RID HIJACKING:

Maintaining Access on Windows Machines.

Sebastián Castro

sebastian.castro@csl.com.co

@r4wd3r

r4wd3r

St. Pölten, Austria.

2018

RID HIJACKING

```
C:\> net user r4wd3r

Username                r4wd3r
Full User name          Sebastián Castro
Comment                 Infosec nerd, xpltdev, win
                        sec, opera singer
User's comment          Terrible at MS Paint :(
Country/region code     Colombia
Account active          No
First logon             1993/05/03  23:56

User profile            Technical & Research Lead
                        <at> CSL Labs
Work directory          https://csl.com.co
```

Agenda

Agenda

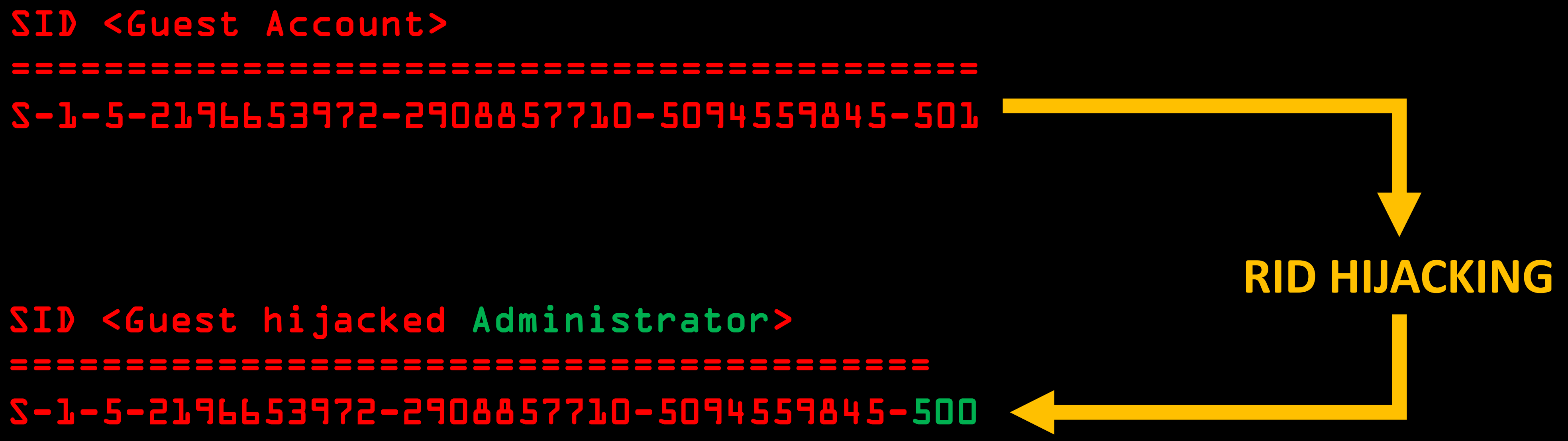
What is RID Hijacking?

- A new **persistence** technique that affects **ALL** Windows Systems since **NT**. (Haven't tried this on Windows 95 nor Phone ☹).
- A stealthy way to maintain access by **only using OS resources.**
- A method which takes advantage of **important security issues** found at the Windows Security Architecture.

⚠ **Not reliable on Domain Controllers (yet).**

What does it do?

This technique **hijacks the RID** of any **existing user account** on the victim host and assigns it to **another one**.



```
SID <Guest Account>
===========================================
S-1-5-2196653972-2908857710-5094559845-501



SID <Guest hijacked Administrator>
=========================================
S-1-5-2196653972-2908857710-5094559845-500
```

What does it do?

0x01. Assigns the privileges of the **hijacked** account to the **hijacker** one, even if the **hijacked** account is **disabled**.

0x02. Allows to authenticate with the **hijacker** account credentials (also remotely, depending on machine's configuration), and obtain authorized access as the **hijacked** user.

0x03. Permits to register any operation executed on the event log as the **hijacked** user, despite of being logged on as the **hijacker** one.

What does it do?

0x01. Assigns the privileges of the hijacked account to the hijacker one, even if the hijacked account is disabled.

0x02. Allows to authenticate with the hijacker account credentials (also remotely, depending on machine's configuration), and obtain authorized access as the hijacked user.

0x03. Permits to register any operation executed on the event log as the **hijacked** user, despite of being logged on as the **hijacker** one.

What does it do?

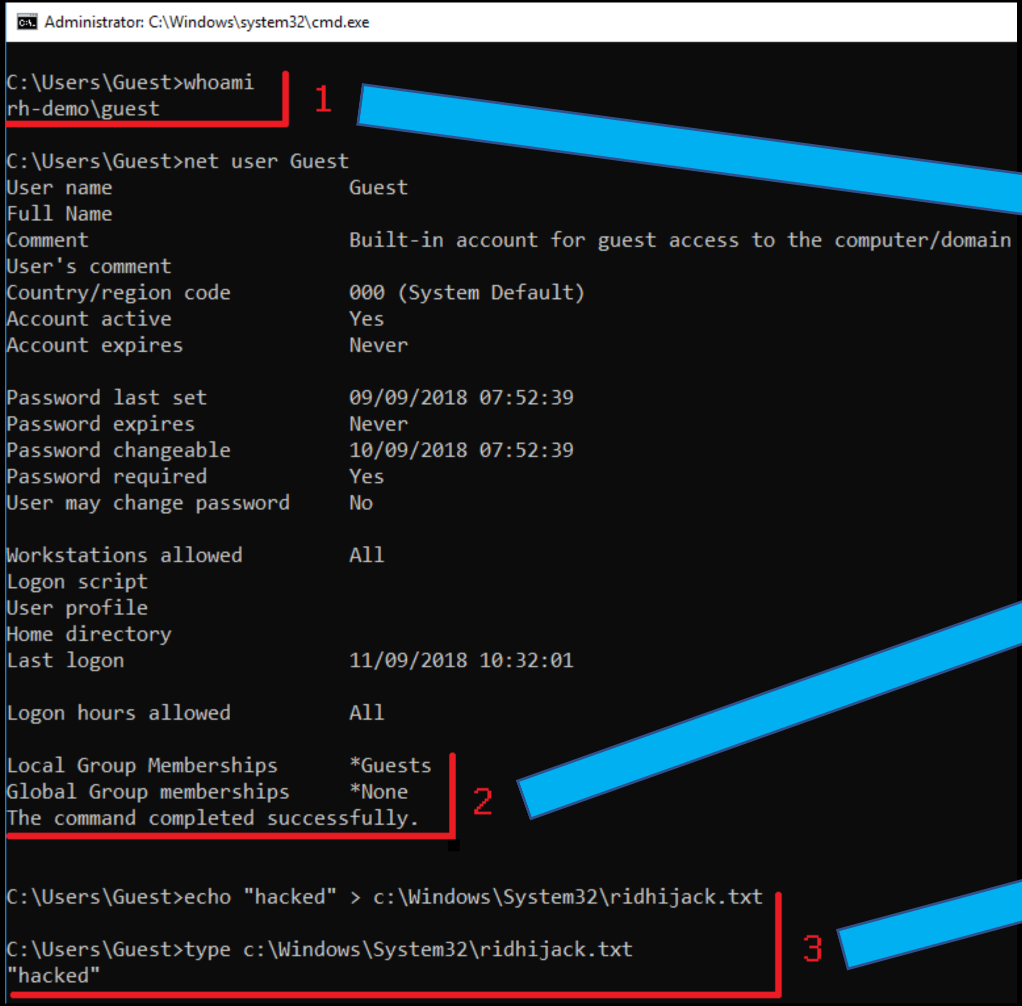
0x01. Assigns the privileges of the hijacked account to the hijacker one, even if the hijacked account is disabled.

0x02. Allows to authenticate with the hijacker account credentials (also remotely, depending on machine's configuration), and obtain authorized access as the hijacked user.

0x03. Permits to register any operation executed on the event log as the hijacked user, despite of being logged on as the hijacker one.

RID HIJACKING

How does it look like?



Administrator: C:\Windows\system32\cmd.exe

```
C:\Users\Guest>whoami
rh-demo\guest

C:\Users\Guest>net user Guest
User name                    Guest
Full Name
Comment                      Built-in account for guest access to the computer/domain
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            09/09/2018 07:52:39
Password expires             Never
Password changeable          10/09/2018 07:52:39
Password required            Yes
User may change password     No

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   11/09/2018 10:32:01

Logon hours allowed          All

Local Group Memberships      *Guests
Global Group memberships     *None
The command completed successfully.

C:\Users\Guest>echo "hacked" > c:\Windows\System32\ridhijack.txt

C:\Users\Guest>type c:\Windows\System32\ridhijack.txt
"hacked"
```

whoami

```
C:\Users\Guest>whoami
rh-demo\guest
```

1

net user Guest

```
Local Group Memberships      *Guests
Global Group memberships     *None
The command completed successfully.
```

2

writing on System32 folder

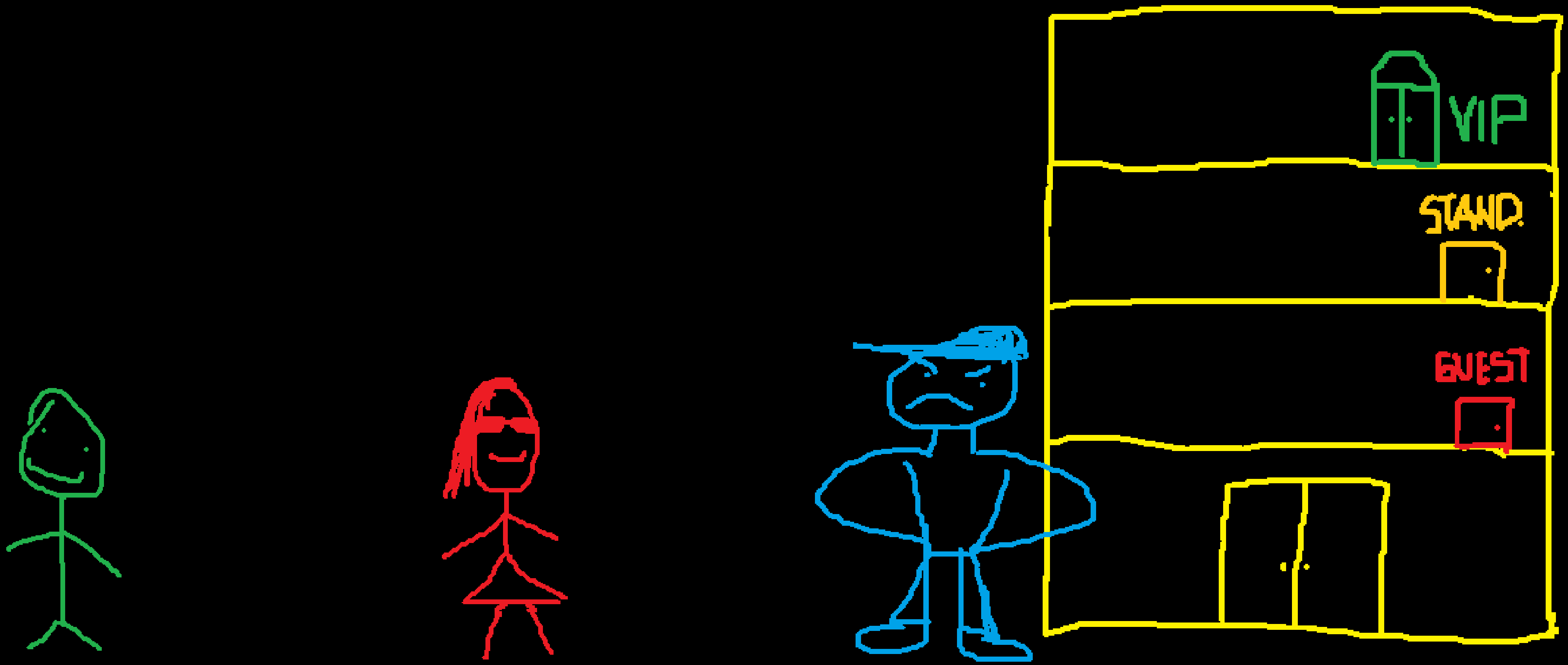
```
C:\Users\Guest>echo "hacked" > c:\Windows\System32\ridhijack.txt

C:\Users\Guest>type c:\Windows\System32\ridhijack.txt
"hacked"
```

