black hat EUROPE 2022

DECEMBER 7-8, 2022

BRIEFINGS

Breaking Kerberos' RC4 Cipher and Spoofing Windows PACs

Tom Tervoort

Introduction

Secura

BUREAU VERITAS 1828

BUREAU
VERITAS

Outline

1. Introduction to Kerberos
2. The problem(s) with RC4-HMAC
3. PAC signatures
4. Spoofing a PAC signature
5. Spoofing with AES encryption types
6. We spoofed a PAC. Now what?
7. Follow-up

What is Kerberos?

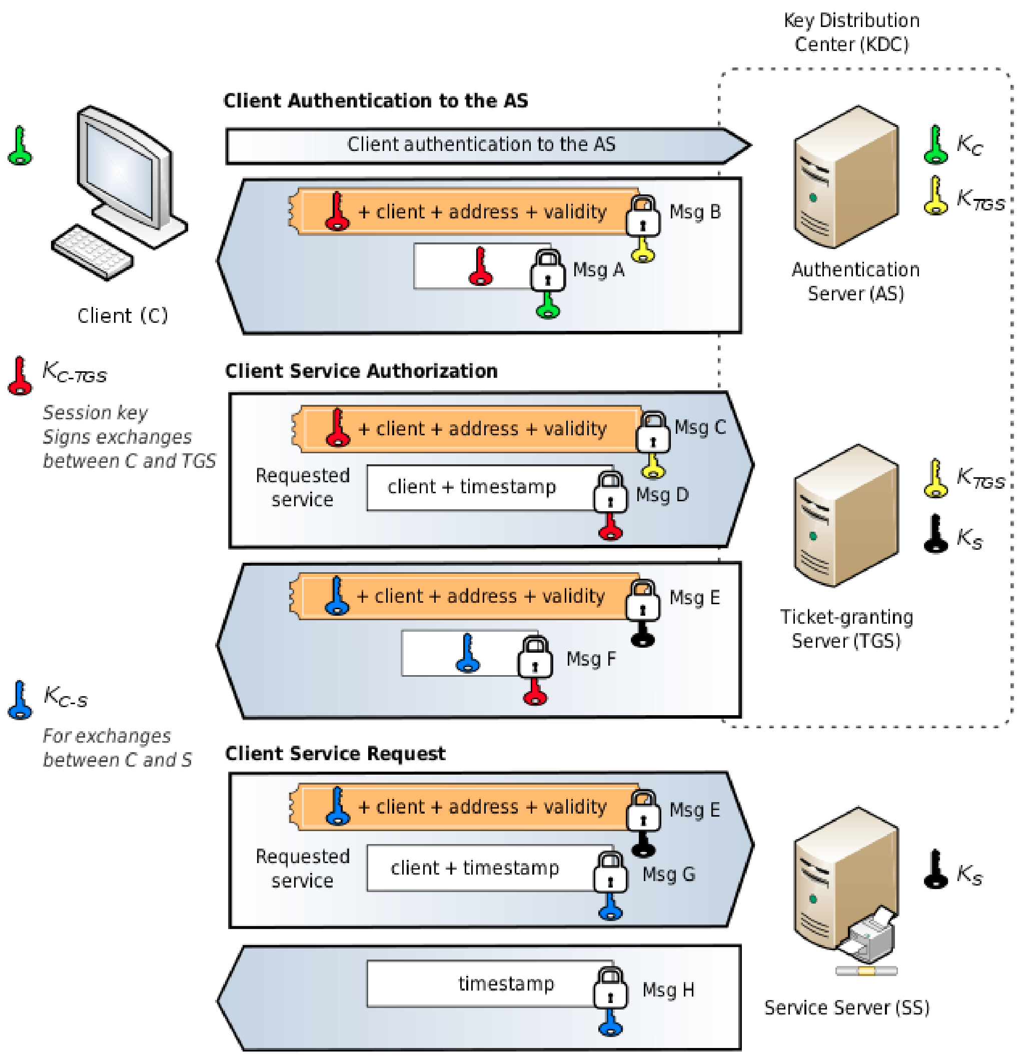


Image source: Jeran Renz, Wikimedia Commons

The problem(s) with RC4-HMAC

Default supported encryption types

RC4_HMAC_MD5	Rivest Cipher 4 with Hashed Message Authentication Code using the Message-Digest algorithm 5 checksum function Supported in Windows 2000 Server, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012 and Windows Server 2012 R2.
AES128_HMAC_SHA1	Advanced Encryption Standard in 128-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.
AES256_HMAC_SHA1	Advanced Encryption Standard in 256-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Default supported encryption types

Statistical biases!

RC4_HMAC_MD5	Rivest Cipher 4 with Hashed Message Authentication Code using the Message-Digest algorithm 5 checksum function Supported in Windows 2000 Server, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012 and Windows Server 2012 R2.
AES128_HMAC_SHA1	Advanced Encryption Standard in 128-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.
AES256_HMAC_SHA1	Advanced Encryption Standard in 256-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Default supported encryption types

Statistical biases!

Collisions!

RC4_HMAC_MD5	Rivest Cipher 4 with Hashed Message Authentication Code using the Message-Digest algorithm 5 checksum function Supported in Windows 2000 Server, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012 and Windows Server 2012 R2.
AES128_HMAC_SHA1	Advanced Encryption Standard in 128-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.
AES256_HMAC_SHA1	Advanced Encryption Standard in 256-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Default supported encryption types

Statistical biases!

Collisions!

Fine, actually

RC4_HMAC_MD5	Rivest Cipher 4 with Hashed Message Authentication Code using the Message-Digest algorithm 5 checksum function Supported in Windows 2000 Server, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012 and Windows Server 2012 R2.
AES128_HMAC_SHA1	Advanced Encryption Standard in 128-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.
AES256_HMAC_SHA1	Advanced Encryption Standard in 256-bit cipher block with Hashed Message Authentication Code using the Secure Hash Algorithm (1). Not supported in Windows 2000 Server, Windows XP, or Windows Server 2003. Supported in Windows Vista, Windows Server 2008, Windows 7, Windows 10, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Known weaknesses

- NTLM compatibility => Overpass-the-hash attack
- Password-based key derivation without salting/stretching => easier Kerberoasting
- AES-to-RC4 ticket downgrading (fixed in Server 2019)
- Recent attack by James Forshaw: downgrade to breakable legacy variant

CIS Benchmarks

2.3.11.4 (L1) Ensure 'Network security: Configure encryption types allowed for Kerberos' is set to 'AES128_HMAC_SHA1, AES256_HMAC_SHA1, Future encryption types' (Automated)

```
                                              BEST CURRENT PRACTICE
Internet Engineering Task Force (IETF)                     B. Kaduk
Request for Comments: 8429                                   Akamai
BCP: 218                                                   M. Short
Updates: 3961, 4120                           Microsoft Corporation
Category: Best Current Practice                        October 2018
ISSN: 2070-1721
```

Deprecate Triple-DES (3DES) and RC4 in Kerberos

A look at RFC 4757

Schemes for use within Kerberos protocols	
CHKSUM	Message Authentication Code (MAC)
ENCRYPT	Authenticated Encryption

GSS-API functions (used by LDAP, RPC etc.)	
GetMIC	Message Authentication Code (MAC)
GSS_Wrap	Authenticated Encryption

This may be a problem...

```
The function is defined as follows:

K = the Key
T = the message type, encoded as a little-endian four-byte integer

CHKSUM(K, T, data)

     Ksign = HMAC(K, "signaturekey")  //includes zero octet at end
     tmp = MD5(concat(T, data))
     CHKSUM = HMAC(Ksign, tmp)
```

This may be a problem...

```
The function is defined as follows:

K = the Key
T = the message type, encoded as a little-endian four-byte integer

CHKSUM(K, T, data)

     Ksign = HMAC(K, "signaturekey")  //includes zero octet at end
     tmp = MD5(concat(T, data))
     CHKSUM = HMAC(Ksign, tmp)
```

MD5 hash collision => CHKSUM MAC collision!

