2020-1472

Análisis

Decompile: NlComputeCredentials - (netlogon_vuln.dll)

```
42    local_42 = *(undefined *)((longlong)param_3 + 0xd);
43    iVar1 = (*(code *)ImgDelayDescr_IAT@1800bd2a0)(local_40,&local_48,param_2);
44    if (iVar1 < 0) {
45      NlAssertFailed("NT_SUCCESS(Status)");
46    }
47    lVar3 = 7;
48    piVar5 = &local_48;
49    while (lVar3 != 0) {
50      lVar3 = lVar3 + -1;
51      *(undefined *)piVar5 = 0;
52      piVar5 = (int *)((longlong)piVar5 + 1);
53    }
54  }
55  else {
56    local_80 = 0;
57    local_54 = 0;
58    local_58 = 0;
59    local_88 = &local_58;
60    iVar1 = (*(code *)__imp_BCryptGetProperty)(NlGlobalAesHandle,L"ObjectLength",&local_54);
61    hMem = (HLOCAL)0x0;
62    if (((-1 < iVar1) && (local_58 == 4)) &&
63       (hMem = LocalAlloc(0x40,(ulonglong)local_54), hMem != (HLOCAL)0x0)) {
64      local_78 = (undefined8 *)((ulonglong)local_78 & 0xffffffff00000000);
65      uVar6 = (ulonglong)local_54;
66      local_80 = 0x10;
67      local_88 = param_3;
68      uVar2 = (*(code *)__imp_BCryptGenerateSymmetricKey)(NlGlobalAesHandle,&local_50);
```

Decompile: NlInitializeCNG - (netlogon_vuln.dll)

```
1
2   uint NlInitializeCNG(void)
3
4   {
5     uint uVar1;
6     wchar_t *pwVar2;
7     undefined8 uVar3;
8     int local_res8 [2];
9     ulonglong in_stack_fffffffffffffe8;
10
11    local_res8[0] = 0;
12    uVar3 = 0;
13    uVar1 = (*(code *)__imp_BCryptOpenAlgorithmProvider)(&NlGlobalAesHandle,L"AES");
14    if ((int)uVar1 < 0) {
15      pwVar2 = L"NlInitializeCNG: Failed to open AES algorithm 0x%x\n";
16    }
17    else {
18      uVar3 = 0x20;
19      uVar1 = (*(code *)__imp_BCryptSetProperty)
20                        (NlGlobalAesHandle,L"ChainingMode",L"ChainingModeCFB",0x20,
21                         in_stack_fffffffffffffe8 & 0xffffffff00000000);
```

3. Vulnerabilidad CVE-2020-1472

Análisis

Decompile: NlComputeCredentials - (netlogon_vuln.dll)

```
65      uVar6 = (ulonglong)local_54;
66      local_80 = 0x10;
67      local_88 = param_3;
68      uVar2 = (*(code *)__imp_BCryptGenerateSymmetricKey)(NlGlobalAesHandle,&local_50);
69      if ((int)uVar2 < 0) {
70        pwVar4 = L"NlComputeCredentials: Failed to initialize encryption 0x%08x\n";
71      }
72      else {
73        local_60 = 0;
74        local_68 = &local_58;
75        uVar6 = 0;
76        local_70 = 8;
77        local_80 = 0;
78        local_88 = (int *)0x0;
79        local_78 = param_2;
80        uVar2 = (*(code *)__imp_BCryptEncrypt)(local_50,param_1);
81        if (-1 < (int)uVar2) goto LAB_180015157;
82        pwVar4 = L"NlComputeCredentials: Failed to encrypt credential 0x%08x\n";
83      }
84      NlPrintRoutine(0x100,pwVar4,(ulonglong)uVar2,uVar6);
85    }
86  }
```

```
else
{
  LODWORD(v8) = BCryptEncrypt(phKey, v7, 8u, 0i64, 0i64, 0, v6, 8u, &pcbResult, 0);
  if ( (int)v8 >= 0 )
    goto LABEL_7;
  v9 = L"NlComputeCredentials: Failed to encrypt credential 0x%08x\n";
```

BCryptEncrypt function (bcrypt.h)

12/05/2018 • 4 minutes to read

The **BCryptEncrypt** function encrypts a block of data.

Syntax

C++ | Copy

```
NTSTATUS BCryptEncrypt(
  BCRYPT_KEY_HANDLE hKey,
  PUCHAR            pbInput,
  ULONG             cbInput,
  VOID              *pPaddingInfo,
  PUCHAR            pbIV,
  ULONG             cbIV,
  PUCHAR            pbOutput,
  ULONG             cbOutput,
  ULONG             *pcbResult,
  ULONG             dwFlags
);
```

`pbIV`

The address of a buffer that contains the initialization vector (IV) to use during encryption. The *cbIV* parameter contains the size of this buffer. This function will modify the contents o this buffer. If you need to reuse the IV later, make sure you make a copy of this buffer before calling this function.

This parameter is optional and can be **NULL** if no IV is used.