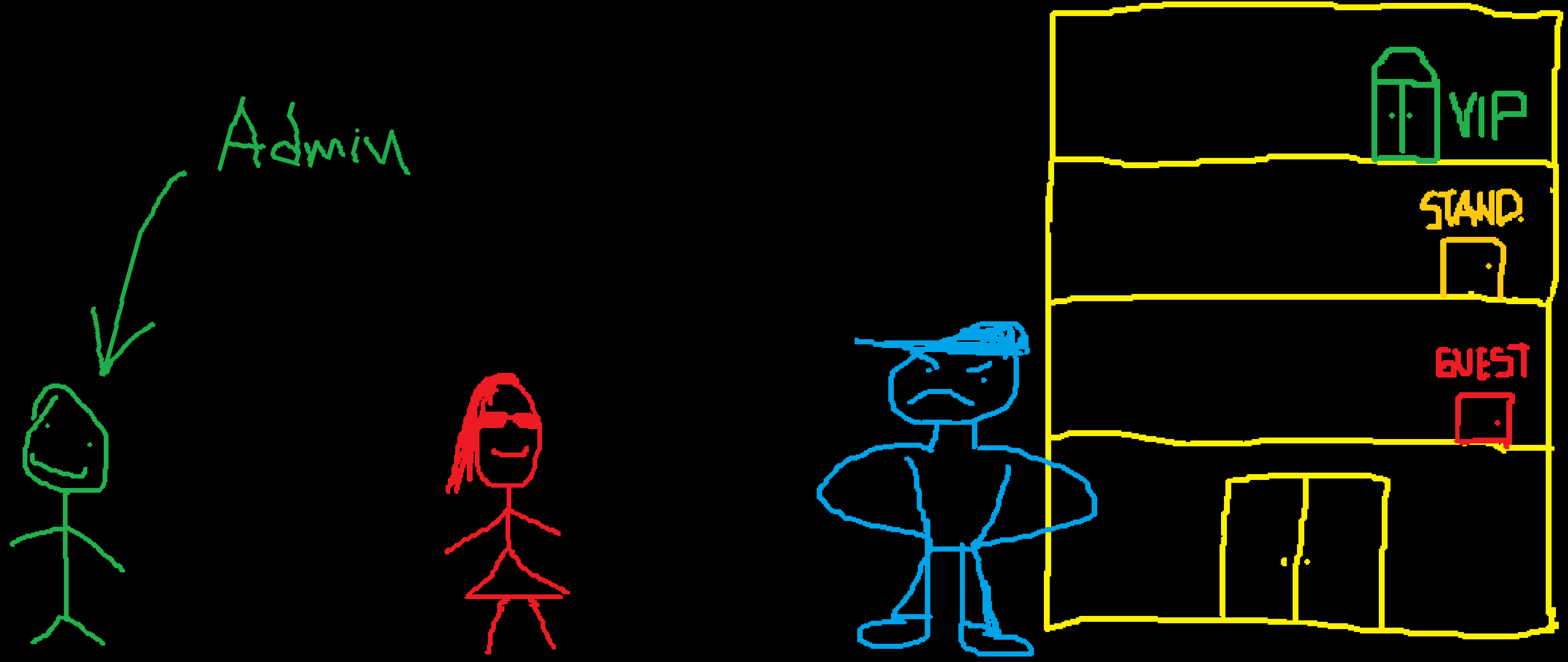
3

Agenda

A Windows Logon Story…



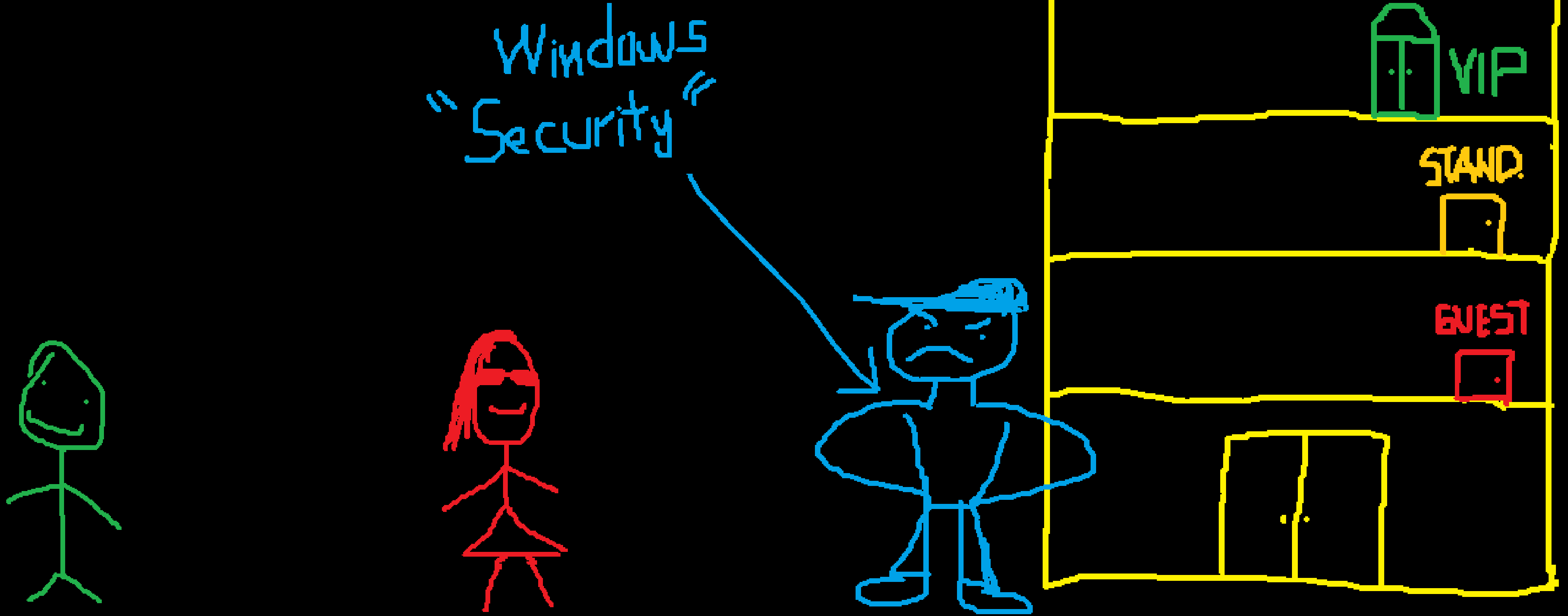
A Windows Logon Story…



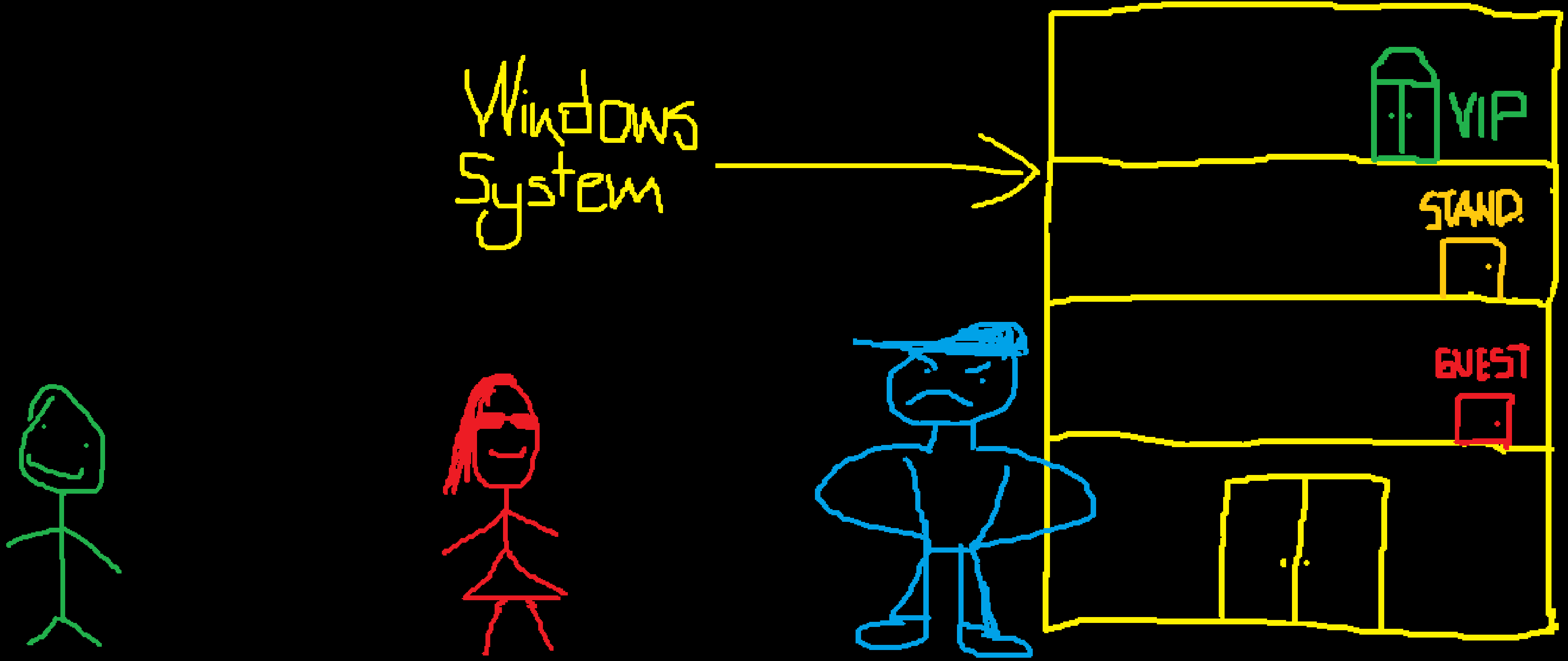
A Windows Logon Story...