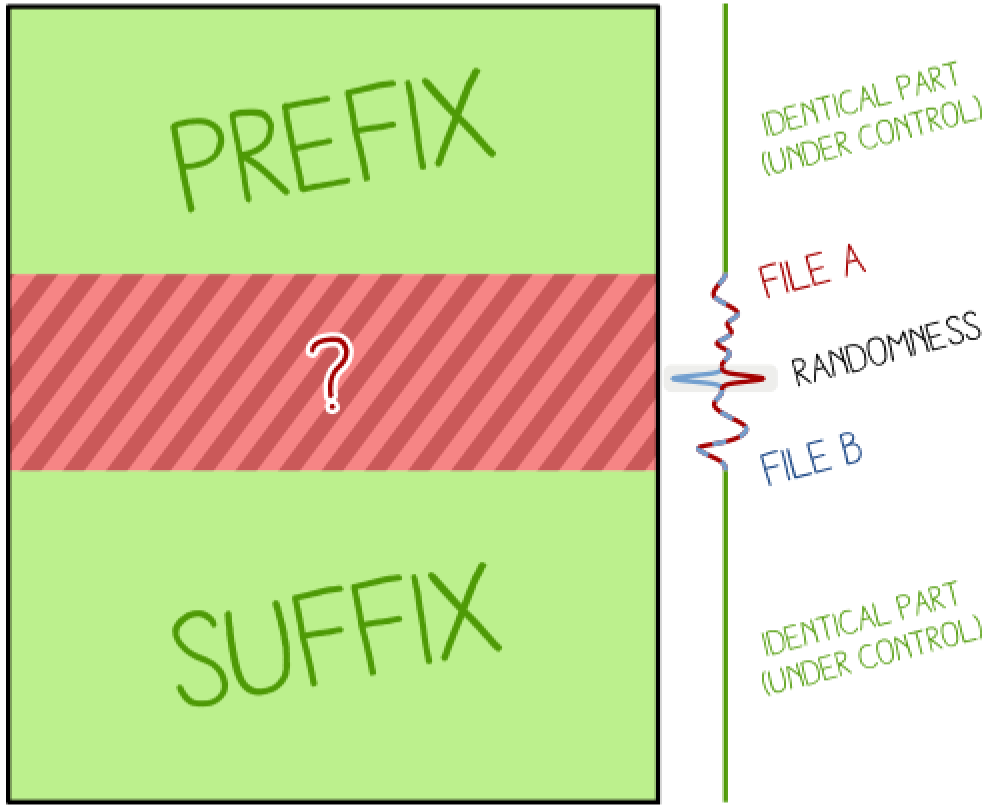
Vulnerable schemes (in theory)

Schemes for use within Kerberos protocols	
CHKSUM	✗ MD5-then-HMAC: forgeable
ENCRYPT	✓ Direct HMAC + encrypt: not vulnerable

GSS-API functions (used by LDAP, RPC etc.)	
GetMIC	✗ MD5-then-HMAC: forgeable
GSS_Wrap	✗ MD5-then-HMAC + encrypt: CCA vulnerability

Computing MD5 collisions

Identical prefix (cheap)



Chosen-prefix (expensive)

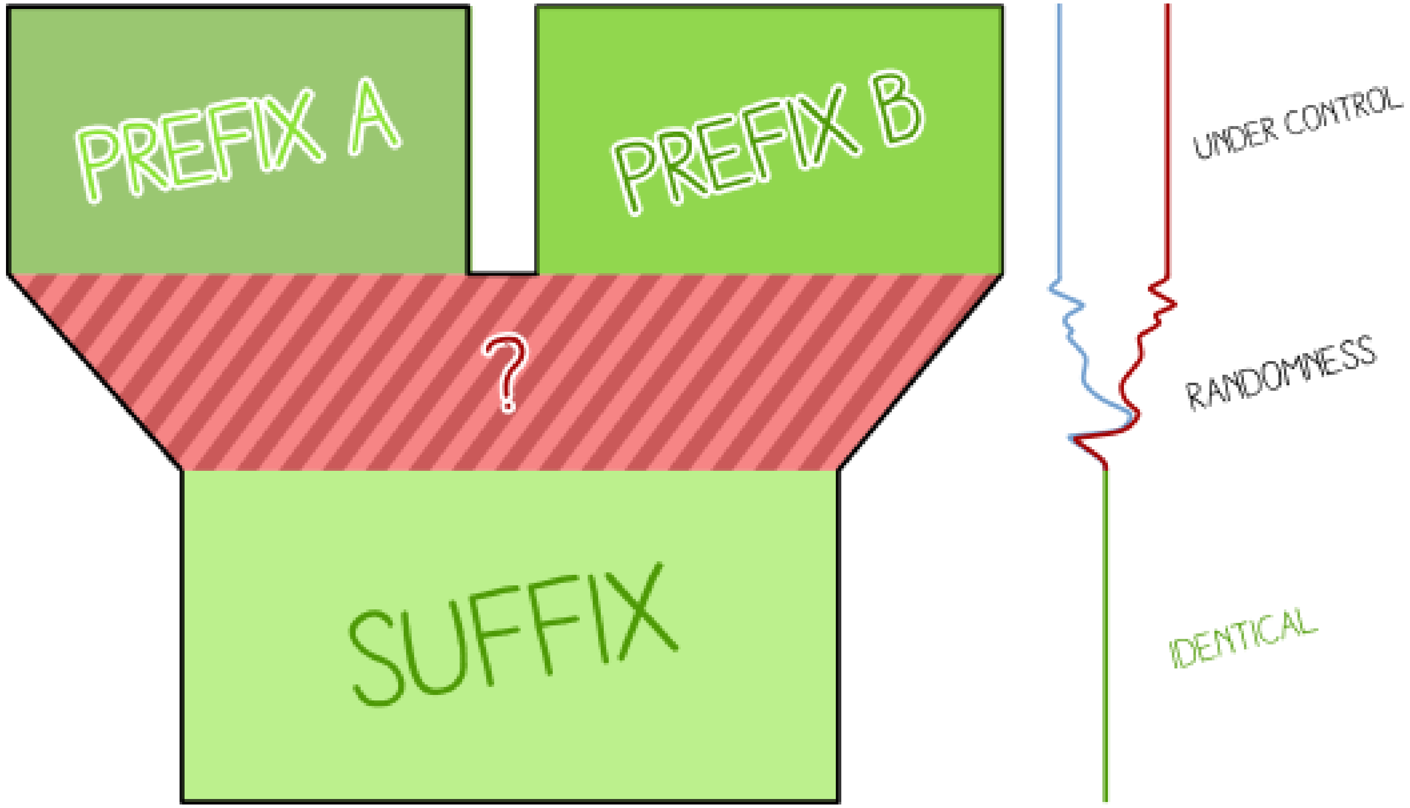
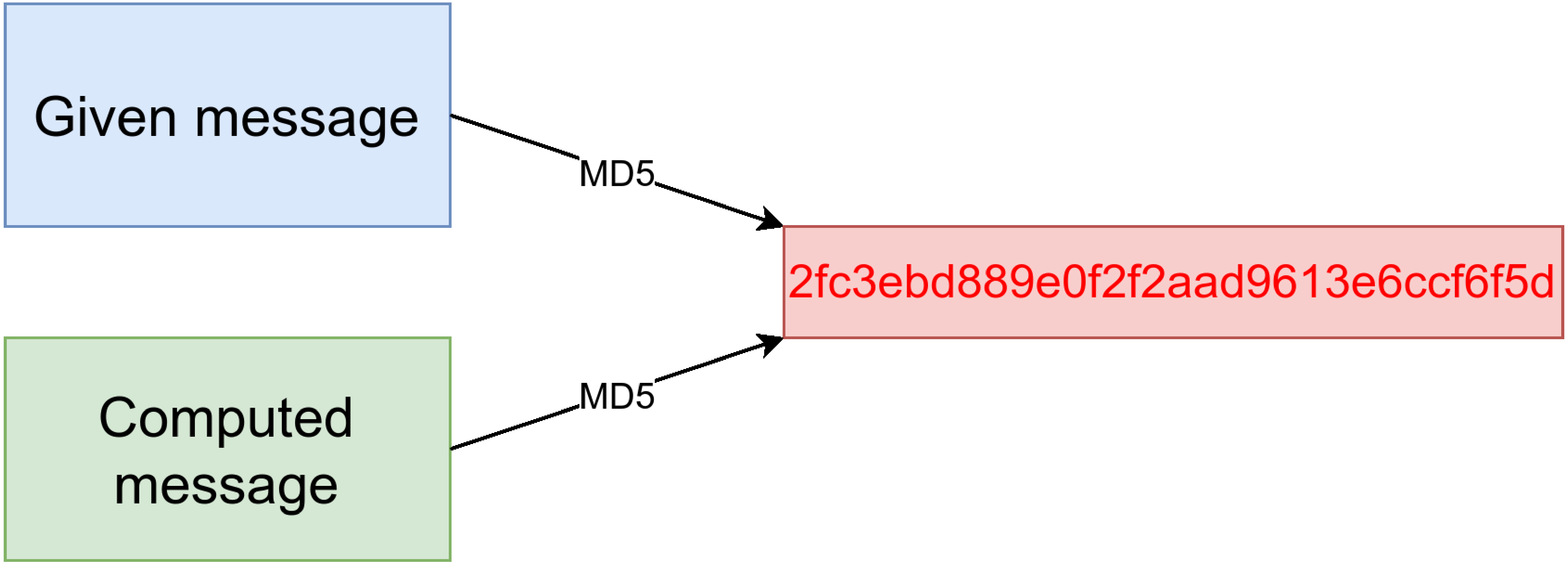