The required size of the IV can be obtained by calling the BCryptGetProperty function to get the **BCRYPT_BLOCK_LENGTH** property. This will provide the size of a block for the algorithm, which is also the size of the IV.

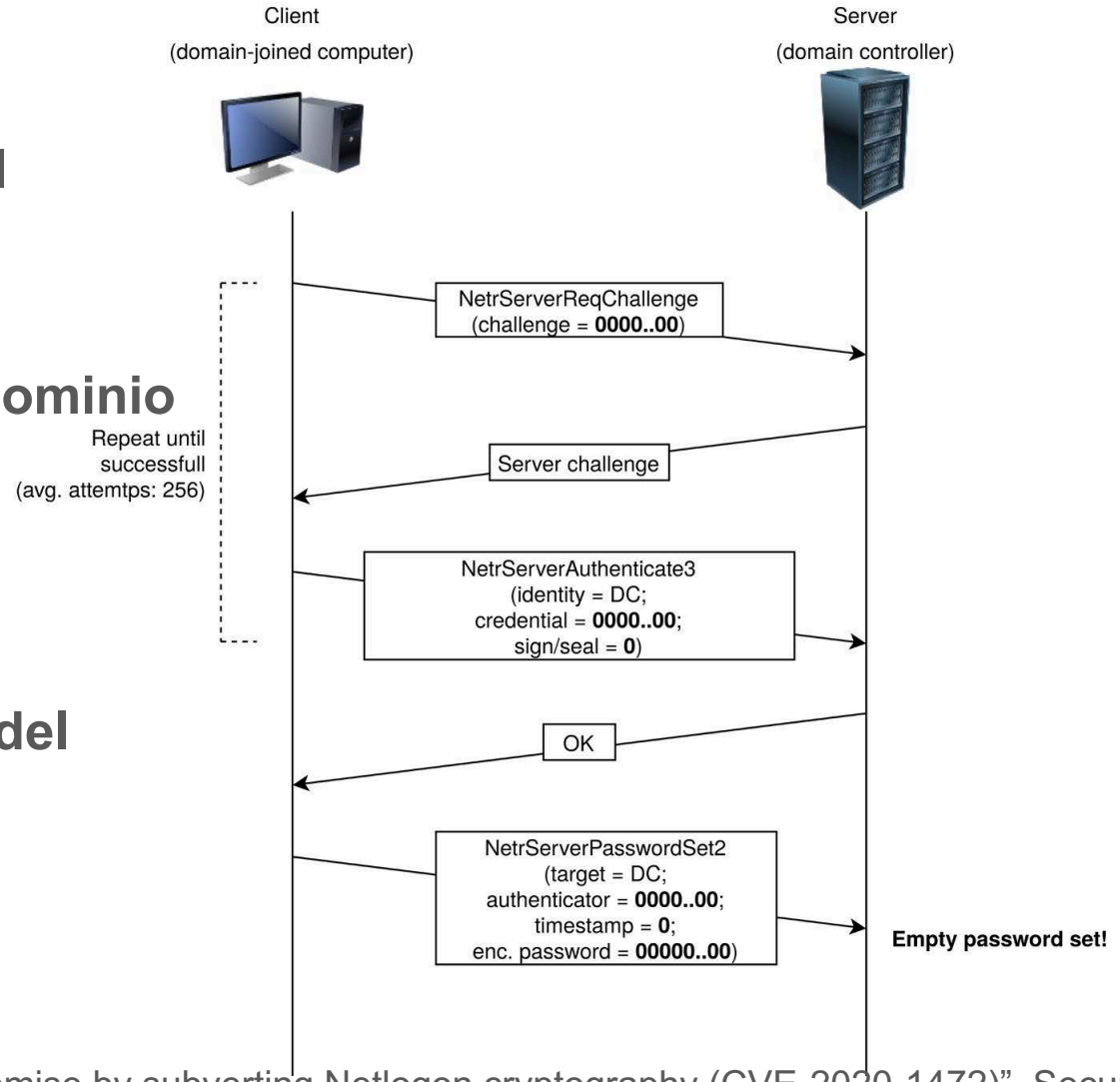
`cbIV`

The size, in bytes, of the *pbIV* buffer.

3. Vulnerabilidad CVE-2020-1472

Impacto

- **Acceso al dominio como cualquier usuario del dominio (hasta el propio controlador)**
- **Cambio de la contraseña del controlador del dominio**
- **Autentificar cualquier RPC**
- **Elevación de privilegios (hasta administrador del dominio)**