Guest
VIP
STAND.
GUEST

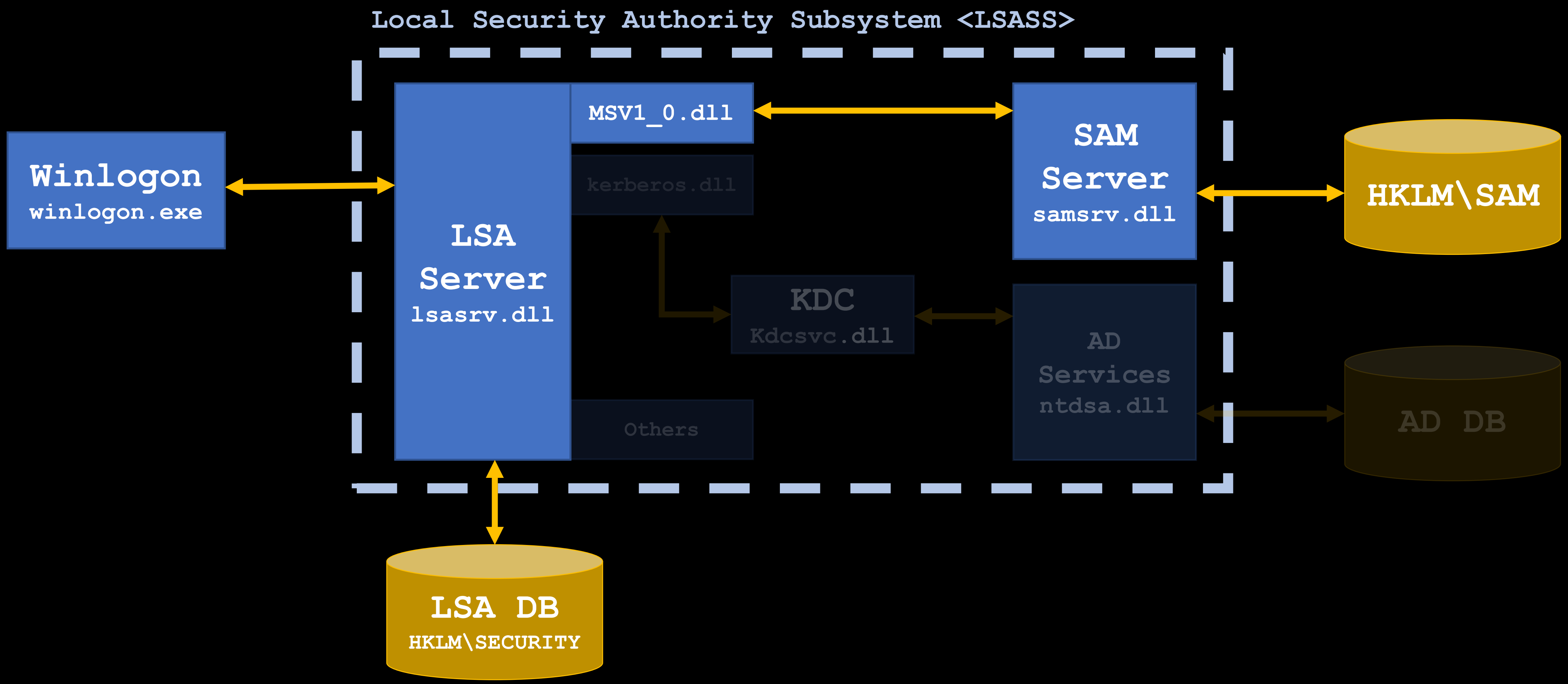
A Windows Logon Story…



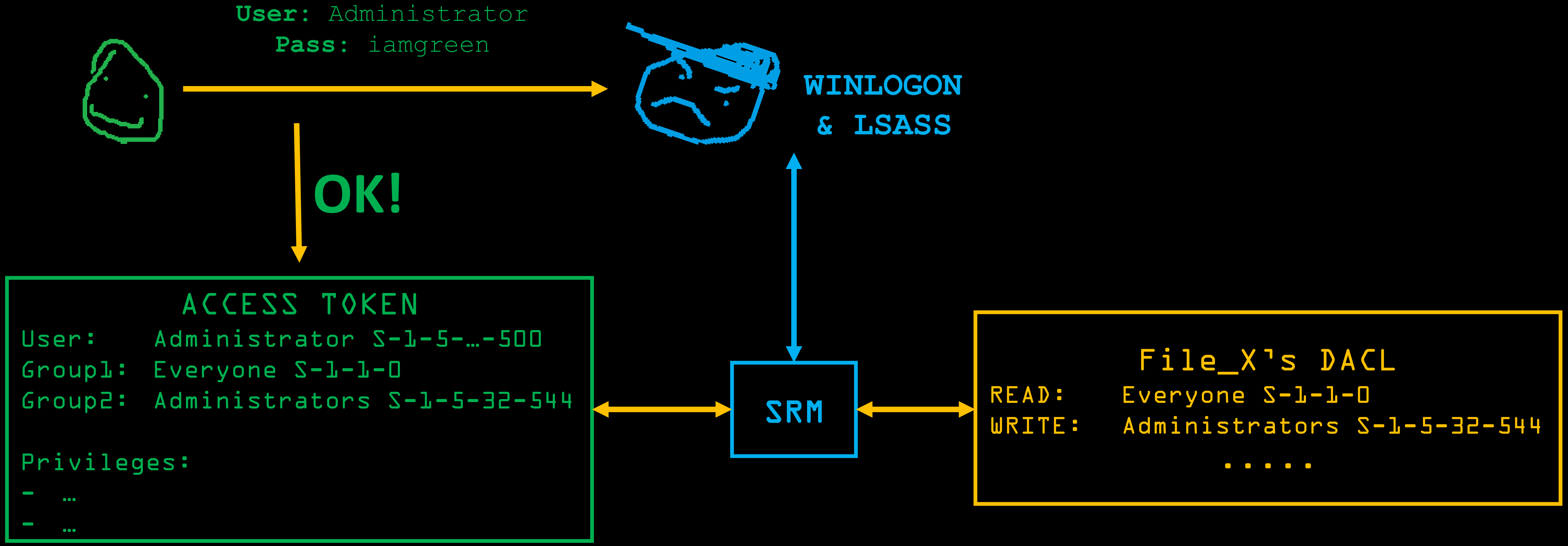
A Windows Logon Story…



Windows Security Architecture



Quick Logon Overview



ACCESS TOKEN

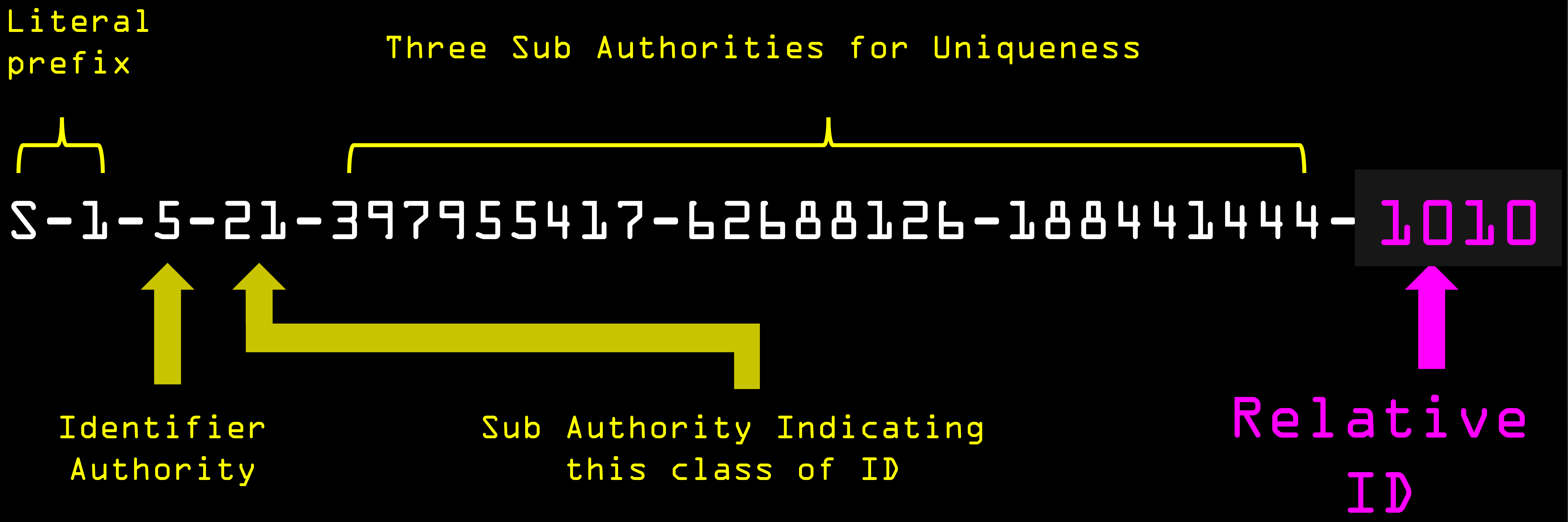
```
User:    Administrator S-1-5-…-500
Group1:  Everyone S-1-1-0
Group2:  Administrators S-1-5-32-544

Privileges:
-  …
-  …
```

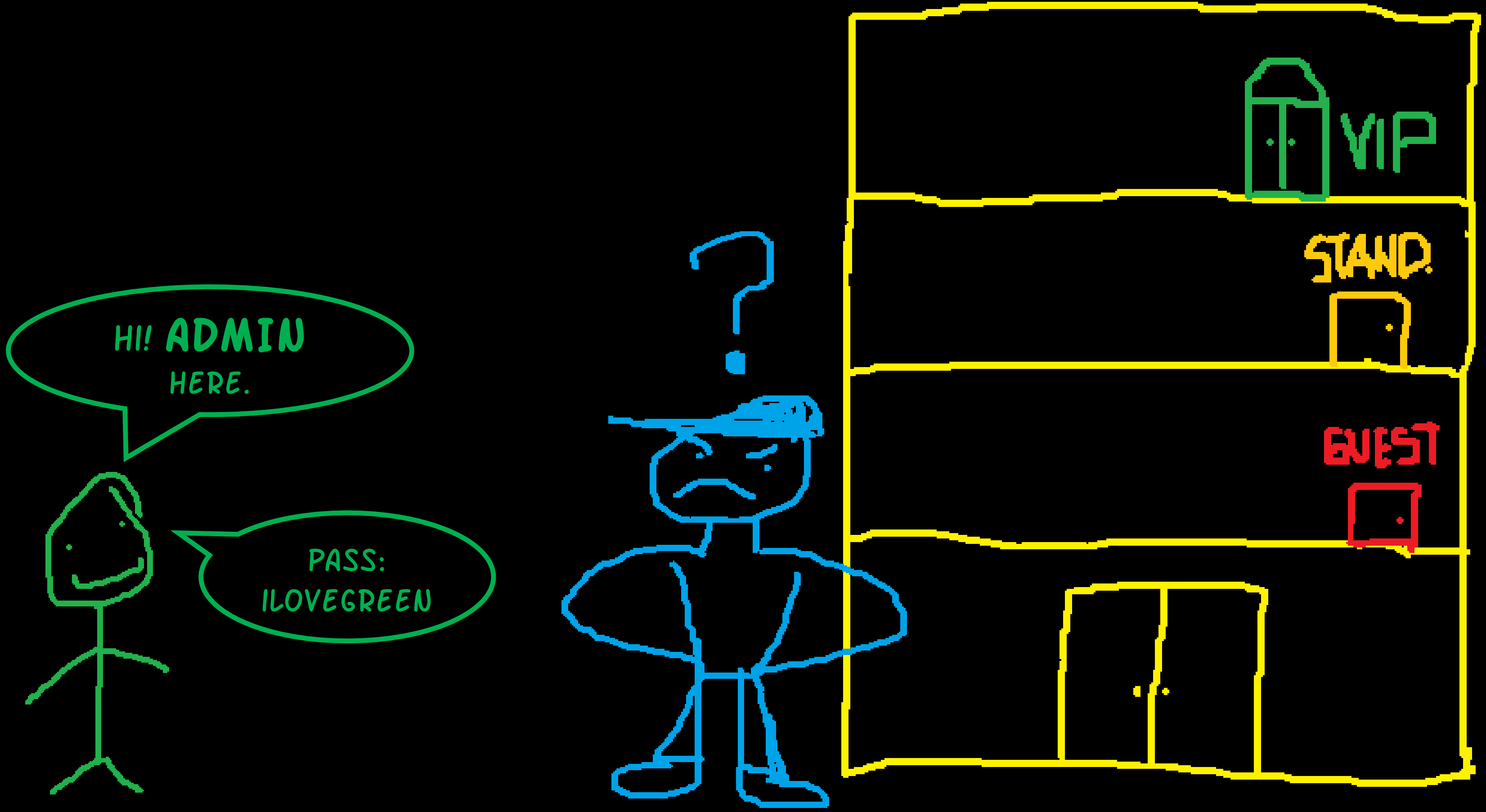
File_X's DACL

```
READ:    Everyone S-1-1-0
WRITE:   Administrators S-1-5-32-544
              .....
```

Security Identifiers <SID>


Literal prefix
Three Sub Authorities for Uniqueness
S-1-5-21-397955417-62688126-188441444-1010
Identifier Authority
Sub Authority Indicating this class of ID
Relative ID

Authentication



Authentication Steps

0x01. WINLOGON Initialization.

0x02. WINLOGON calls LOGONUI (using CPs).

0x03. WINLOGON creates an unique LOGON SID.

0x04. WINLOGON calls LSASS and prepares a handle for an Authentication Package.

Authentication Steps

0x05. WINLOGON sends logon info to the MSV1_0 calling LsaLogonUser.

Logon Info:

Username/Password.

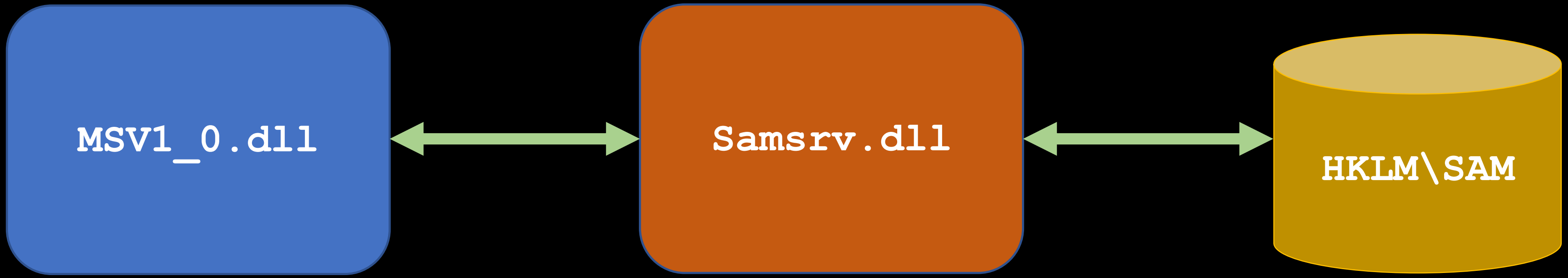
LOGON SID.

MSV1_0 is also used on domain-member computers when are disconnected of the network.

Authentication Steps

0x06. MSV1_0 sends username and hashed password to the SAMSRV.

0x07. SAMSRV queries on the SAM database with the logon data, retrieving some security info.



Authentication Steps

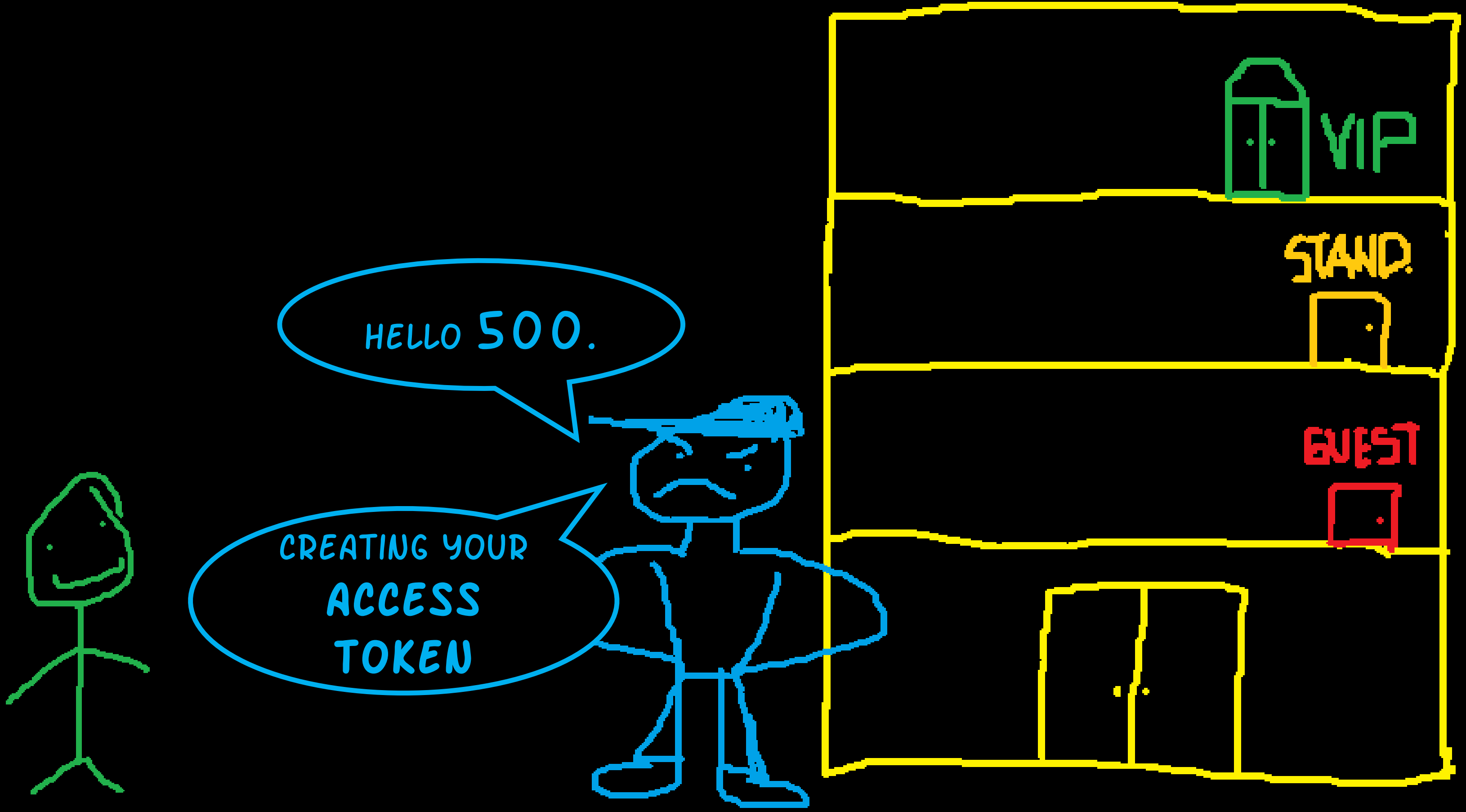
0x08. MSV1_0 checks the information obtained from the SAMSRV response.

0x09. If OK, MSV1_0 generates a LUID for the session.

0x0A. MSV1_0 sends the login information (including LUID) to LSASS.

⚠ All the data sent will be used for the further access token creation.

Authorization



Access Token

Object used by the SRM to identify the security context of a process.

LSASS creates an initial access token for every user which logs on.

Child processes inherit a copy of the token of their creator.

Processes in a user's session will be executed using the same access token.

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session
Mandatory Policy
Default primary group
Default DACL
User account SID
Group 1 SID
…
Group n SID
Restricted SID 1
…
Restricted SID n
Privilege 1
…
Privilege n

Authorization Steps

0x0B. LSASS checks the LSA database for the user's allowed access.

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session
Mandatory Policy
Default primary group
Default DACL
User account SID

Authorization Steps

0x0B. LSASS checks the LSA database for the user's allowed access.

0x0C. LSASS adds the Groups, SIDs and privileges to the access token.