Image source: Ange Albertini (https://github.com/corkami/collisions)

Can’t do this, though…

Pre-image attack



How to exploit?

1. Prepare messages M1 and M2 with same MD5 hash (different prefixes, followed by **collision blocks**, followed by identical suffix)
2. **Get server to produce M1 message,** authenticated with some unknown key
3. Stick authentication tag from M1 on M2
4. **Achieve some attack goal** with spoofed M2

Challenges:

- Find producer of predictable M1, that attacker can query
- Find way to slot a bunch of collision blocks in M1
- Find a security boundary M2 would break, that an attacker couldn’t bypass otherwise

PAC signatures

A protocol using the broken CHKSUM

[MS-PAC]: Privilege Attribute Certificate Data Structure

Article • 04/27/2022 • 4 minutes to read

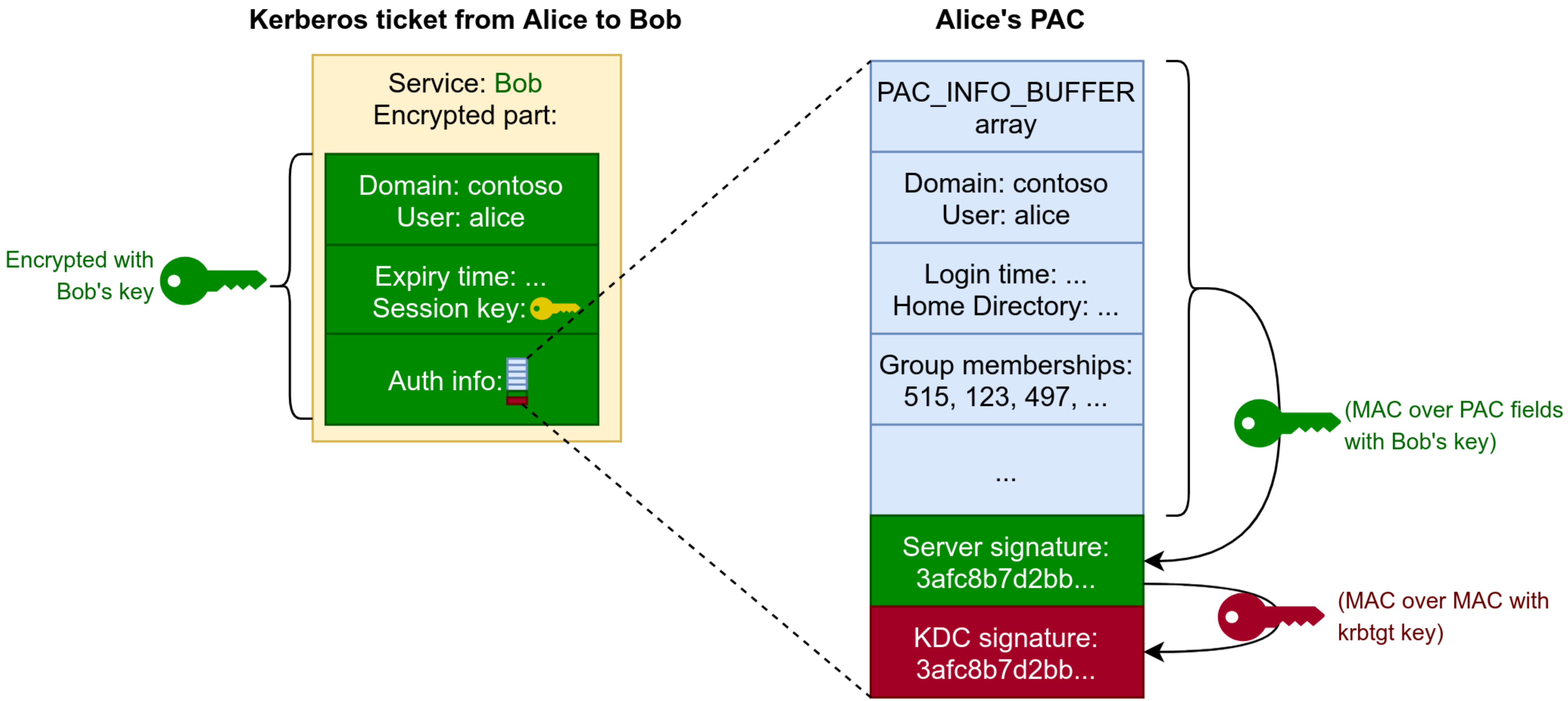
Feedback

Specifies the Privilege Attribute Certificate Data Structure, which is used to encode authorization information. The Privilege Attribute Certificate also contains memberships, additional credential information, profile and policy information, and supporting security metadata.