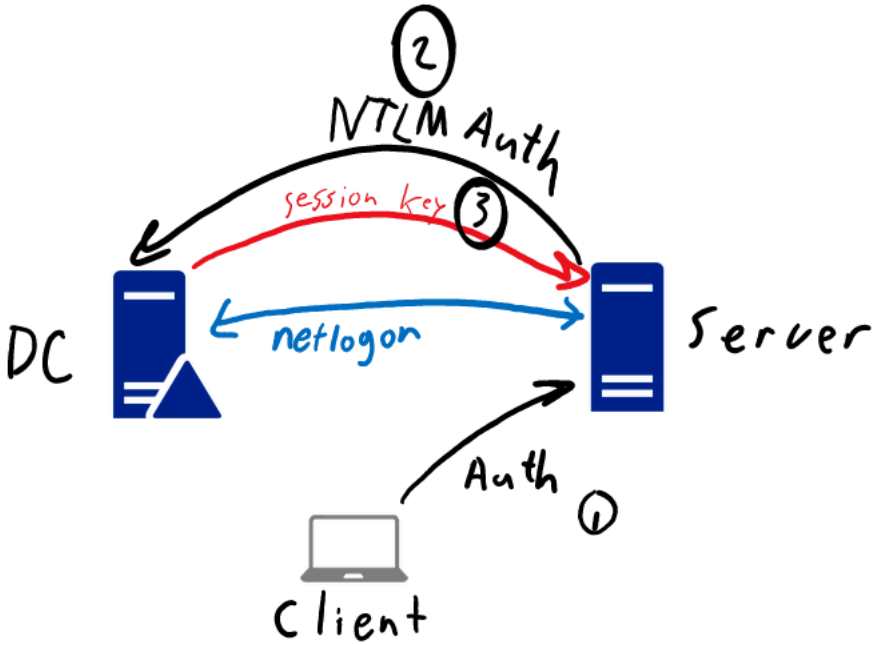


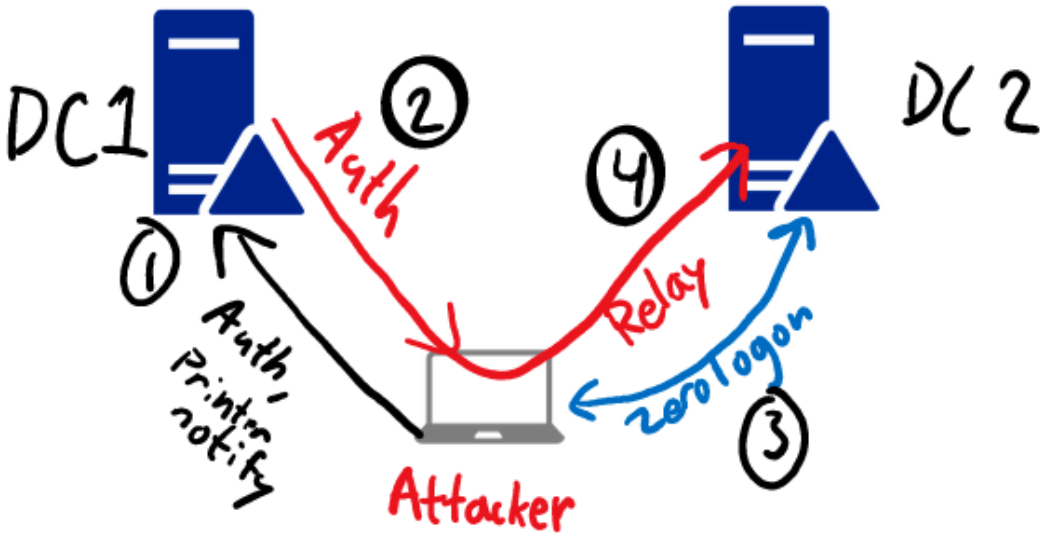
Créditos: T. Tervoort, “Zerologon:Unauthenticated domain controller compromise by subverting Netlogon cryptography (CVE-2020-1472)”, Secura, Septiembre 2020

3. Vulnerabilidad CVE-2020-1472

Impacto – BONUS track

- **Explotación del protocolo NTLM**
  - Requiere ciertas condiciones para la explotación
  - **Obtención de hashes de las contraseñas de los usuarios del dominio**





Créditos: Dirk-jan Mollema, https://dirkjanm.io/a-different-way-of-abusing-zerologon/

3. Vulnerabilidad CVE-2020-1472

Análisis mediante binary diffing

- **Binary diffing (o patch diffing)**
  - **Técnica de análisis de binarios mediante comparación entre código binario**
  - *Normalmente, se compara el fichero binario original con el fichero binario actualizado (después de aplicar el parche)*
  - Muy aplicado para detección de vulnerabilidades tras publicación de parches
    - Ejemplo: https://googleprojectzero.blogspot.com/2017/10/using-binary-diffing-to-discover.html

- **Herramientas**
  - IDA Pro + Diaphora (https://github.com/joxeankoret/diaphora)
  - IDA Pro + BinDiff (https://www.zynamics.com/bindiff.html)
  - Ghidra + BinExport + BinDiff
    - https://ghidra-sre.org/
    - https://github.com/google/binexport/tree/master/java/BinExport