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session
Mandatory Policy
Default primary group
Default DACL
User account SID
Group 1 SID
…
Group n SID
Restricted SID 1
…
Restricted SID n
Privilege 1
…
Privilege n

Authorization Steps

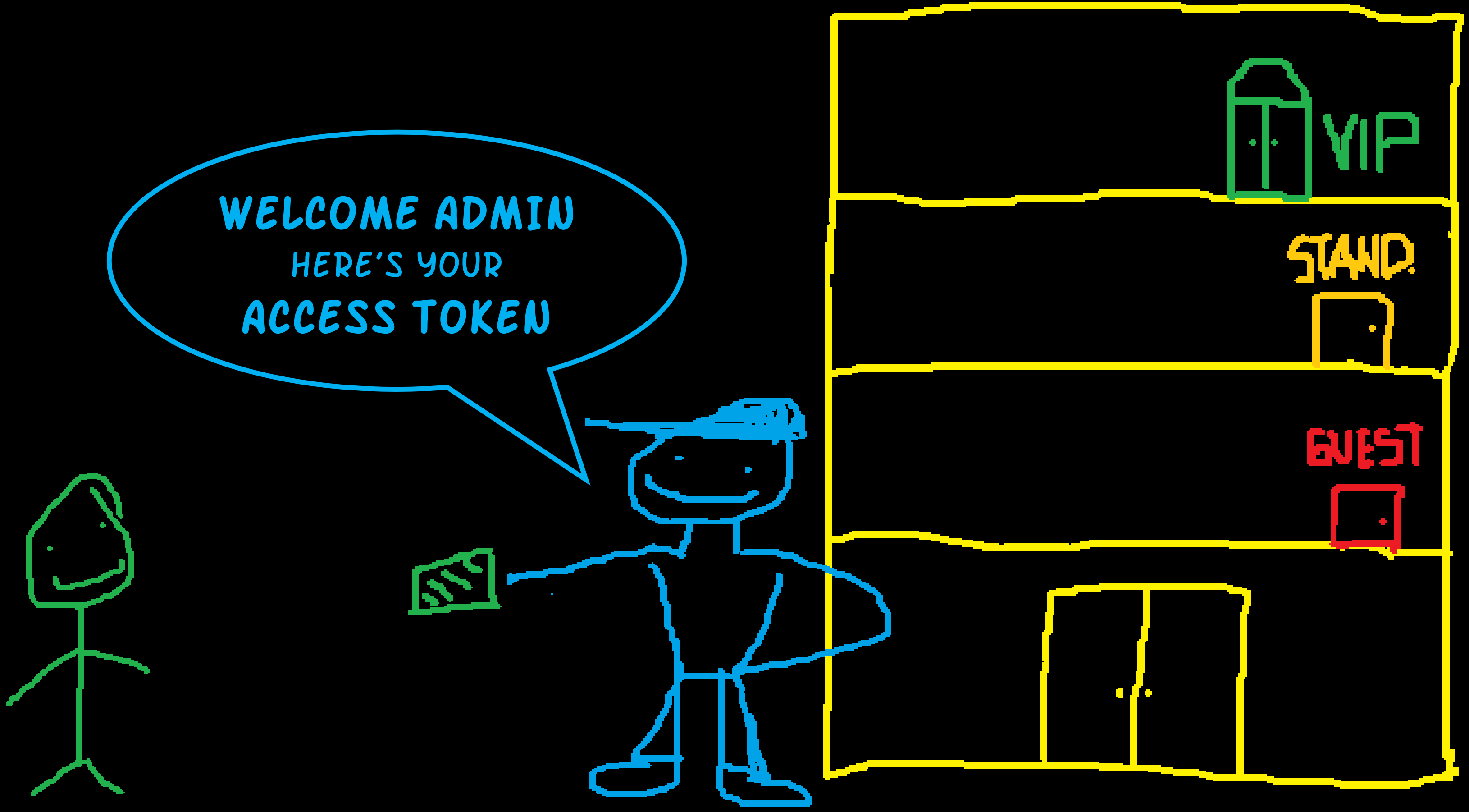
0x0B. LSASS checks the LSA database for the user's allowed access.

0x0C. LSASS adds the Groups, SIDs and privileges to the access token.

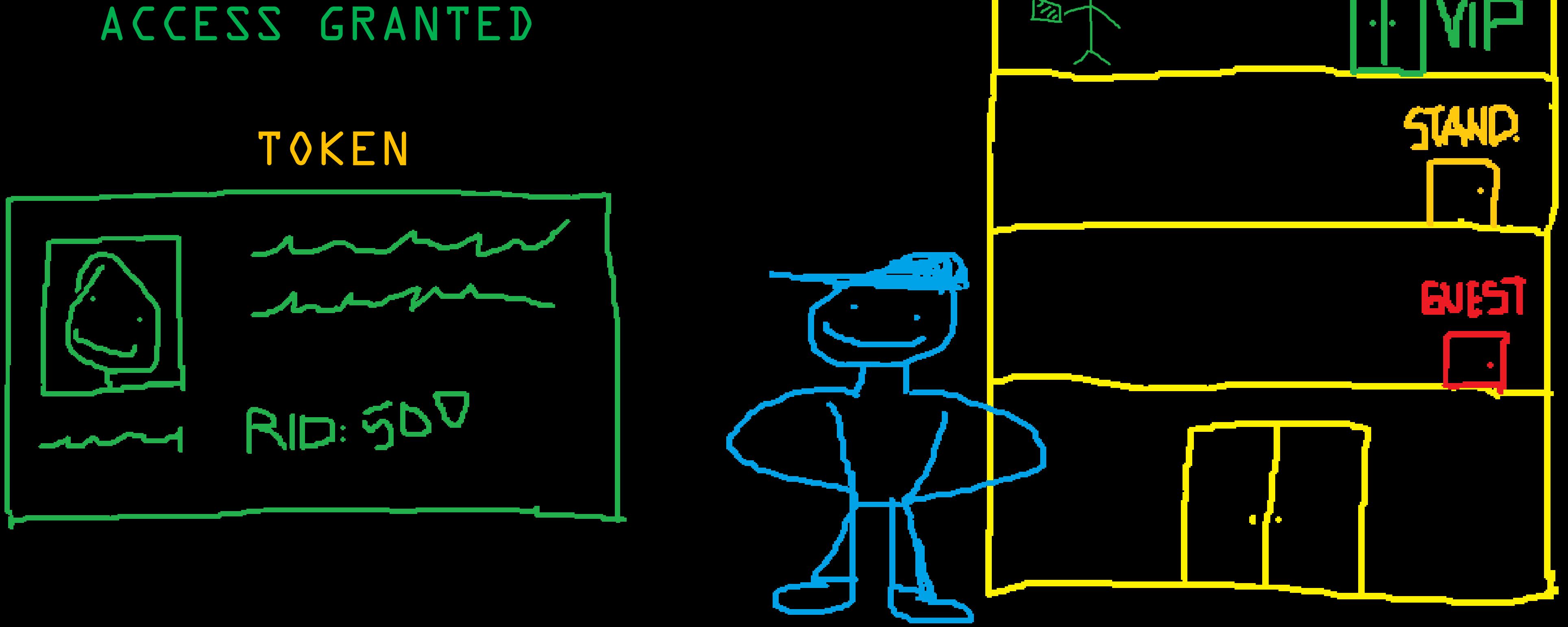
0x0D. LSASS formally creates a primary access token.

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session
Mandatory Policy
Default primary group
Default DACL
User account SID
Group 1 SID
…
Group n SID
Restricted SID 1
…
Restricted SID n
Privilege 1
…
Privilege n

Authorization



Authorization



Agenda

Understanding the attack

How is the user identified by the system after being successfully authenticated?

Understanding the attack

How is the user identified by the system after being successfully authenticated?

S-1-5-21966553972-2908857710-5094559845-500

Understanding the attack

How is the user identified by the system after being successfully authenticated?

S-1-5-2196653972-2908857710-5094559845-500

How does the system associate an username with his SID?

Understanding the attack

How is the user identified by the system after being successfully authenticated?

S-1-5-2196653972-2908857710-5094559845-500

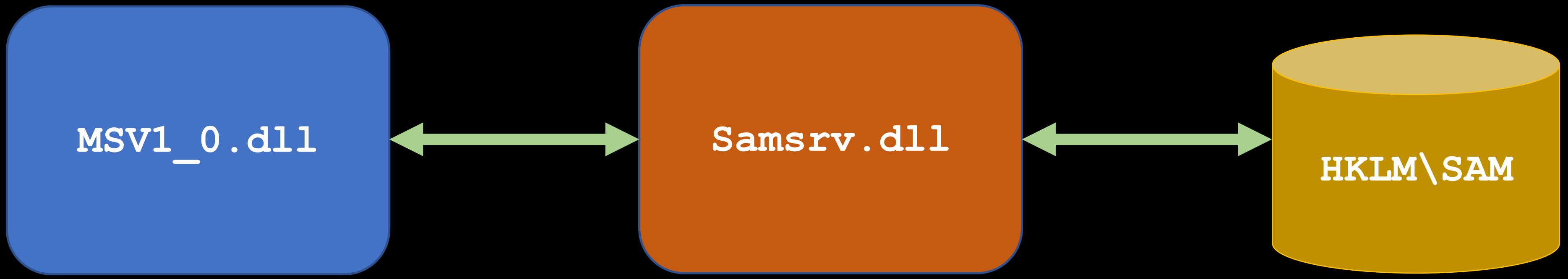
How does the system associate an username with his SID?

Using the Samsrv.dll black magic :)

Remembering…

0x06. MSV1_0 sents username and hashed password to the SAMSRV.

0x07. SAMSRV queries on the SAM database with the logon data, retrieving some security info.



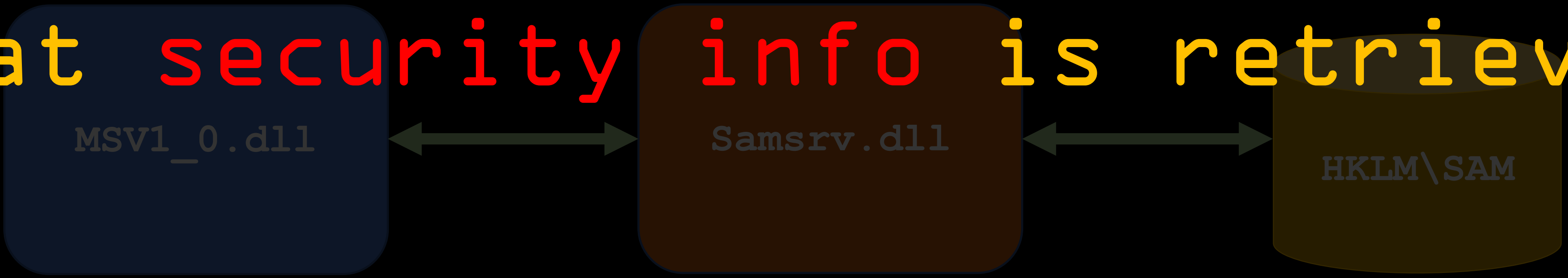
Remembering...

0x06. MSV1_0 sents username and hashed password to the SAMSRV.

0x07. SAMSRV queries on the SAM database with the logon data, retrieving some security info.

How is the username associated with the SID?

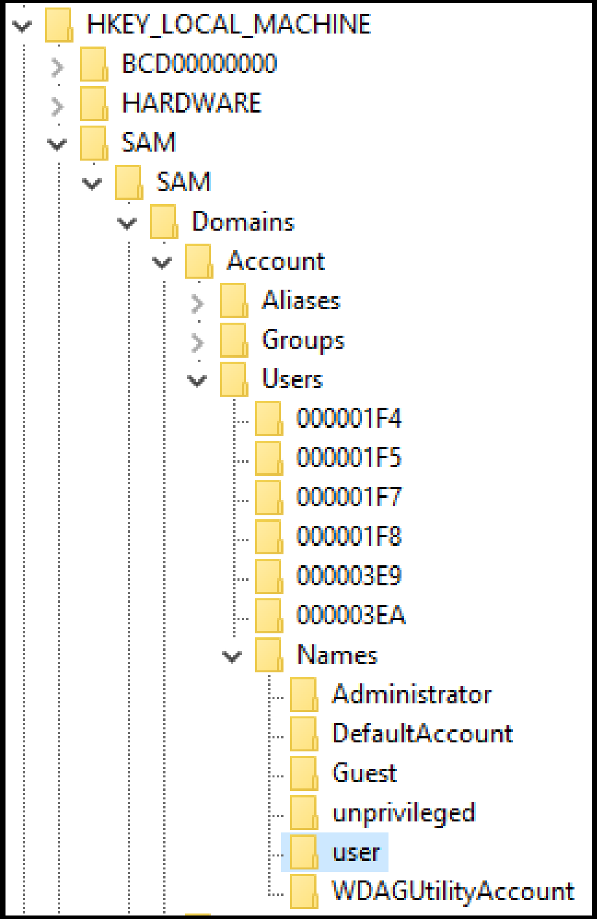
What security info is retrieved?



Samsrv.dll and SAM

SAMSRV looks for the username at the SAM database.