SignatureType (4 bytes): A 32-bit unsigned integer value in little-endian format that defines the cryptographic system used to calculate the checksum. This MUST be one of the values defined in the following table. The corresponding sizes of the signatures are also given. The key used with the cryptographic system corresponds to the value of the **ulType** field of the outer PAC_INFO_BUFFER (section 2.4) structure. The value 0x00000006 specifies the server's key, and the value 0x00000007 specifies the KDC's key.

Value	Meaning
KERB_CHECKSUM_HMAC_MD5 0xFFFFFF76	As specified in [RFC4120] and [RFC4757] section 4. Signature size is 16 bytes. Decimal value is -138.

PAC authorization data



This may be worth attacking...

PAC validation is a security feature that addresses PAC spoofing, preventing an attacker from gaining unauthorized access to a system or its resources by using a tampered PAC. This is critical in applications where user impersonation is utilized. PAC validation ensures the user presents exact authorization data as it was granted in the Kerberos ticket.

One key reason why a PAC should be verified as unaltered is to ensure that no additional privileges have been maliciously added to - or removed from - the ticket. This makes the PAC validation an important consideration when designing applications that impersonate users and access sensitive data, or applications that grant administrative privileges.

Source: Microsoft Open Specifications Support Team Blog

black hat EUROPE 2022

Spoofing a PAC signature

Getting collision bytes into a signed PAC

```
PS C:\Users\Administrator> New-MachineAccount -MachineAccount pacpoc-computer
Enter a password for the new machine account: *****************************
[+] Machine account pacpoc-computer added
PS C:\Users\Administrator> Set-MachineAccountAttribute -MachineAccount pacpoc-computer -Attribute scriptPath -Value "xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx"
[+] Machine account pacpoc-computer attribute scriptPath updated
```

- By default, domain users can join up to 10 computer accounts to the AD.
- Gives control of scriptPath/LogonScript property.
- Max string length more than enough!
- However, the value must be valid Unicode (UTF-16 encoded in PAC).

Are random bytes valid UTF-16?

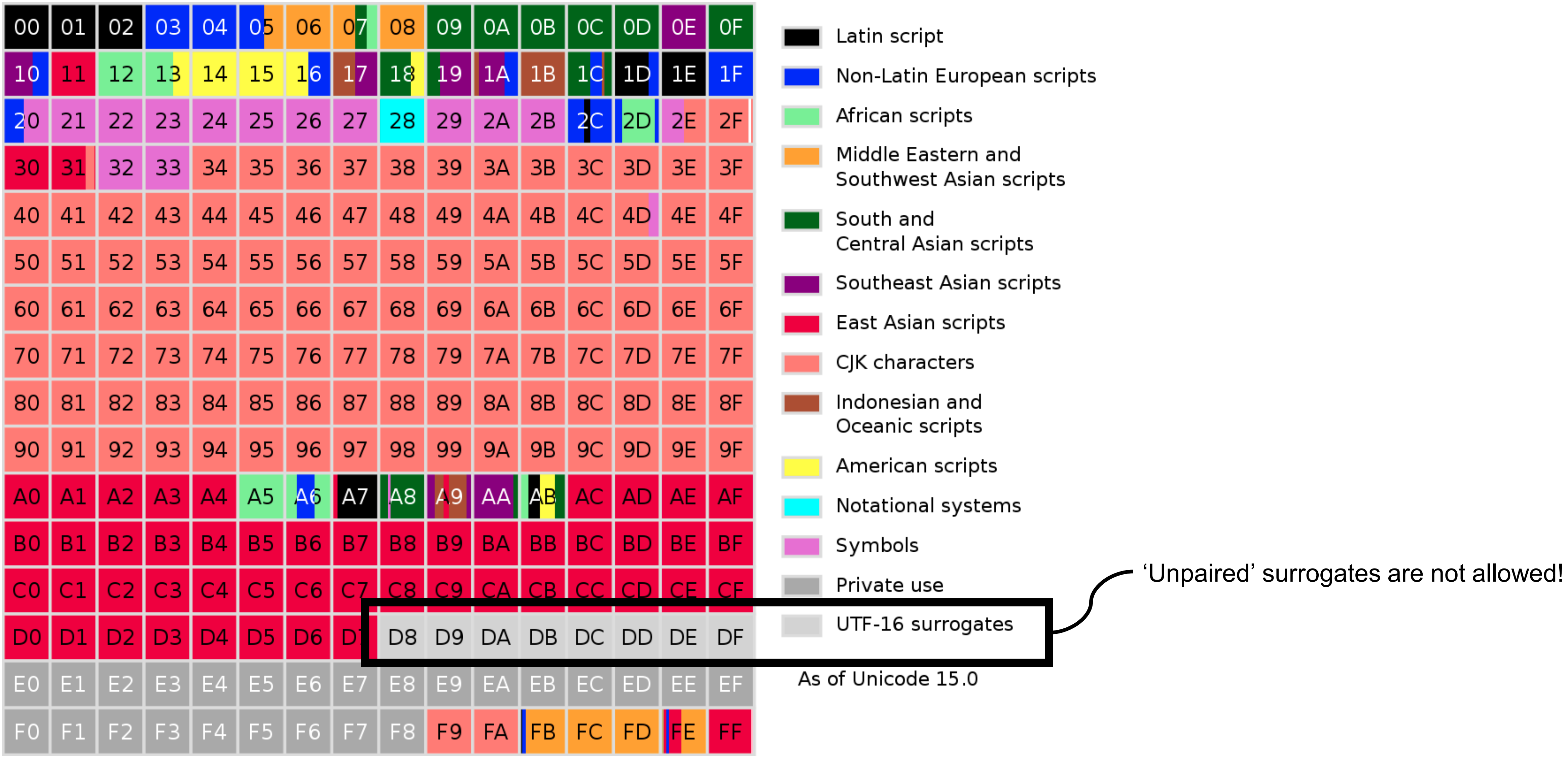


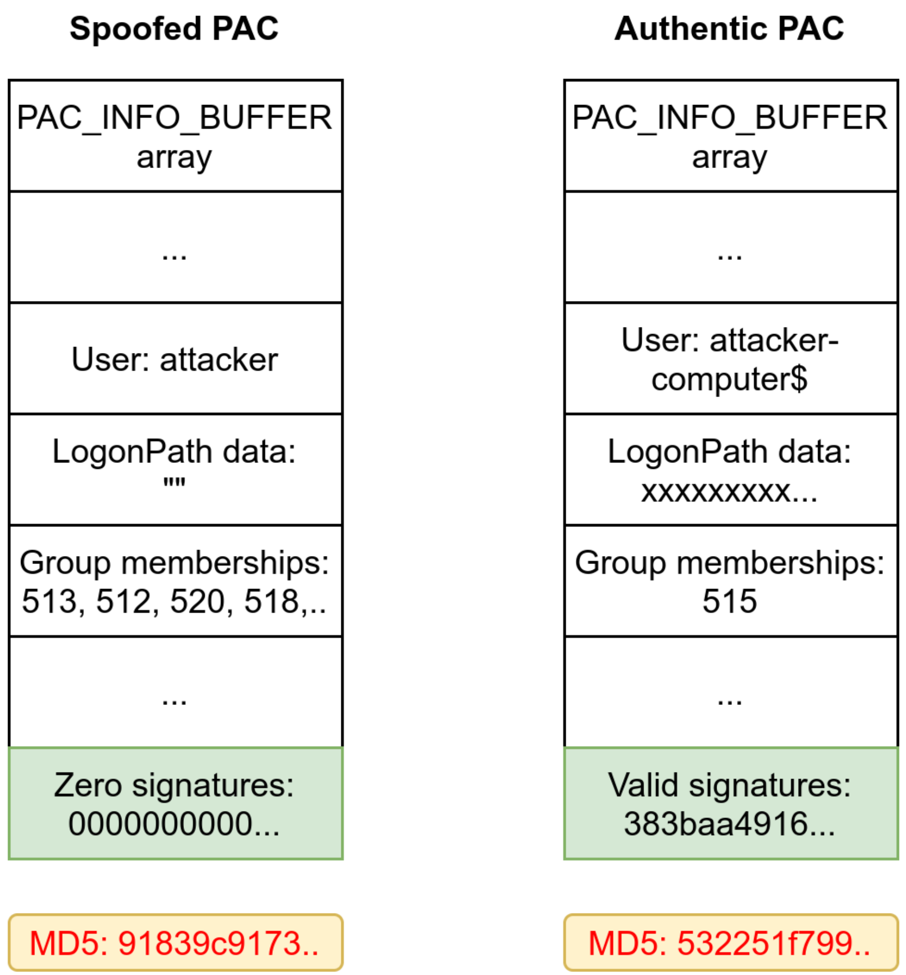
Image source: Drmccreedy, Wikimedia Commons