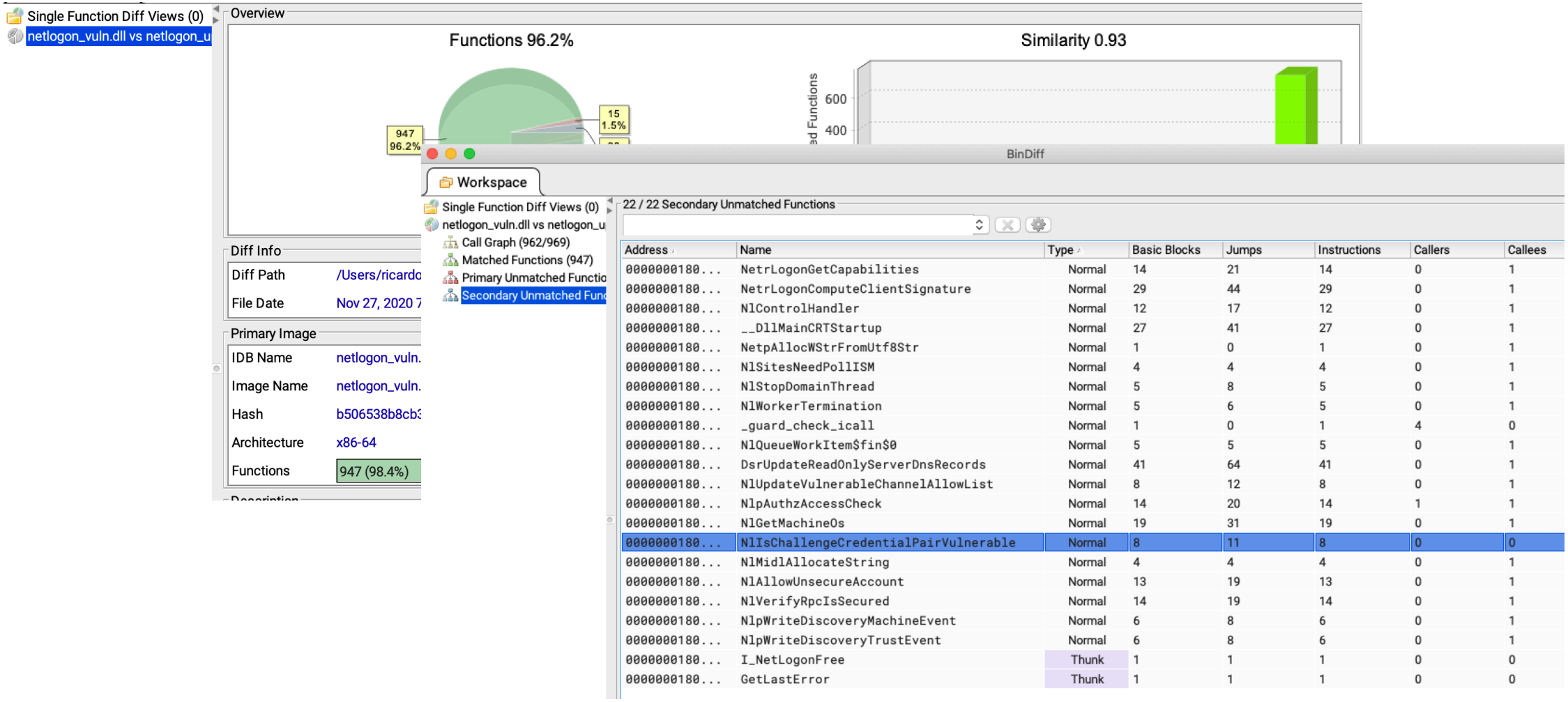
3. Vulnerabilidad CVE-2020-1472

Análisis mediante binary diffing

- **Instalación de herramientas**
  - Compila BinExport (con gradle) y listo
  - Añade la extensión de BinExport a Ghidra (*File -> Install Extensions …*)
- **Pasos para el análisis**
  - Obtener ficheros binarios (vulnerable y actualizado)
    - Suerte con las actualizaciones de Windows update… (déjale unos días)
  - Análisis con Ghidra
    - Abre los ficheros en Ghidra, descarga el PDB manualmente y analiza el fichero con la información del PDB descargada
    - Guarda la BDD generada
  - Exportación con BinExport desde Ghidra
  - Crea el diff de los ficheros exportados
    - Vía comando, bindiff file1.BinExport file2.BinExport
    - Vía interfaz de BinDiff
  - **A jugar!**