HKLM\SAM\SAM\Domains\Account\Users\Names



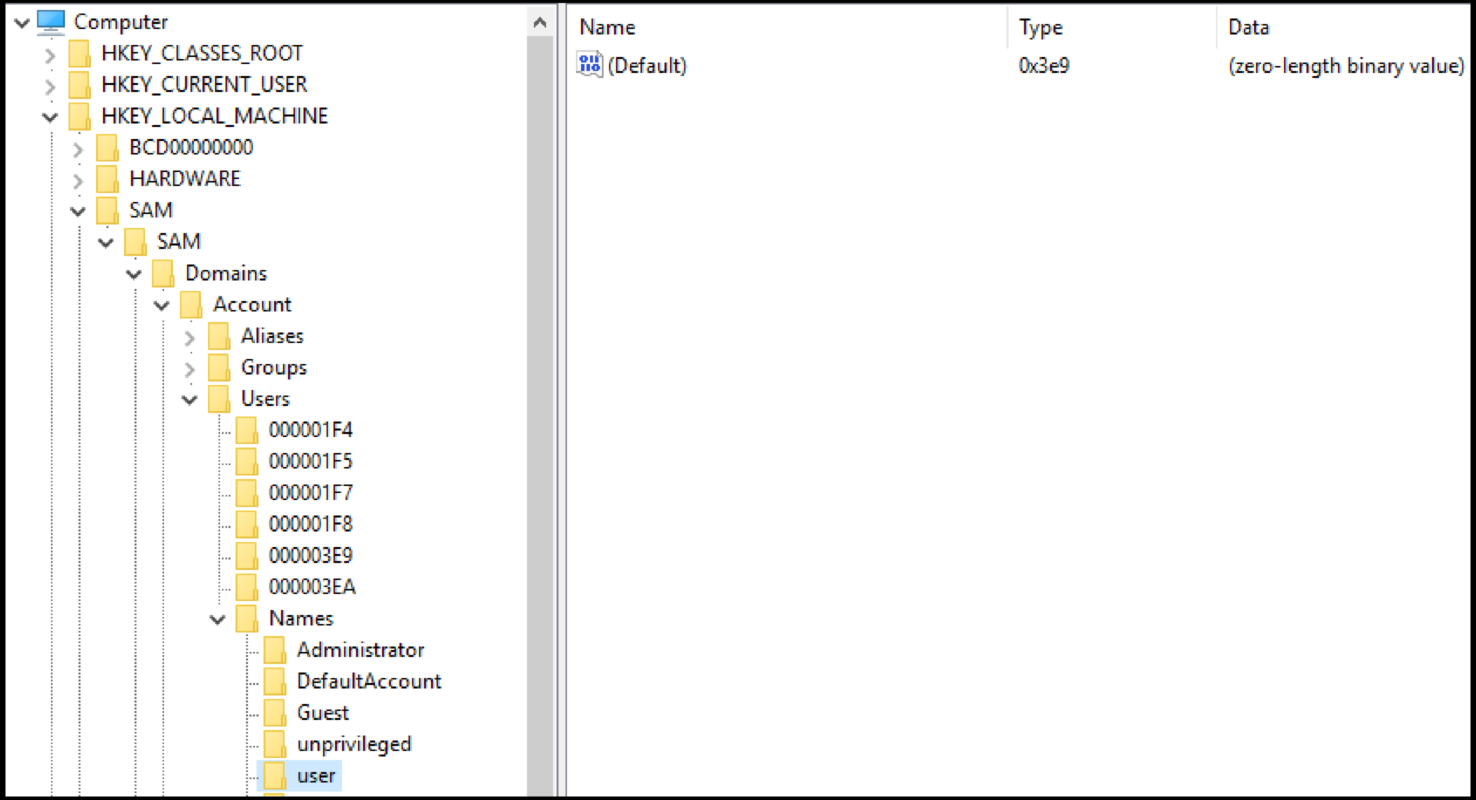
```
HKEY_LOCAL_MACHINE
  BCD00000000
  HARDWARE
  SAM
    SAM
      Domains
        Account
          Aliases
          Groups
          Users
            000001F4
            000001F5
            000001F7
            000001F8
            000003E9
            000003EA
            Names
              Administrator
              DefaultAccount
              Guest
              unprivileged
              user
              WDAGUtilityAccount
```

Samsrv.dll and SAM

SAMSRV looks for the username at the SAM database.

Each key contains a REG_BINARY value.

HKLM\SAM\SAM\Domains\Account\Users\Names



```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
      SAM
        Domains
          Account
            Aliases
            Groups
            Users
              000001F4
              000001F5
              000001F7
              000001F8
              000003E9
              000003EA
              Names
                Administrator
                DefaultAccount
                Guest
                unprivileged
                user
```

Name	Type	Data
(Default)	0x3e9	(zero-length binary value)

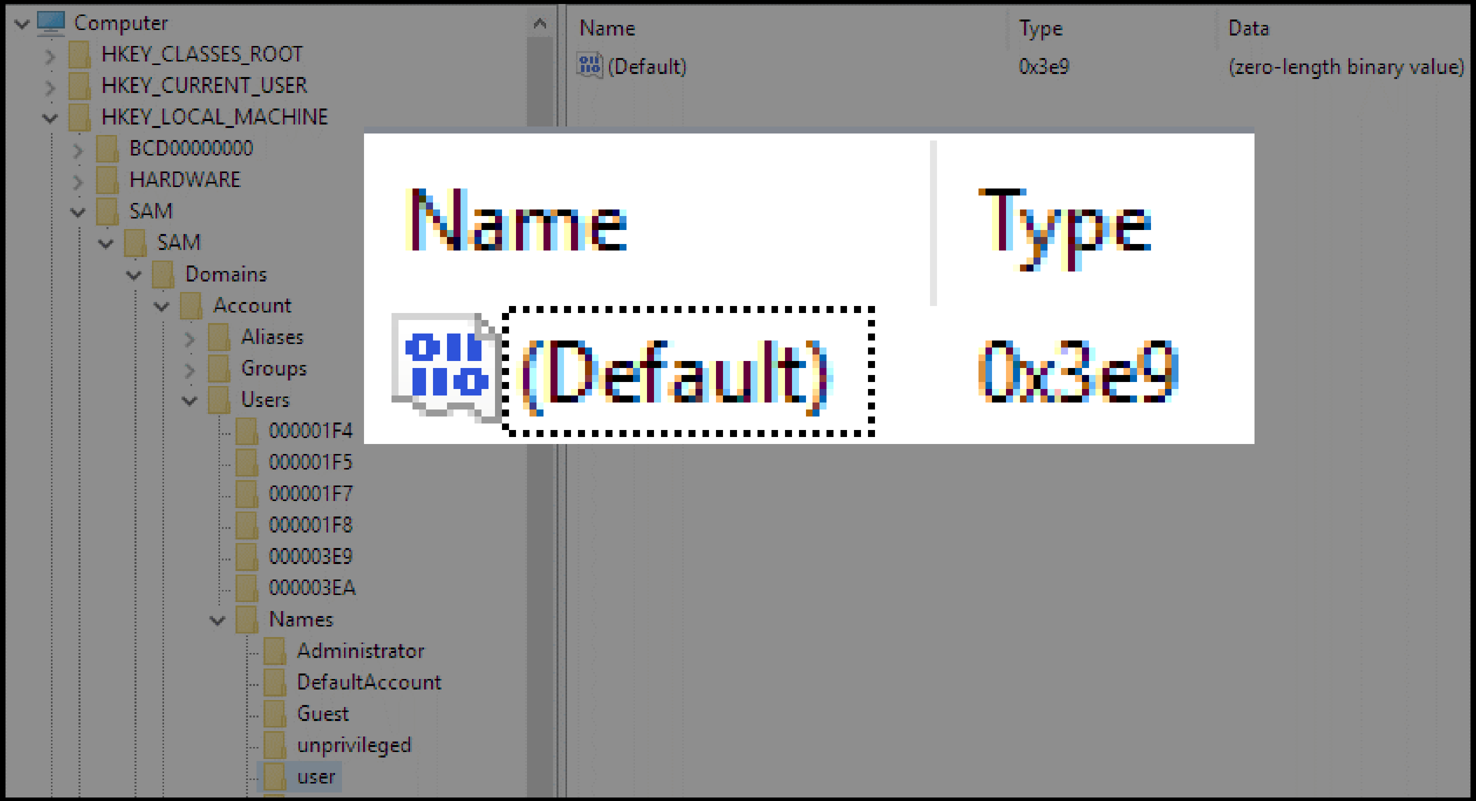
Samsrv.dll and SAM

SAMSRV looks for the username at the SAM database.

Each key contains a REG_BINARY value.

The REG_BINARY has as Type the RID of the account.

HKLM\SAM\SAM\Domains\Account\Users\Names



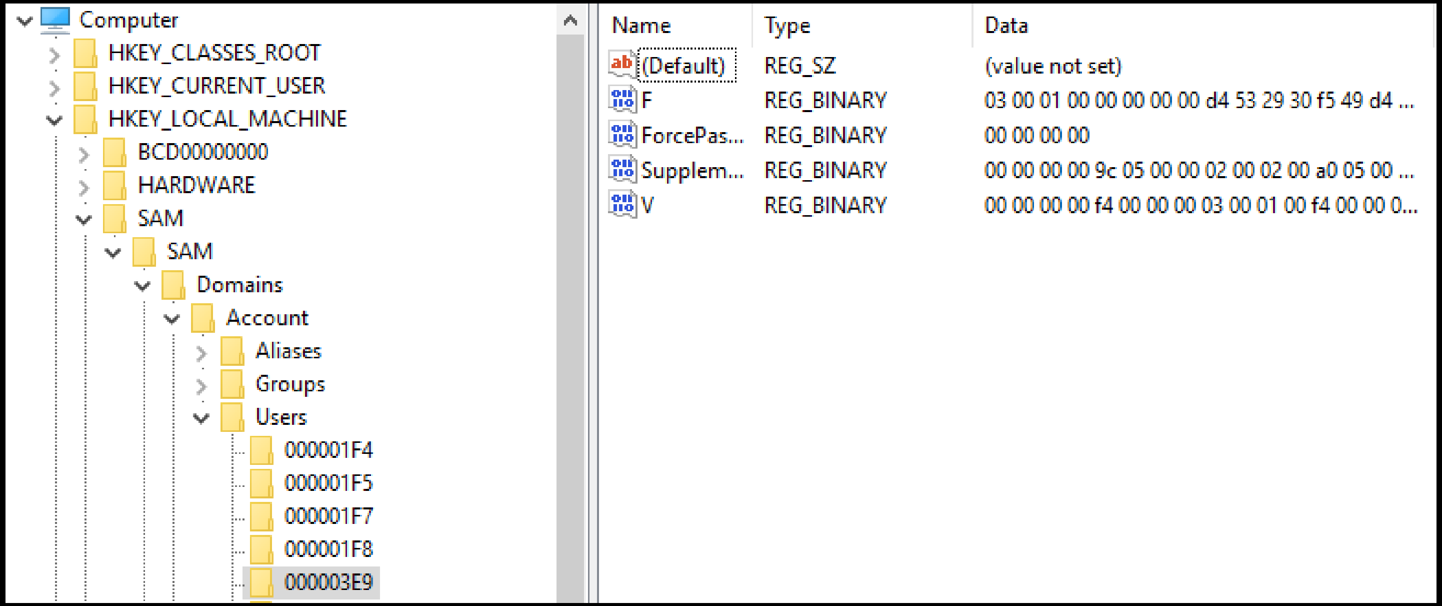
```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
      SAM
        Domains
          Account
            Aliases
            Groups
            Users
              000001F4
              000001F5
              000001F7
              000001F8
              000003E9
              000003EA
              Names
                Administrator
                DefaultAccount
                Guest
                unprivileged
                user
```

Name	Type	Data
(Default)	0x3e9	(zero-length binary value)

Samsrv.dll and MSV1_0.dll

SAMSRV looks for the key associated with the RID.

HKLM\SAM\SAM\Domains\Account\Users



```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
      SAM
        Domains
          Account
            Aliases
            Groups
            Users
              000001F4
              000001F5
              000001F7
              000001F8
              000003E9
```

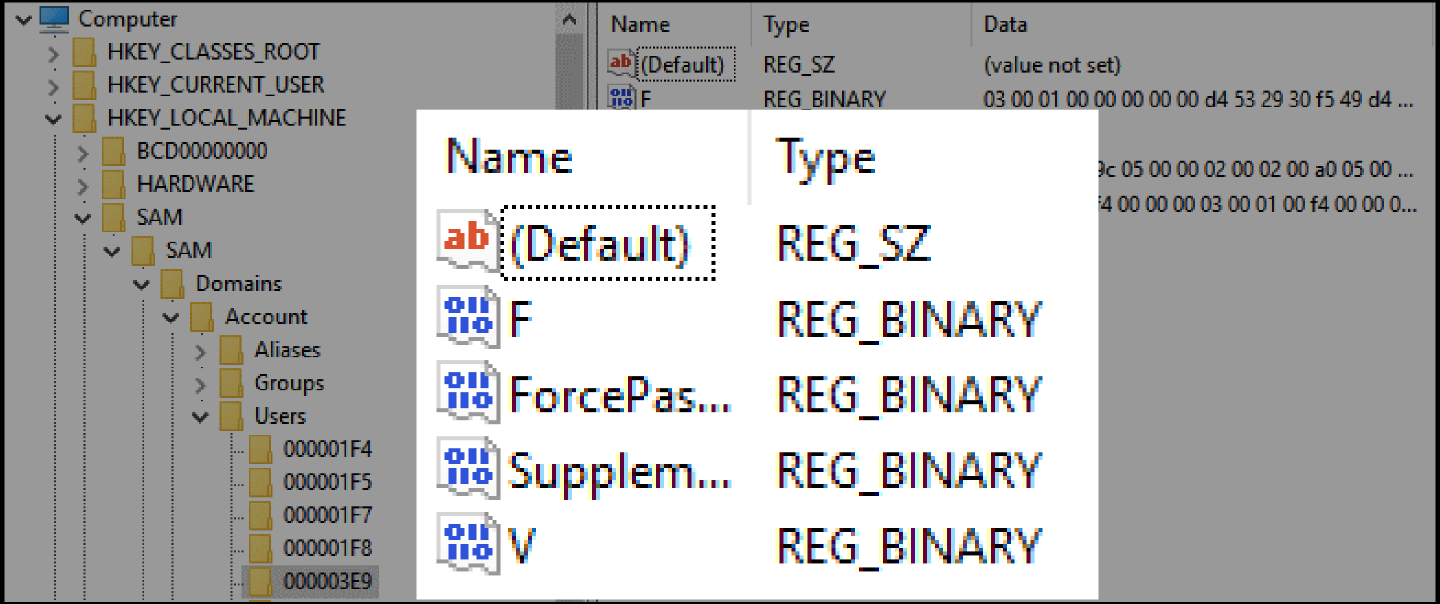
Name	Type	Data
(Default)	REG_SZ	(value not set)
F	REG_BINARY	03 00 01 00 00 00 00 00 d4 53 29 30 f5 49 d4 ...
ForcePas...	REG_BINARY	00 00 00 00
Supplem...	REG_BINARY	00 00 00 00 9c 05 00 00 02 00 02 00 a0 05 00 ...
V	REG_BINARY	00 00 00 00 f4 00 00 00 03 00 01 00 f4 00 00 0...

Samsrv.dll and MSV1_0.dll

SAMSRV looks for the key associated with the RID.

SAMSRV grabs all the data stored in the referenced key.