A small HashClash hack

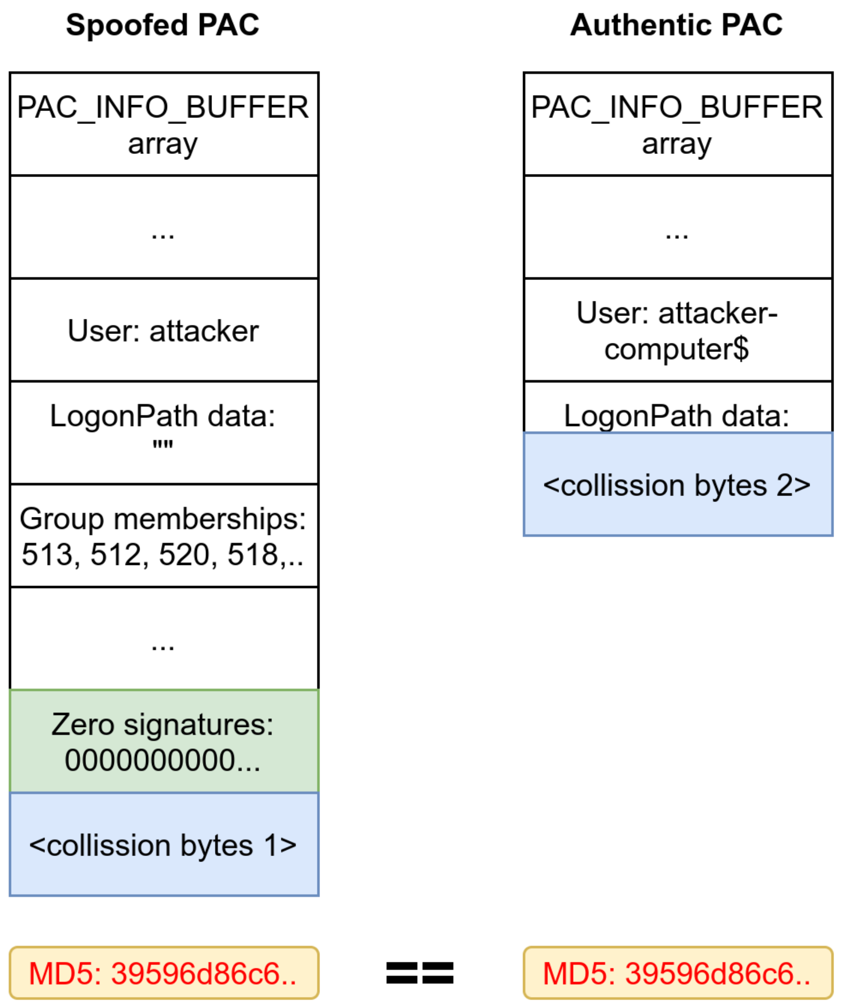
```
    # Start the computation
    rm -f step$k.log
    (dostepk $k 2>&1 &) | tee step$k.log
    kill $autokillerpid &>/dev/null
    killall -r md5_ &>/dev/null

    # Check for forbidden bit pattern (i.e. invalid UTF-16)
    illegal=`cat workdir$k/coll2* | xxd -p -c 2 | grep -q 'd[89a-f]$'; echo $?`

    # Check if the termination was completed or killed
    if [[ -f workdir$k/killed ]]; then
        failedk=$k
        let k=$((k > 1 ? k-1 : 0))
        notify "Step $failedk failed. Backtracking to step $k"
        let backtracks=backtracks+1
    elif [[ $illegal -eq 0 ]]; then
        notify "Illegal bit pattern. Retrying step $k"
    else
        notify "Step $k completed"
        let k=k+1
    fi
    sleep 2
done
```

- 2 in 3 chance an intermediate 64-byte block contains UTF-16 surrogate.
- Reject those; takes thrice as long but results in a **UTF-16 compatible collision**.

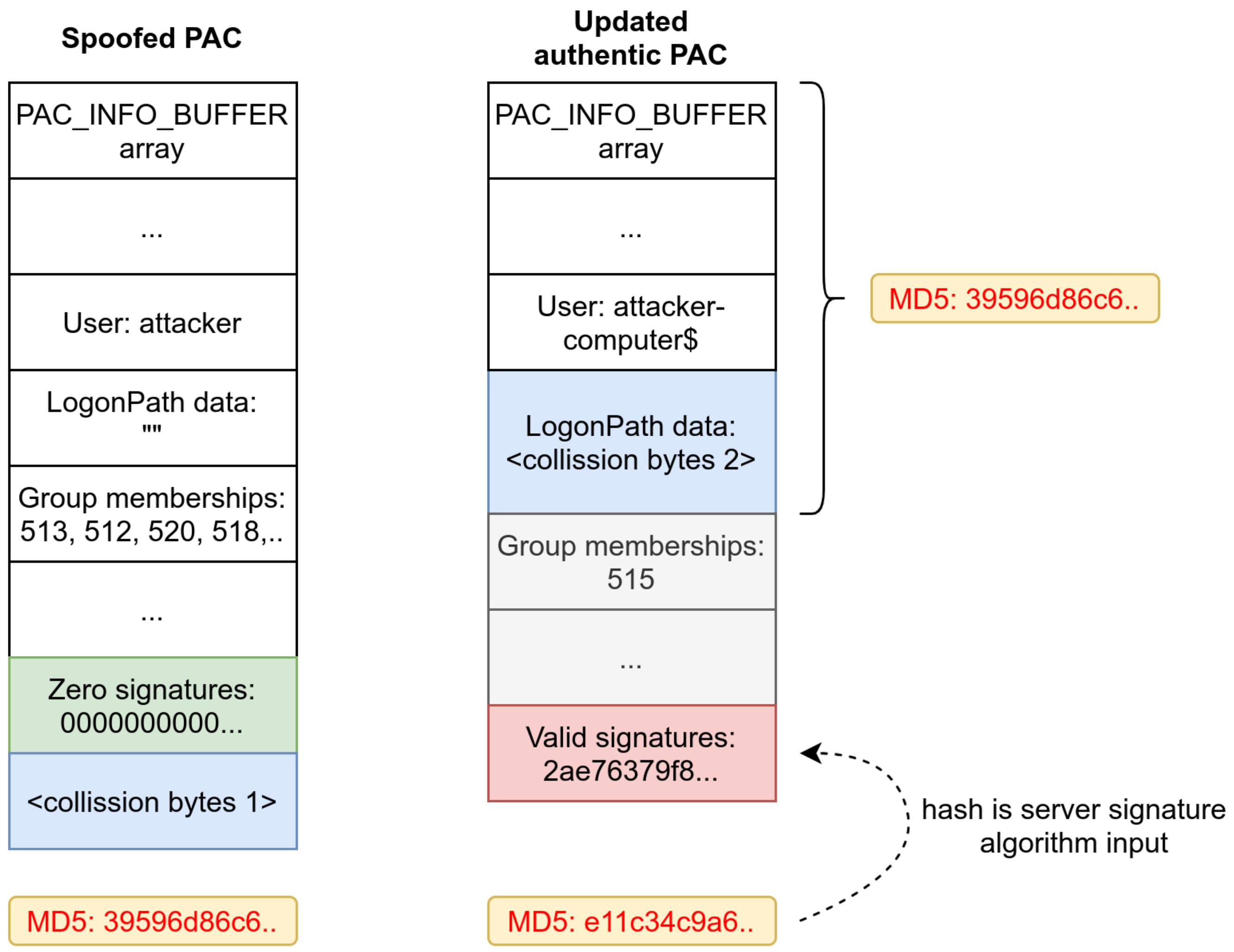
Attack step 1: prepare fake PAC and request real PAC