3. Vulnerabilidad CVE-2020-1472

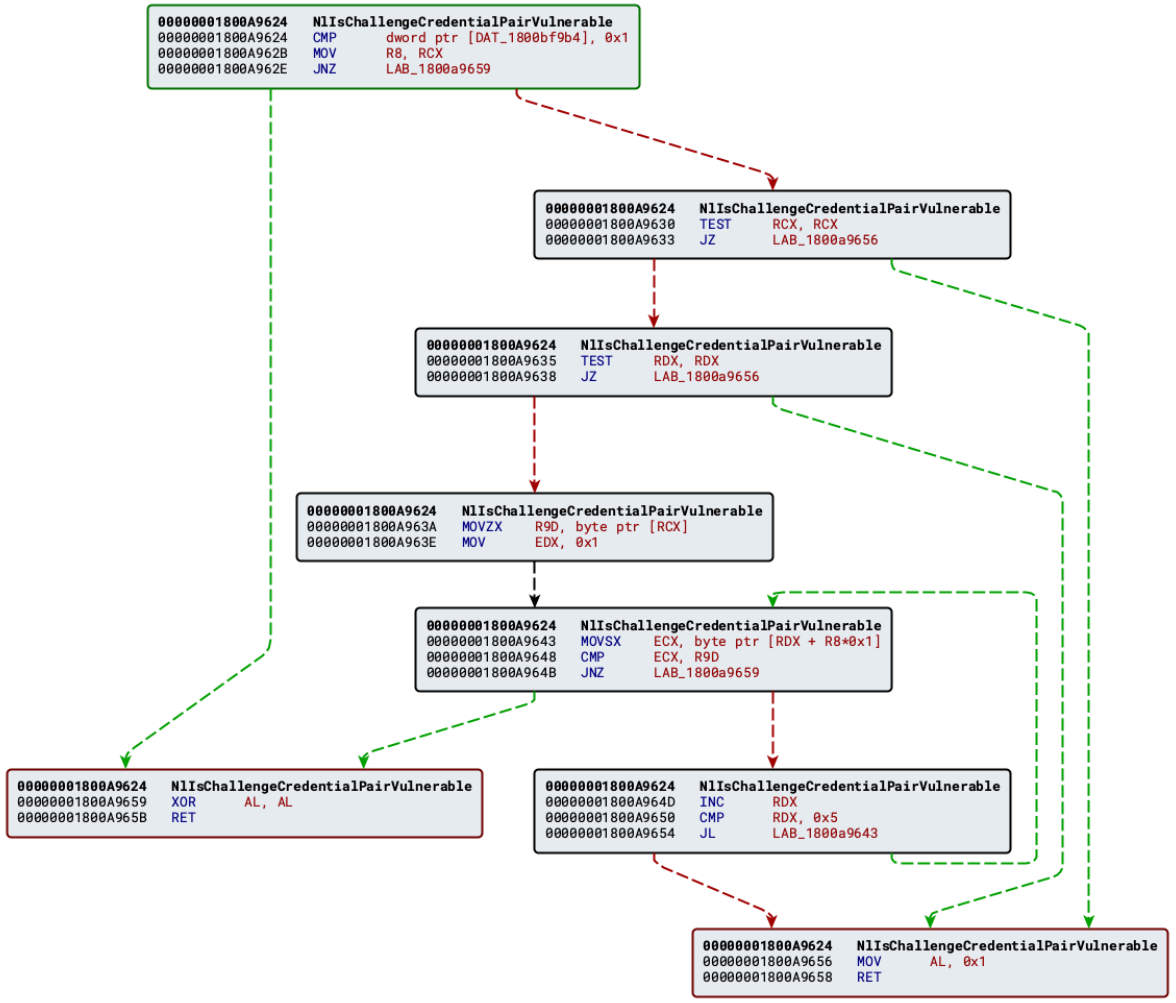
Análisis mediante binary diffing



Address	Name	Type	Basic Blocks	Jumps	Instructions	Callers	Callees
0000000180...	NetrLogonGetCapabilities	Normal	14	21	14	0	1
0000000180...	NetrLogonComputeClientSignature	Normal	29	44	29	0	1
0000000180...	NlControlHandler	Normal	12	17	12	0	1
0000000180...	__DllMainCRTStartup	Normal	27	41	27	0	1
0000000180...	NetpAllocWStrFromUtf8Str	Normal	1	0	1	0	1
0000000180...	NlSitesNeedPollISM	Normal	4	4	4	0	1
0000000180...	NlStopDomainThread	Normal	5	8	5	0	1
0000000180...	NlWorkerTermination	Normal	5	6	5	0	1
0000000180...	_guard_check_icall	Normal	1	0	1	4	0
0000000180...	NlQueueWorkItemfin0	Normal	5	5	5	0	1
0000000180...	DsrUpdateReadOnlyServerDnsRecords	Normal	41	64	41	0	1
0000000180...	NlUpdateVulnerableChannelAllowList	Normal	8	12	8	0	1
0000000180...	NlpAuthzAccessCheck	Normal	14	20	14	1	1
0000000180...	NlGetMachineOs	Normal	19	31	19	0	1
0000000180...	NlIsChallengeCredentialPairVulnerable	Normal	8	11	8	0	0
0000000180...	NlMidlAllocateString	Normal	4	4	4	0	1
0000000180...	NlAllowUnsecureAccount	Normal	13	19	13	0	1
0000000180...	NlVerifyRpcIsSecured	Normal	14	19	14	0	1
0000000180...	NlpWriteDiscoveryMachineEvent	Normal	6	8	6	0	1
0000000180...	NlpWriteDiscoveryTrustEvent	Normal	6	8	6	0	1
0000000180...	I_NetLogonFree	Thunk	1	1	1	0	0
0000000180...	GetLastError	Thunk	1	1	1	0	0