HKLM\SAM\SAM\Domains\Account\Users



```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
      SAM
        Domains
          Account
            Aliases
            Groups
            Users
              000001F4
              000001F5
              000001F7
              000001F8
              000003E9
```

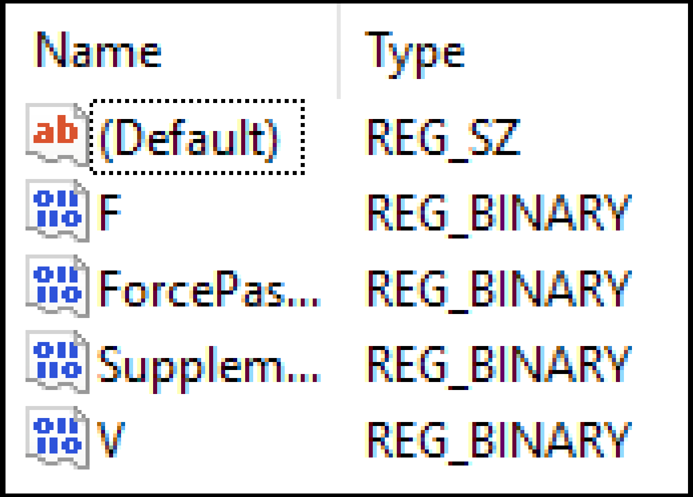
Name	Type	Data
(Default)	REG_SZ	(value not set)
F	REG_BINARY	03 00 01 00 00 00 00 00 d4 53 29 30 f5 49 d4 ...
ForcePas...	REG_BINARY	9c 05 00 00 02 00 02 00 a0 05 00 ...
Supplem...	REG_BINARY	f4 00 00 00 03 00 01 00 f4 00 00 0...
V	REG_BINARY	

Samsrv.dll and MSV1_0.dll

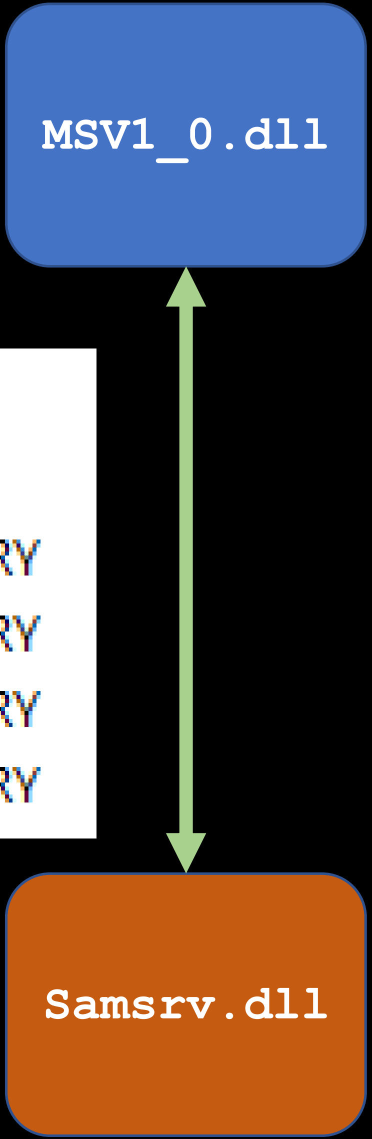
SAMSRV looks for the key associated with the RID.

SAMSRV grabs all the data stored in the referenced key.

MSV1_0.dll receives back all the data from SAMSRV.



Name	Type
(Default)	REG_SZ
F	REG_BINARY
ForcePas...	REG_BINARY
Supplem...	REG_BINARY
V	REG_BINARY



Understanding the attack

Why does the SAM store only the RID?

Understanding the attack

Why does the SAM store only the RID?

$$\underbrace{\text{S-1-5-2196653972-2908857710-5094559845}}_{\text{Consistent for all local users SIDs}}\text{-}\underbrace{\text{500}}_{\text{Relative}}$$

Understanding the attack

Why does the SAM store only the RID?

$$\underbrace{\text{S-1-5-2196653972-2908857710-5094559845}}_{\text{Consistent for all local users SIDs}}\text{-}\underbrace{\text{500}}_{\text{Relative}}$$

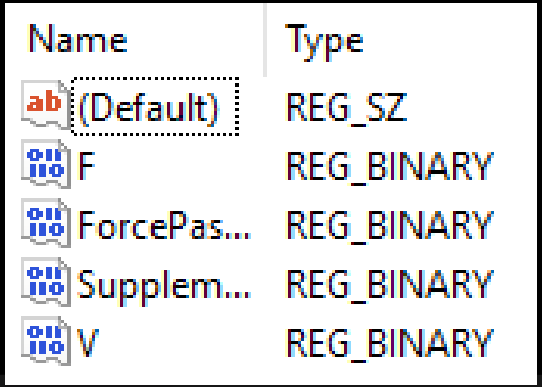
What info is retrieved from the SAM?

Understanding the attack

Why does the SAM store only the RID?

$$\underbrace{\text{S-1-5-2196653972-2908857710-5094559845}}_{\text{Consistent for all local users SIDs}}\text{-}\underbrace{\text{500}}_{\text{Relative}}$$

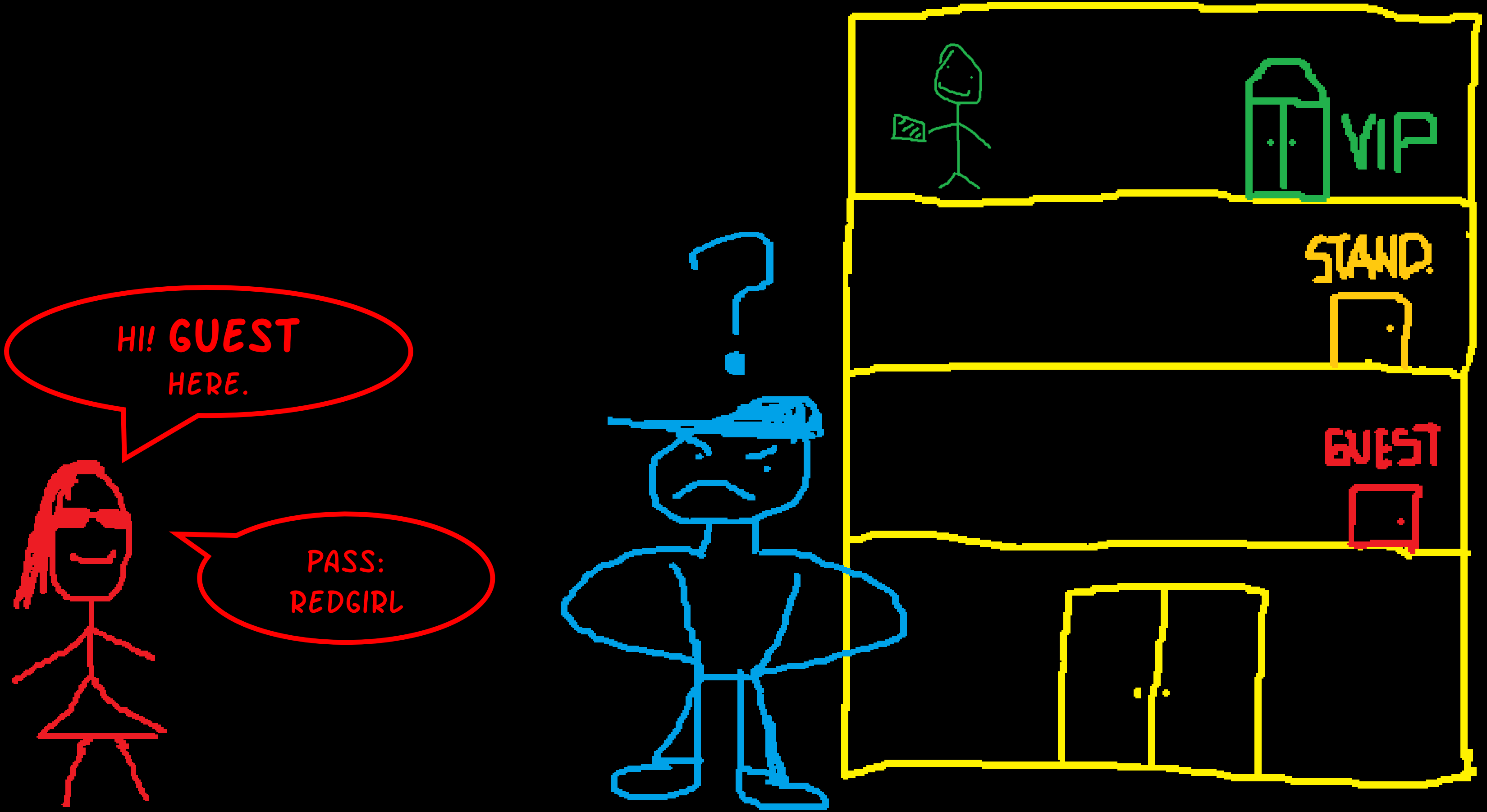
What info is retrieved from the SAM?



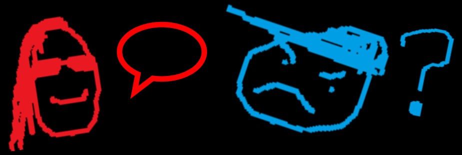
Name	Type
(Default)	REG_SZ
F	REG_BINARY
ForcePas...	REG_BINARY
Supplem...	REG_BINARY
V	REG_BINARY

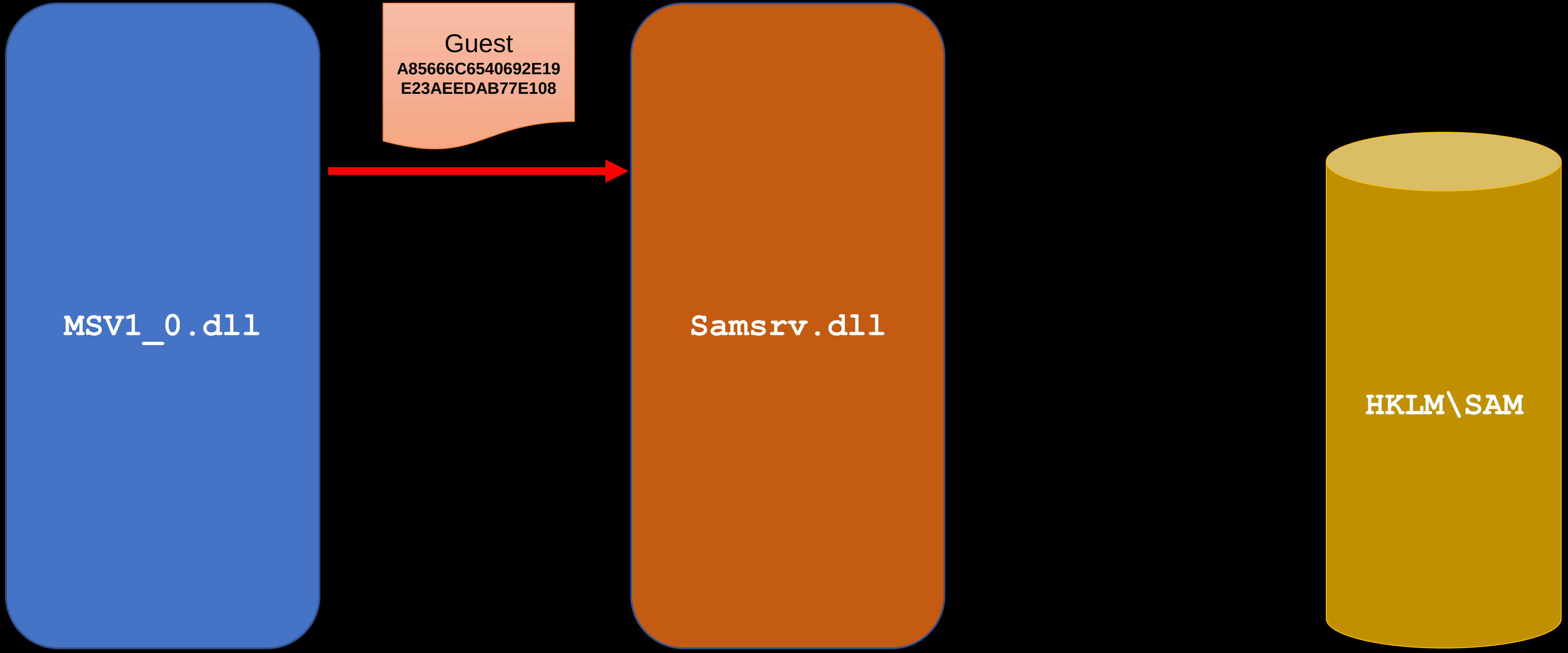
Password's Hash.
Account status (Active: Y/N).
Some account restrictions.
A copy of the user's RID.