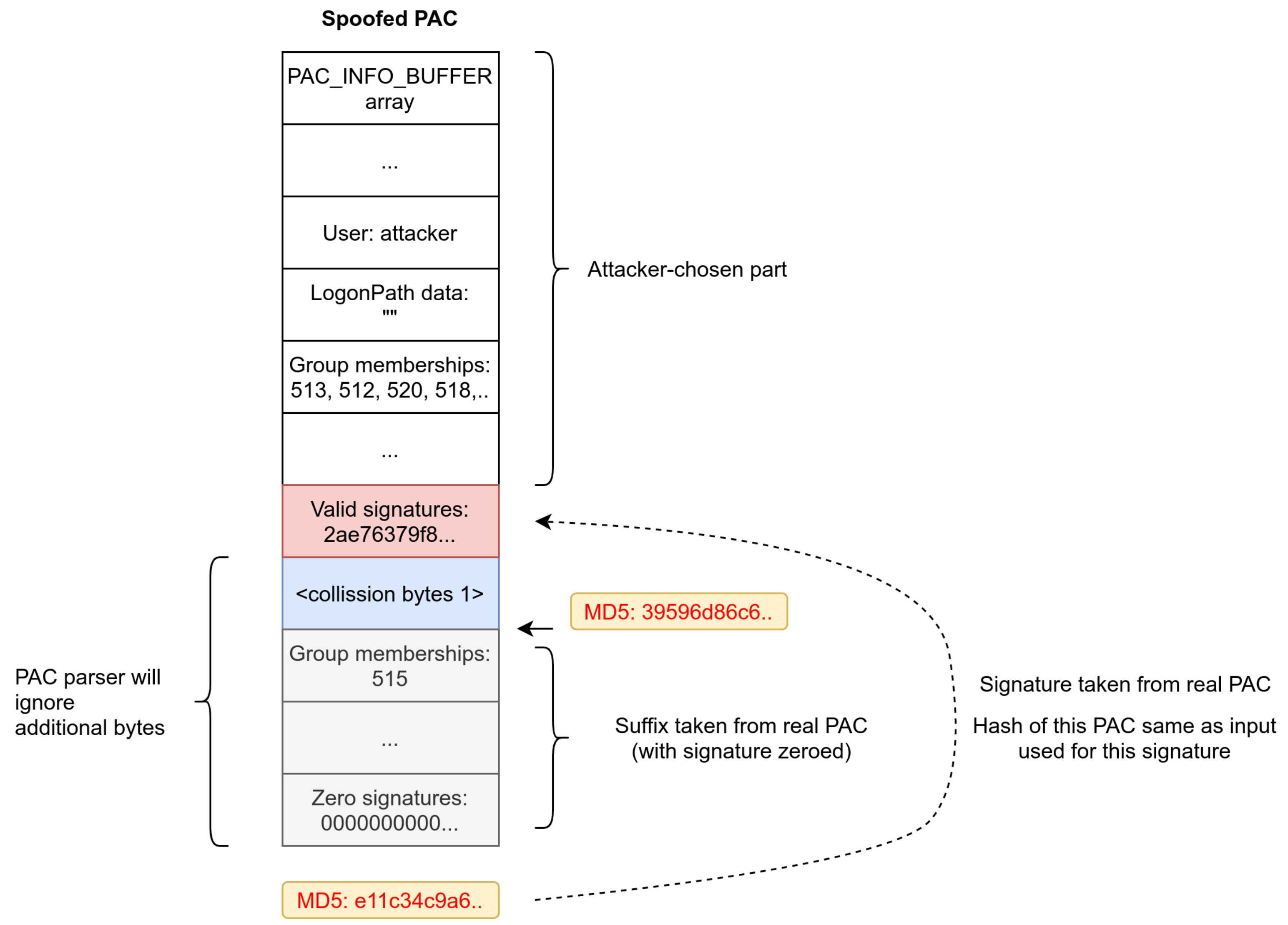
Step 2: compute collision



Step 3: store collision bytes in scriptPath request PAC again

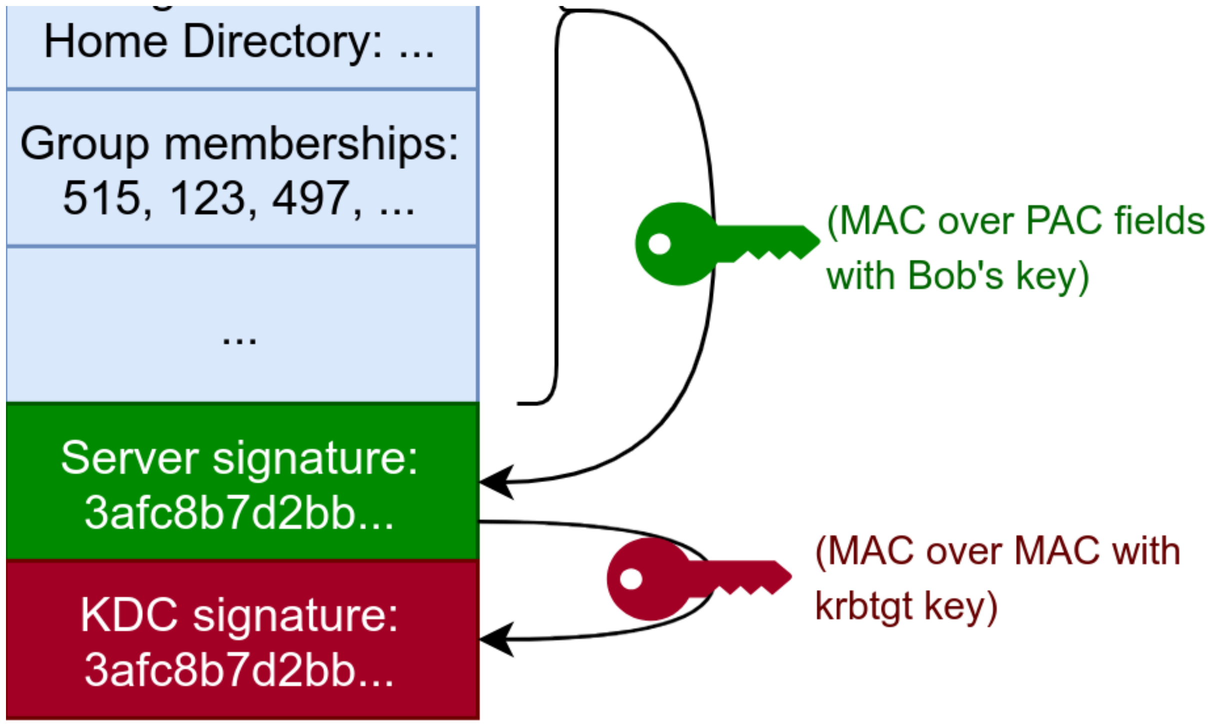


Attack step 4: combine to form final PAC



Spoofing with AES encryption types

The problem with MAC-over-MAC



Problem 1: A secure MAC over an insecure MAC is still an insecure MAC.