3. Vulnerabilidad CVE-2020-1472

Análisis mediante binary diffing



Decompile: NlIsChallengeCredentialPairVulnerable - (netlogon_updated.dll)

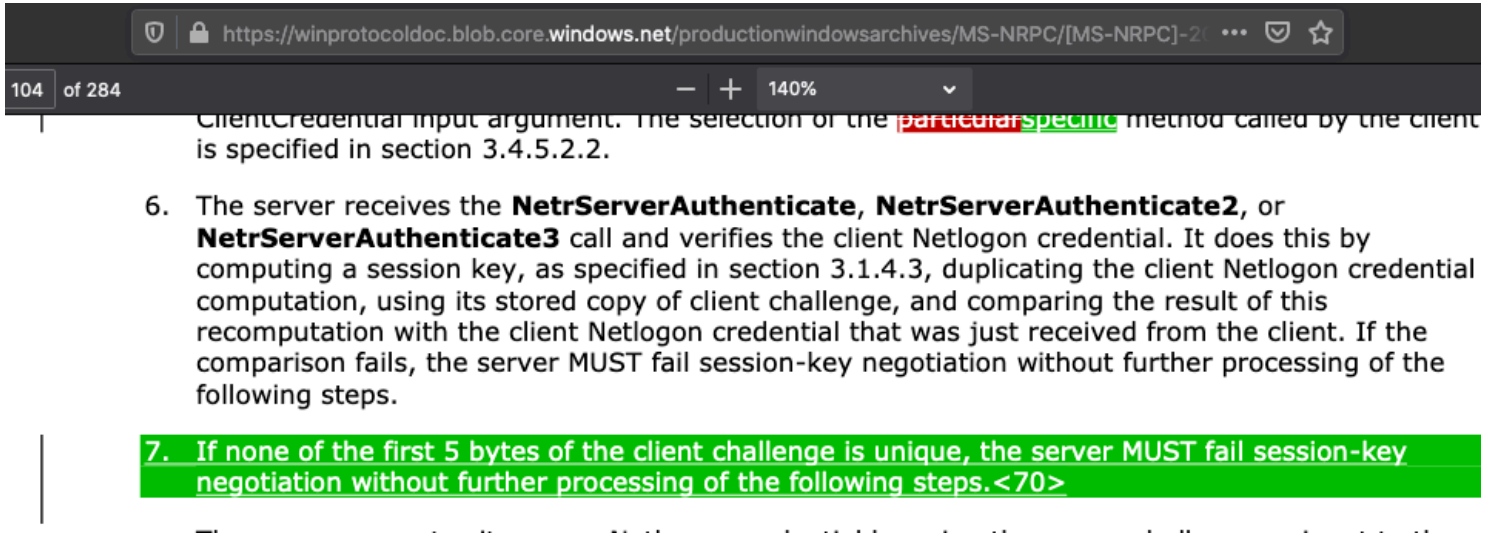
```
1
2  /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
3
4  undefined NlIsChallengeCredentialPairVulnerable(byte *param_1,longlong param_2)
5
6  {
7    longlong lVar1;
8
9    if (_DAT_1800bf9b4 != 1) {
10     return 0;
11   }
12   if ((param_1 != (byte *)0x0) && (param_2 != 0)) {
13     lVar1 = 1;
14     do {
15       if ((int)(char)param_1[lVar1] != (uint)*param_1) {
16         return 0;
17       }
18       lVar1 = lVar1 + 1;
19     } while (lVar1 < 5);
20   }
21   return 1;
22 }
23
```

- **Como parámetro recibe el challenge del cliente**
- Comprueba los 5 primeros bytes del challenge, comparando del 2 al 5 con el primero
  - Devuelve 0 (no vulnerable) si hay al menos dos bytes diferentes
  - Si son todos iguales, devuelve 1 (vulnerable)
- ¿Recuerdas lo que vimos antes? S
  - si IV tiene todo ceros, $\exists x \in \mathbb{N}, 0 \leq x \leq 255 : P_i = x \therefore C_i = 0, 1 \leq i \leq N$, donde x está distribuido uniformemente de manera aleatoria