Login as Guest

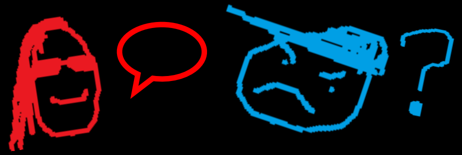


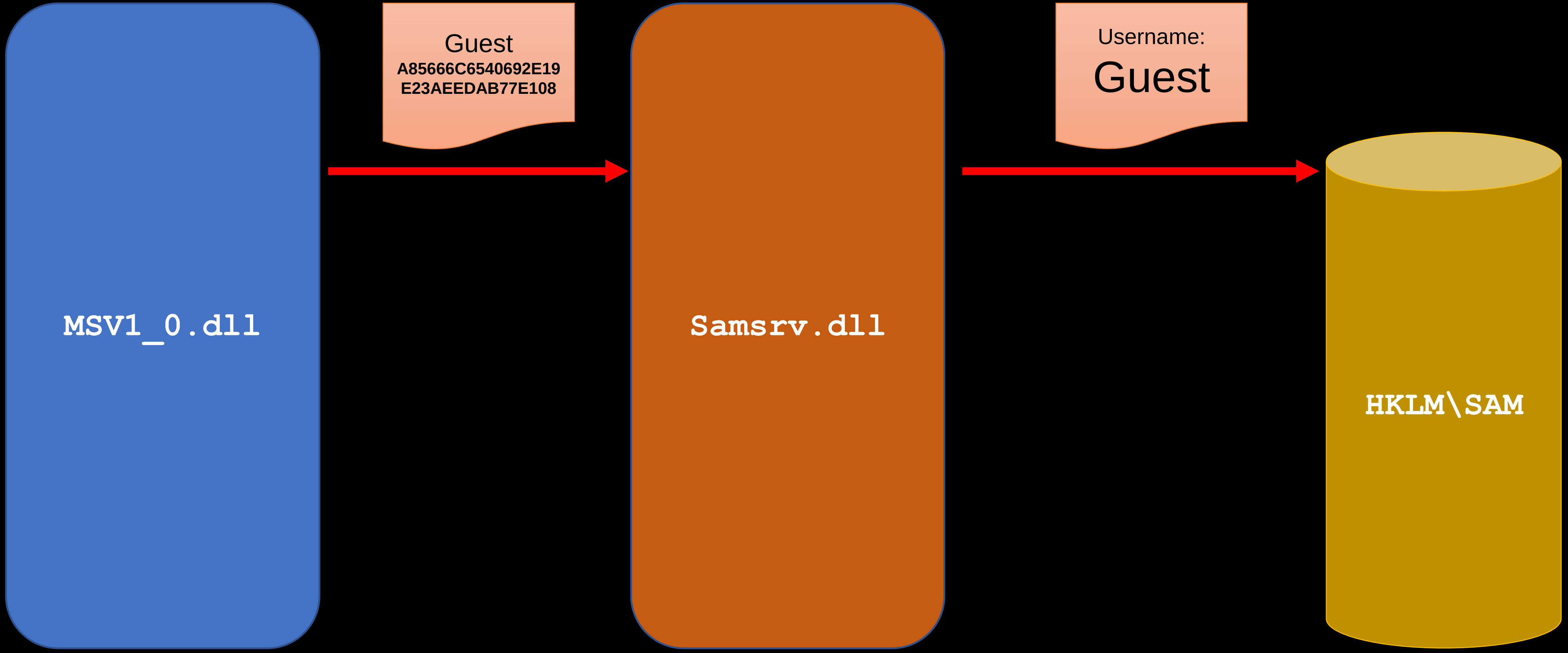
Login as Guest



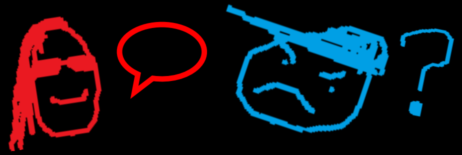


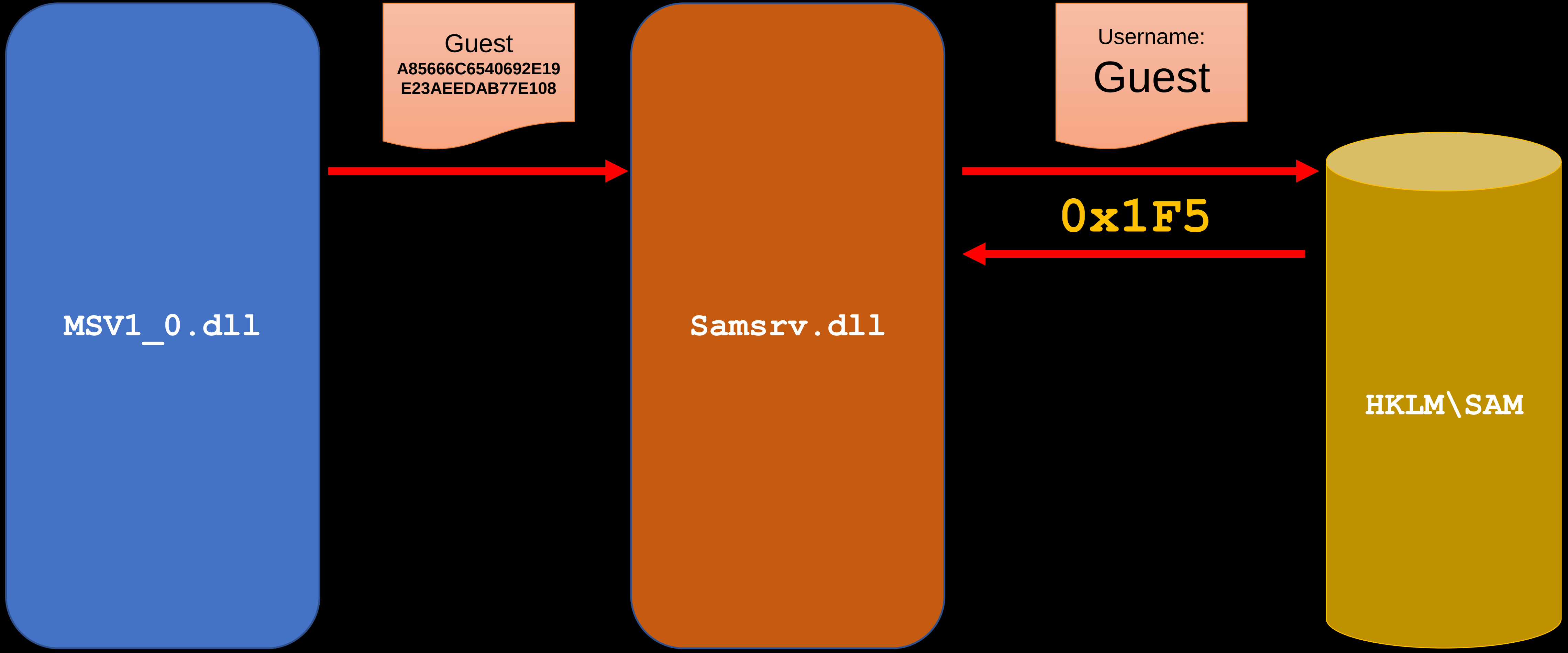
Login as Guest



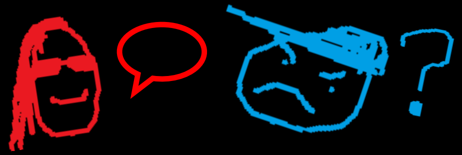


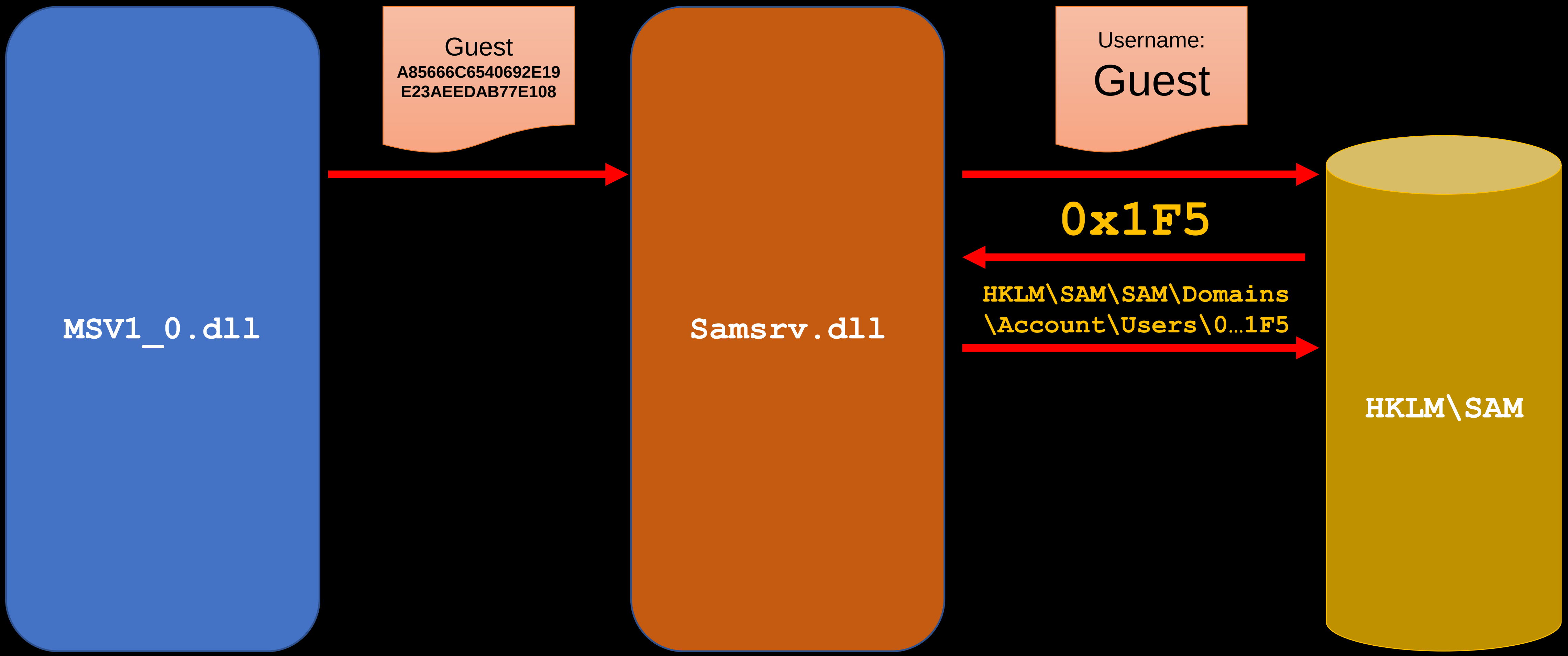
Login as Guest



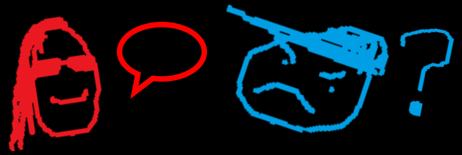


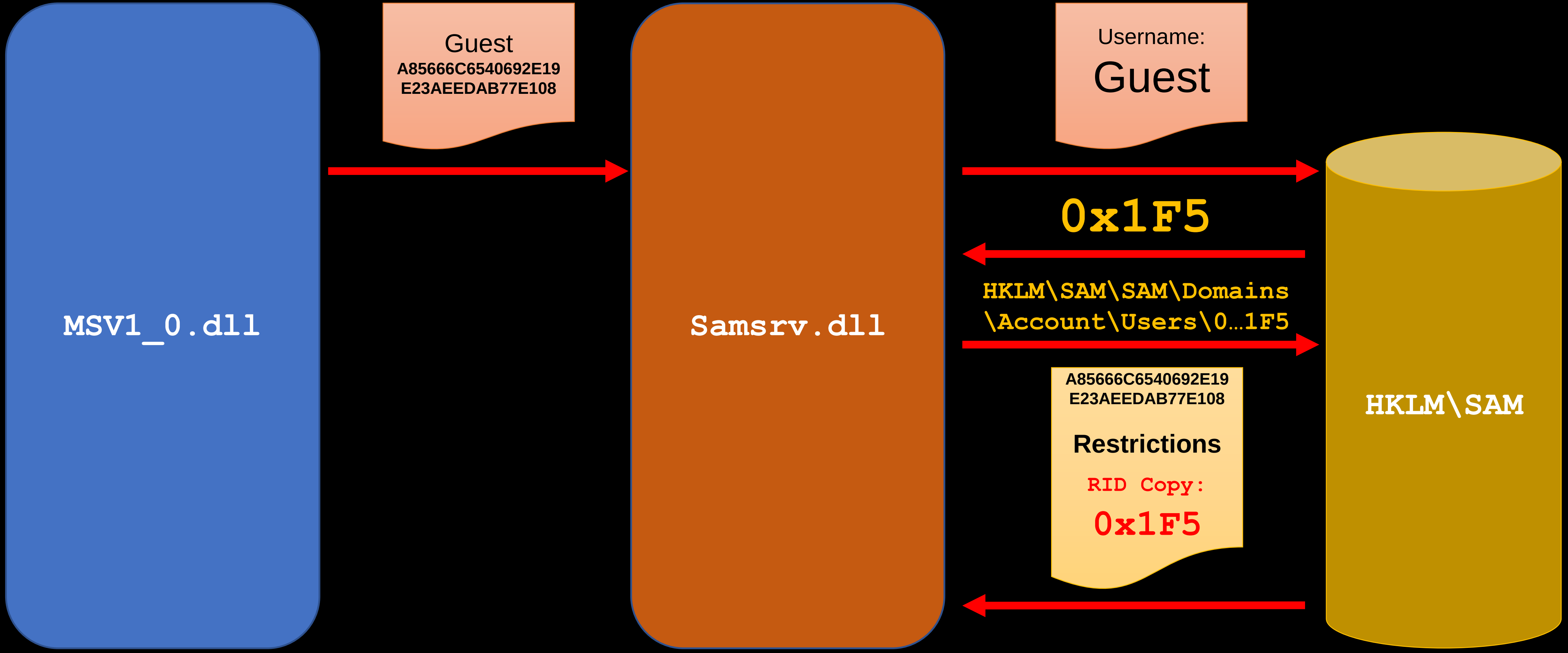
Login as Guest



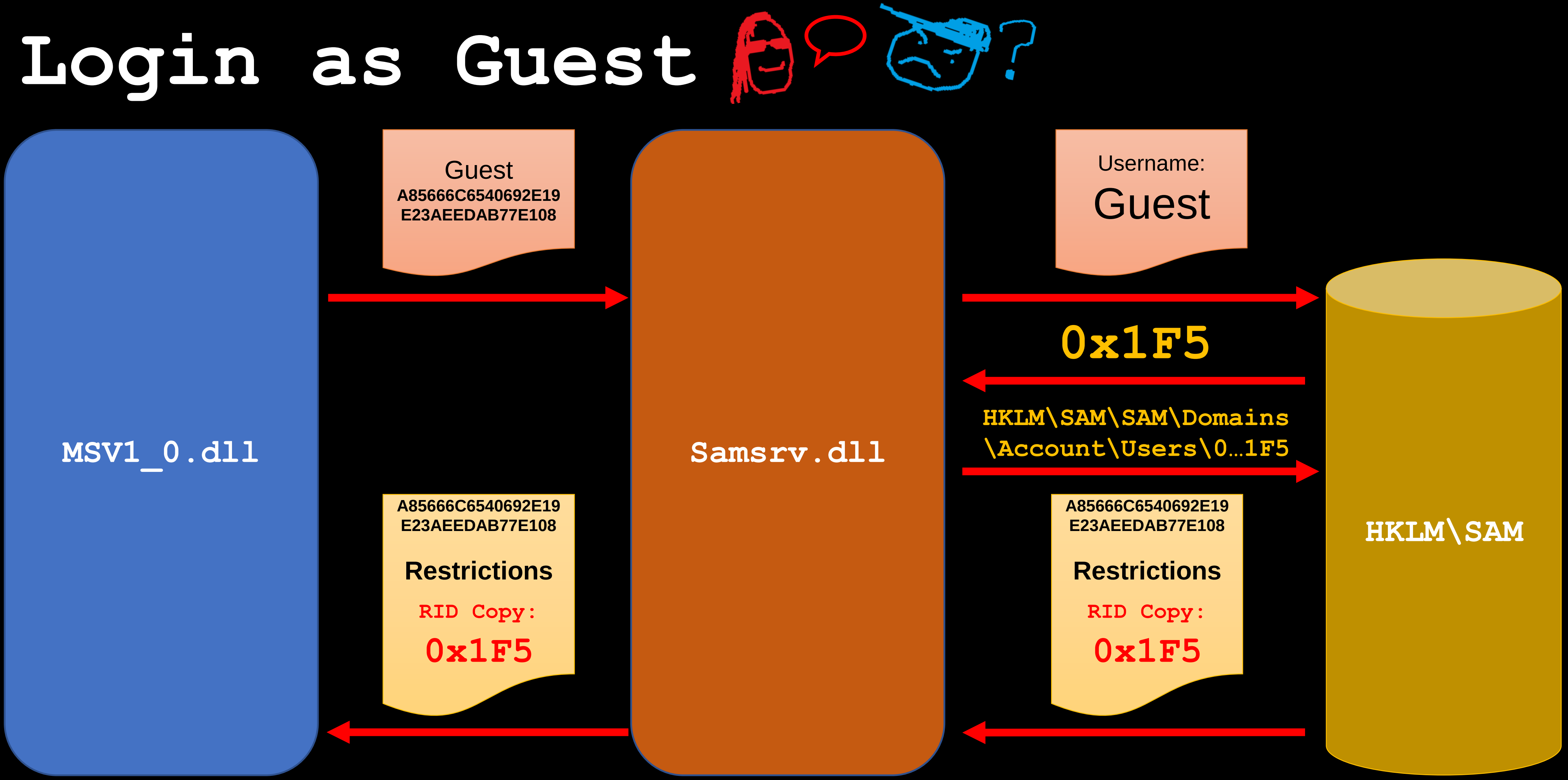


Login as Guest





Login as Guest

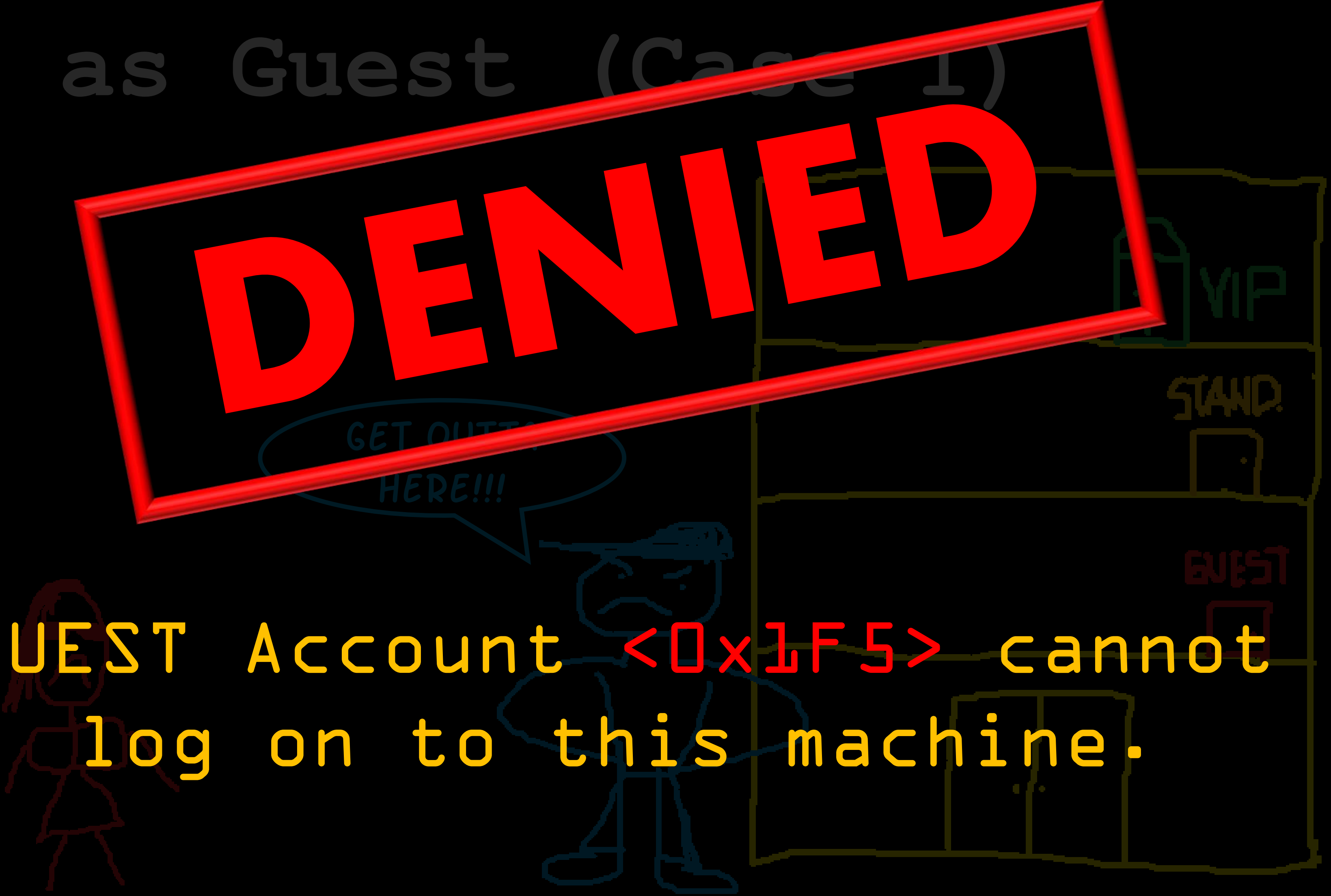