Problem 2: Collision-resistance against an attacker who knows the secret key is not a standard security requirements for MAC's.

Colliding AES∗-HMAC-SHA1 ciphers

- Alternatives to RC4-HMAC use HMAC-SHA1, truncated to 12 bytes.
- Birthday attack with known key: ≈ 1 in 2^48 collision chance.
- Brute-forcing this is much slower than finding an MD5 CPC, but still very feasible with a bunch of GPU's.
- **Allows spoofing PAC signatures that use AES∗-HMAC-SHA1**.

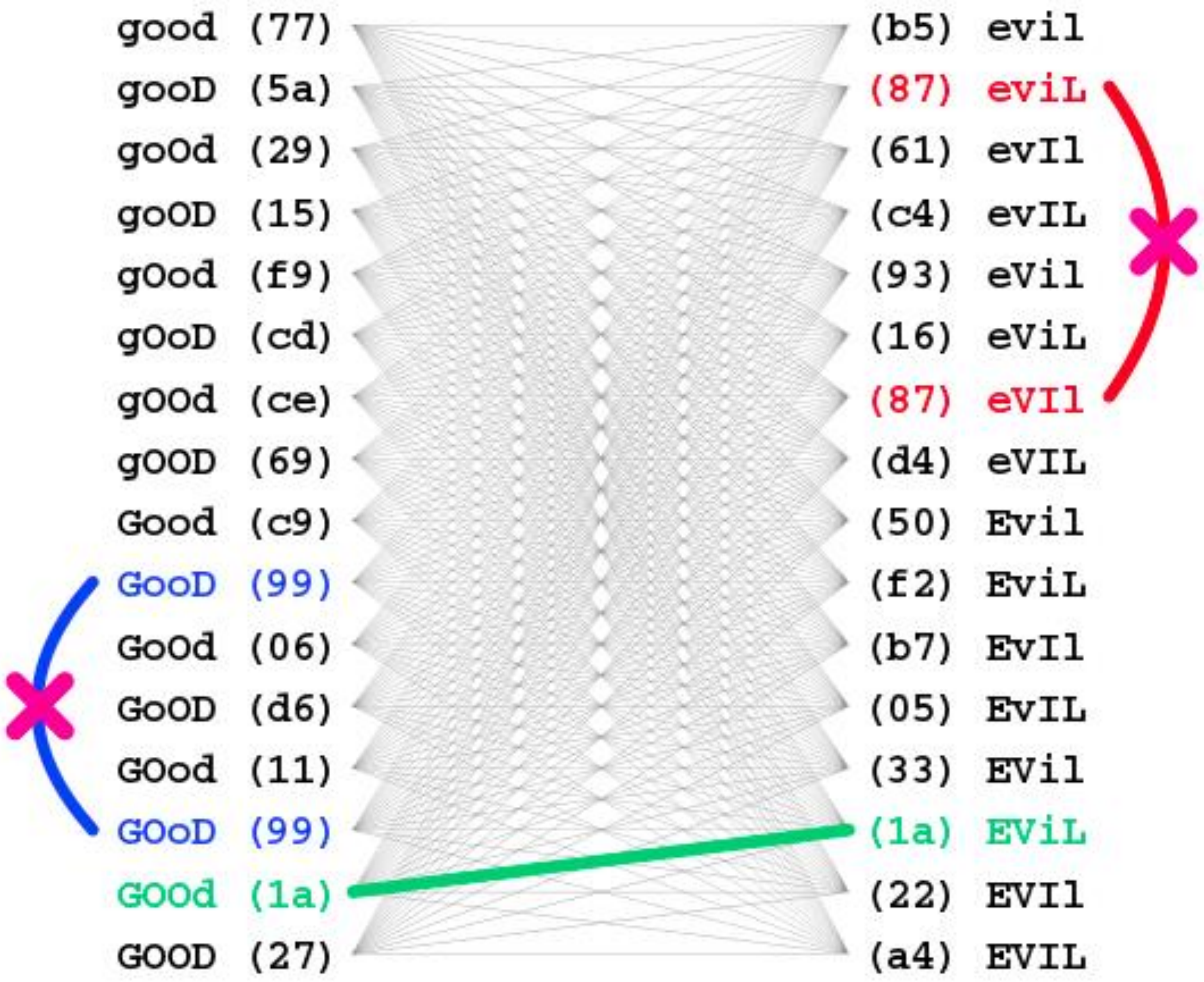
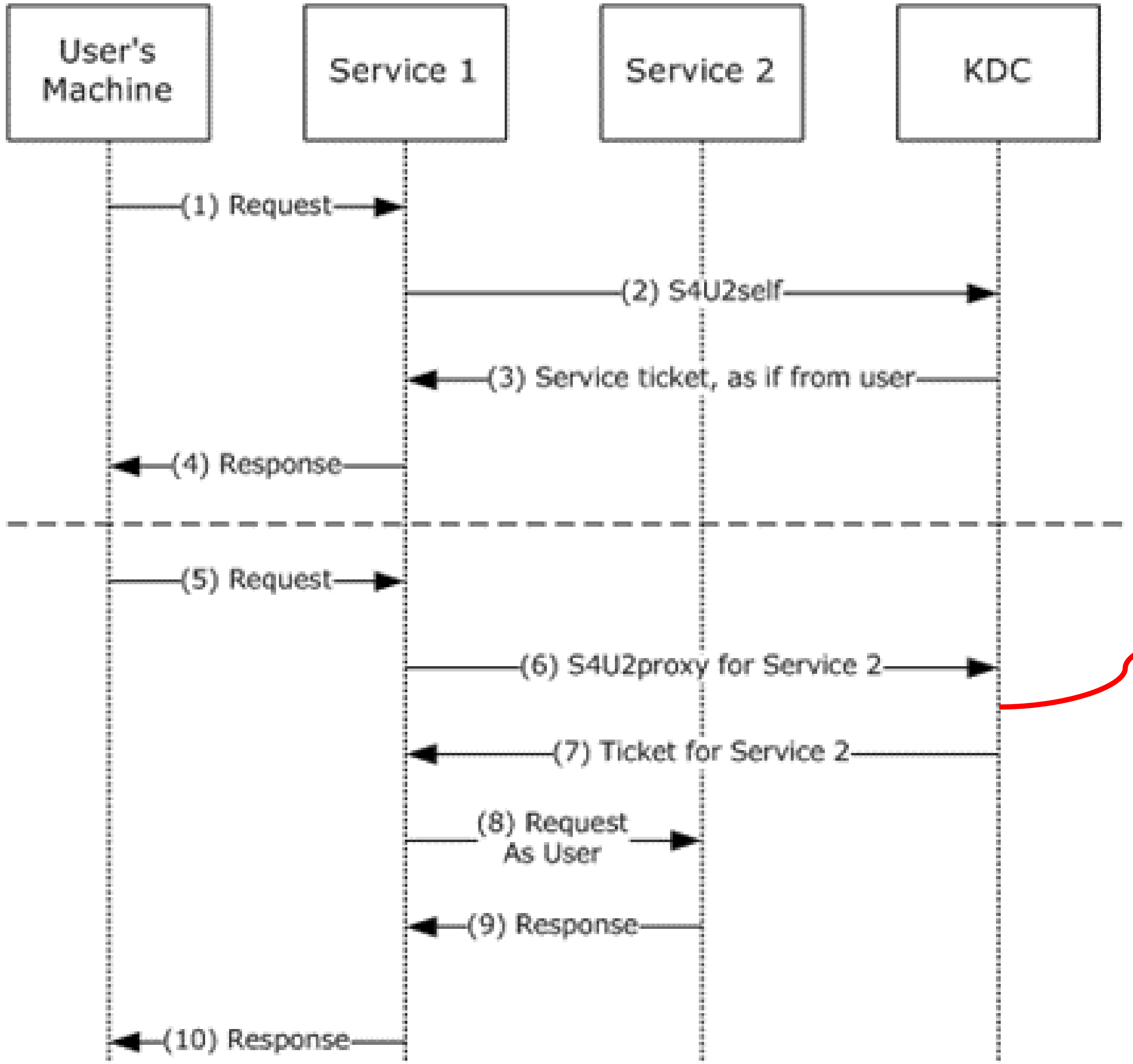