3. Vulnerabilidad CVE-2020-1472

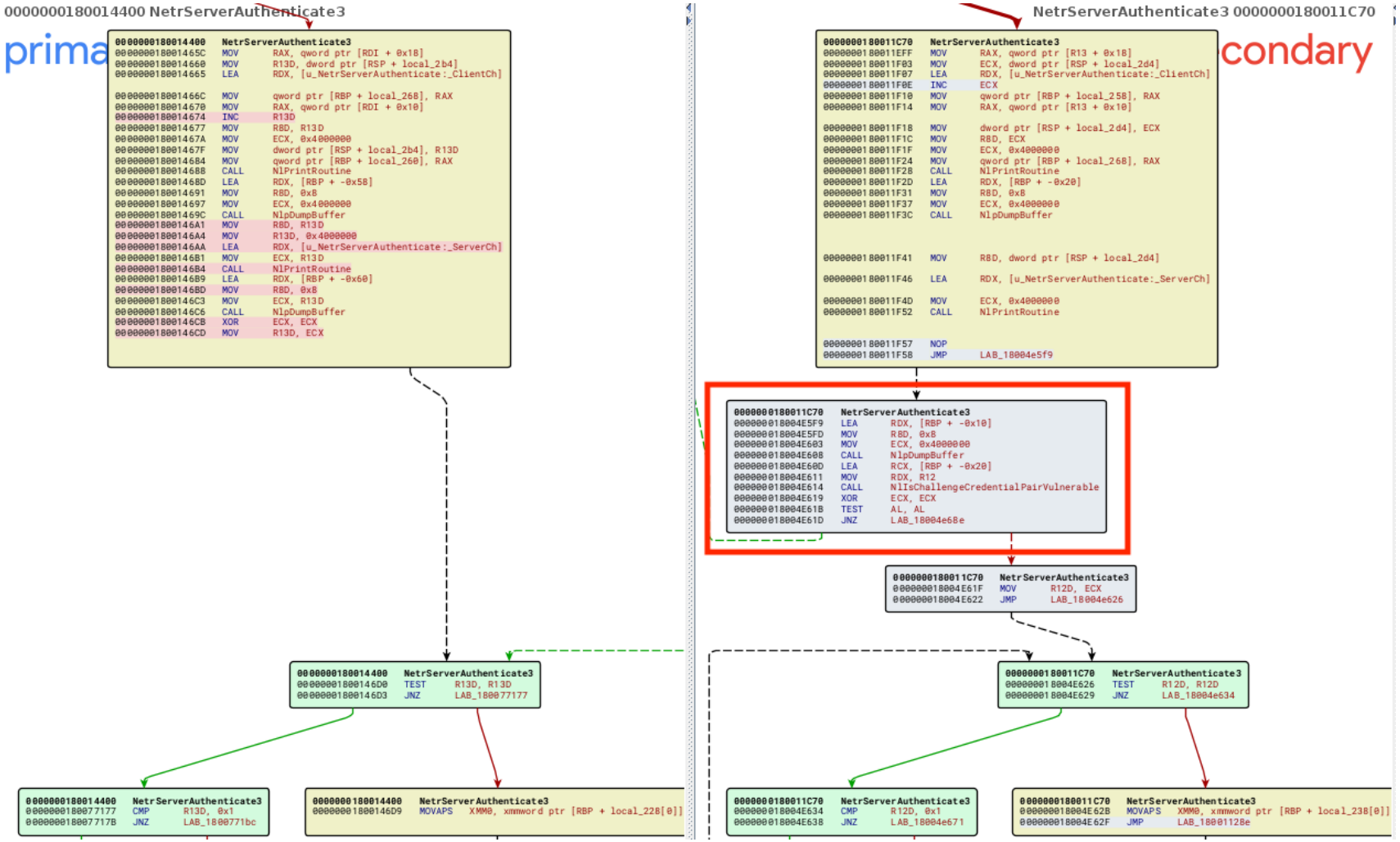
Análisis mediante binary diffing

- **Diff de la actualización del protocolo Netlogon:**
  - https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-NRPC/%5bMS-NRPC%5d-200826-diff.pdf



3. Vulnerabilidad CVE-2020-1472

Análisis mediante binary diffing



4. Soluciones

- **Actualizar Windows:**
  - https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472
  - *Solucionan el problema de challenges del cliente que puedan debilitar AES-CFB8,* ***pero no han cambiado la inicialización con IV de ceros***
  - **Tutorial de Microsoft acerca de los cambios necesarios a aplicar (para administradores)**
    - https://support.microsoft.com/en-us/help/4557222/how-to-manage-the-changes-in-netlogon-secure-channel-connections-assoc
    - Modo DC Enforcement: activarlo vía Registro (hasta febrero 2021 no estará disponible vía parche)
      - RECOMENDADO activarlo ahora de este modo, hasta que esté el parche
- **Métodos para detección de explotación:**
  - https://www.kroll.com/en/insights/publications/cyber/cve-2020-1472-zerologon-exploit-detection-cheat-sheet
  - https://yelata.medium.com/zerologon-cve-2020-1472-turning-microsofts-patch-to-a-snort-rule-and-a-little-extra-534bbaf4fc45
  - https://blog.zsec.uk/zerologon-attacking-defending/#how-do-we-fix-it