Image source: Cmglee, Wikimedia Commons

We spoofed a PAC. Now what?

PAC signatures and constrained delegation



KDC signature validation happens here!

Image source: Microsoft Open Specifications

A successful (but limited) exploit

- Well-known attack: if you compromise a server A, and server A has a constrained delegation relationship with server B; you can impersonate users when connecting to server B.
- However, accounts in the *Protected Users* group can't be delegated.
- Common for administrators highly-privileged users.
- PAC spoofing: turn a user into domain admin while delegating.

- Conclusion: we can bypass this security feature!
- Impact similar to "bronze bit attack" (CVE-2020-17049).

black hat EUROPE 2022

Follow-up

The patches

- Disclosed in May 2021
- Patches released in November 2022 (!)

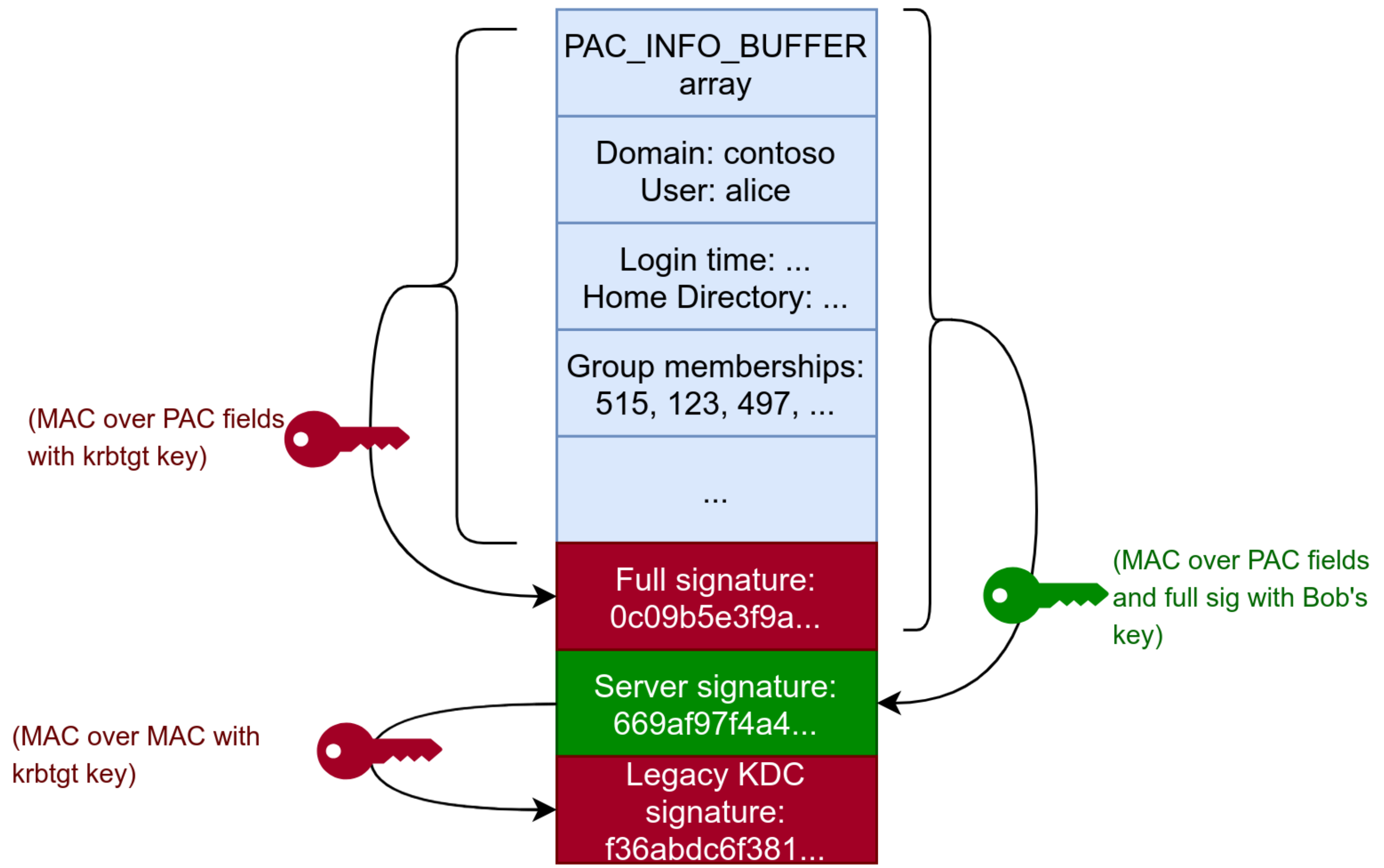
- RC4-HMAC flaw: CVE-2022-37966
- General PAC signature flaw: CVE-2022-37967

CVE-2022-37966 fix

- By default: accounts can't get ticket with RC4-HMAC session key
- … unless: account has **ms-DS-SupportedEncryptionType** flag that allows it
- RC4-HMAC can be re-enabled domain-wide by updating **DefaultDomainSupportedEncTypes** registry key
- Windows event 42 when account does not have keys for AES ciphers

- Side-effect: prevent RC4 Kerberoasting (?)
- May limit overpass-the-hash as well

CVE-2022-37967 fix



November 2022: signature added; not checked

December 2022: audit event 43/44 if wrong

July 2023 : “enforcement mode”

Can skip ahead with KrbtgtFullPacSignature setting.

Without enforcement mode, you are (probably) **still vulnerable to the AES*-HMAC-SHA1 collision!**

Black Hat Sound Bytes

- RC4-HMAC is broken. Not just in theory but also in practice.
- MD5 collisions are apparently still relevant in 2022.
- AD security relies on legacy crypto protocols that are very hard to modify or get rid of.