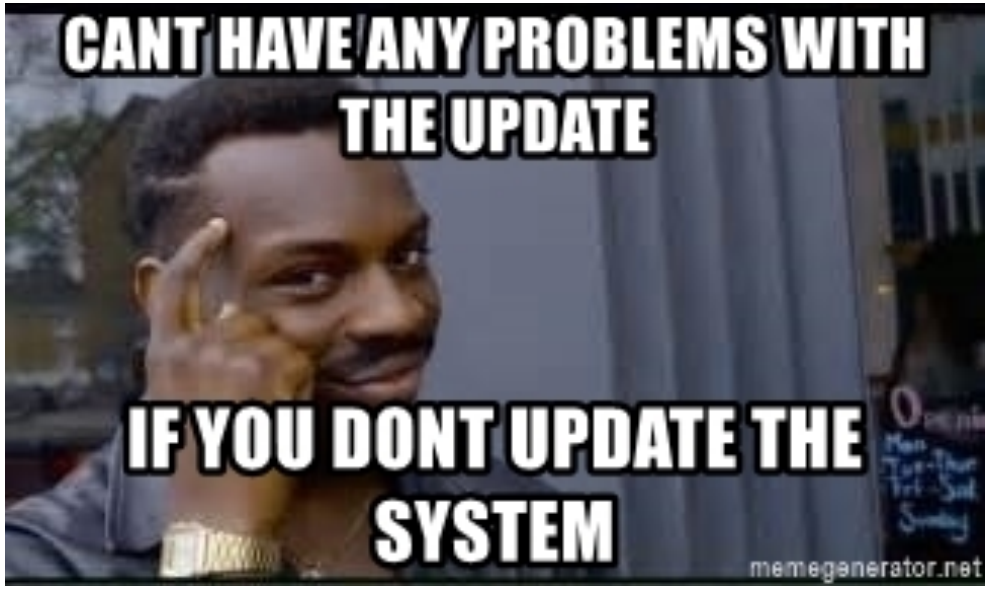
	Artifacts Able to Detect Exploit:			
Common Zerologon Exploits	Windows event logs	Dump of hash history	LSASS (Yara rules)	Snort / Suricata
DC password reset with original password reestablished	✓	✓	✓	✓
DC password reset with original password reestablished	✓	✓	✓	✓
Spool service (printer bug) + NTLM Relay without password reset	✓		✓	✓

5. Conclusiones

Take-home messages

- **Evita implementar vuestro propio algoritmo criptográfico**
  - Si lo haces, libéralo. Deja que la comunidad encuentre lo que tú no ves
- **Estudia bien los algoritmos criptográficos que vas a usar**
  - Los modos de AES son un mundo, y ciertas combinaciones pueden ser peligrosas
- **Usa estándares: cuanto más contrastados, mejor**
  - Menor probabilidad de “fallos inesperados”

ACTUALIZA SIEMPRE TU SISTEMA OPERATIVO

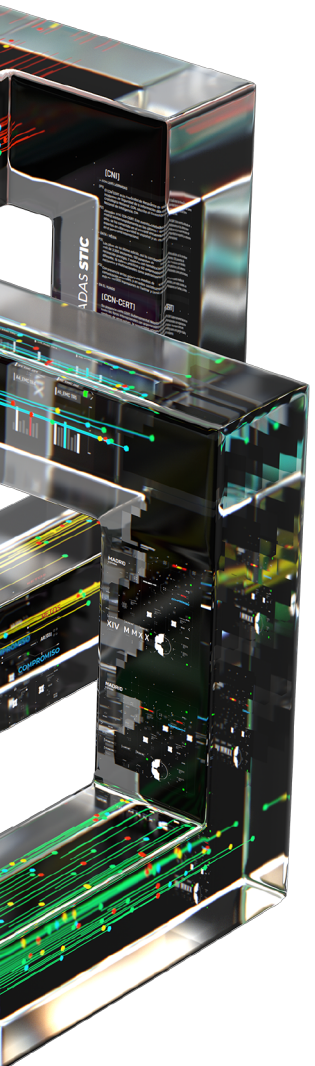


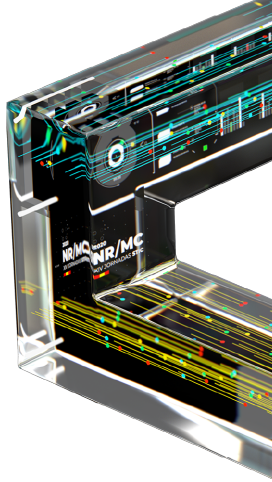
Algunas referencias de interés

- https://www.secura.com/pathtoimg.php?id=2055
- https://github.com/SecuraBV/CVE-2020-1472 (prueba de concepto original)

- https://www.kb.cert.org/vuls/id/490028
- https://nvd.nist.gov/vuln/detail/CVE-2020-1472
- https://media.cert.europa.eu/static/SecurityAdvisories/2020/CERT-EU-SA2020-046.pdf

- https://packetstormsecurity.com/files/159190/Zerologon-Proof-Of-Concept.html
- https://packetstormsecurity.com/files/160127/Zerologon-Netlogon-Privilege-Escalation.html
- https://blog.0patch.com/2020/09/micropatch-for-zerologon-perfect.html
- https://nakedsecurity.sophos.com/2020/09/17/zerologon-hacking-windows-servers-with-a-bunch-of-zeros/





MUCHAS GRACIAS

#XIVJORNADASCCNCERT