Login as Guest (Case 1)

VIP
STAND.
GUEST
GET OUTTA HERE!!!

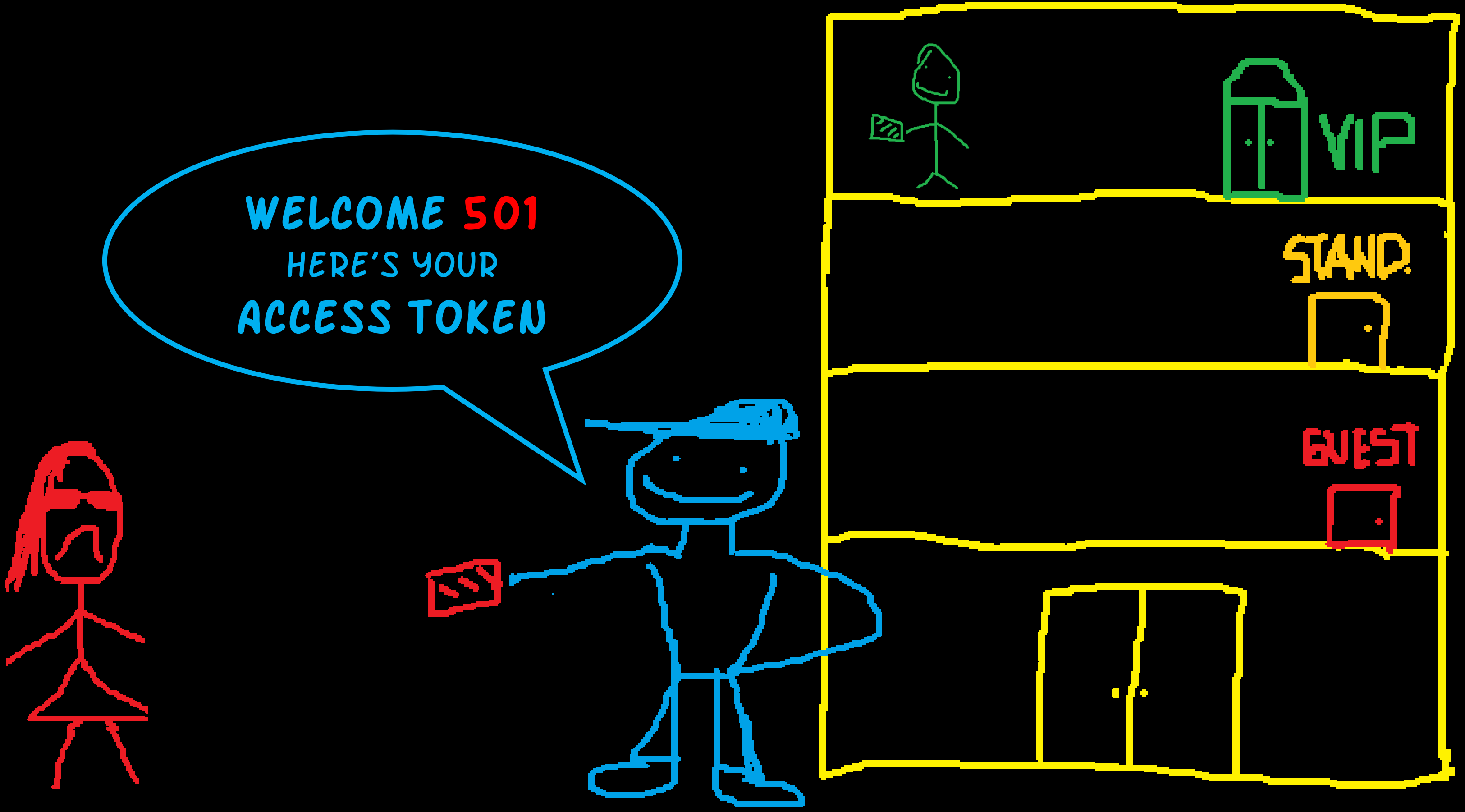
RID HIJACKING

Login as Guest (Case 1)



GUEST Account <0x1F5> cannot log on to this machine.

Login as Guest (Case 2)

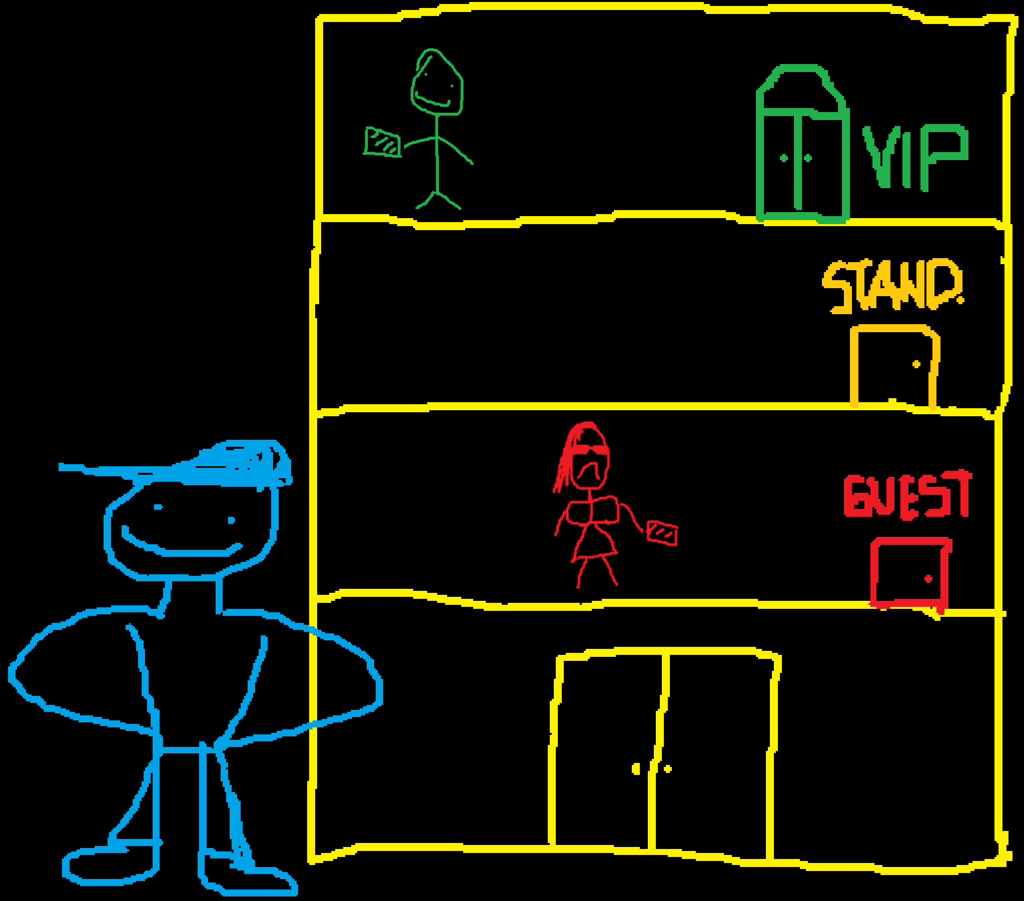


Login as Guest (Case 2)

Not

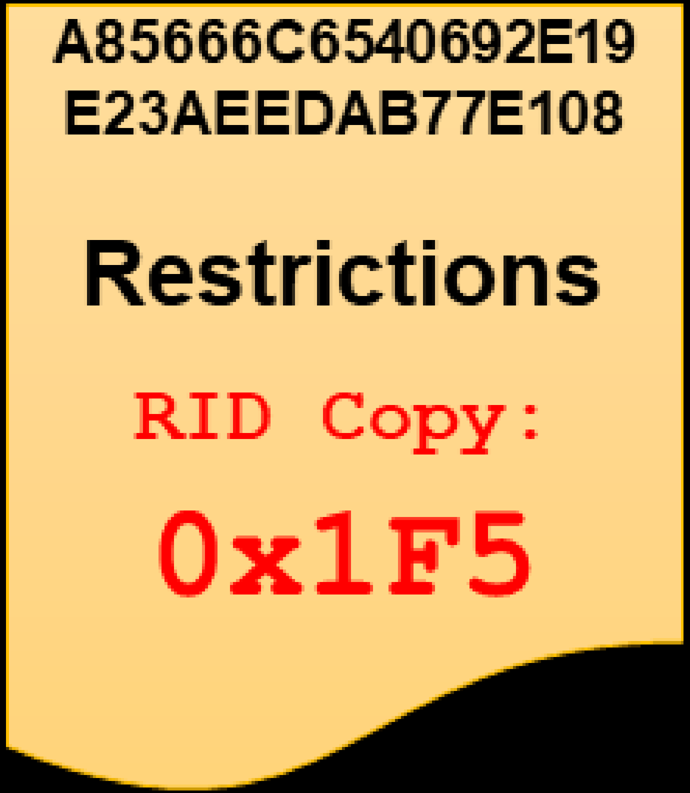
DENIED

but
could be
better!



What if…?

What would happen if the RID COPY is changed to another value?



A85666C6540692E19
E23AEEDAB77E108

Restrictions

RID Copy:

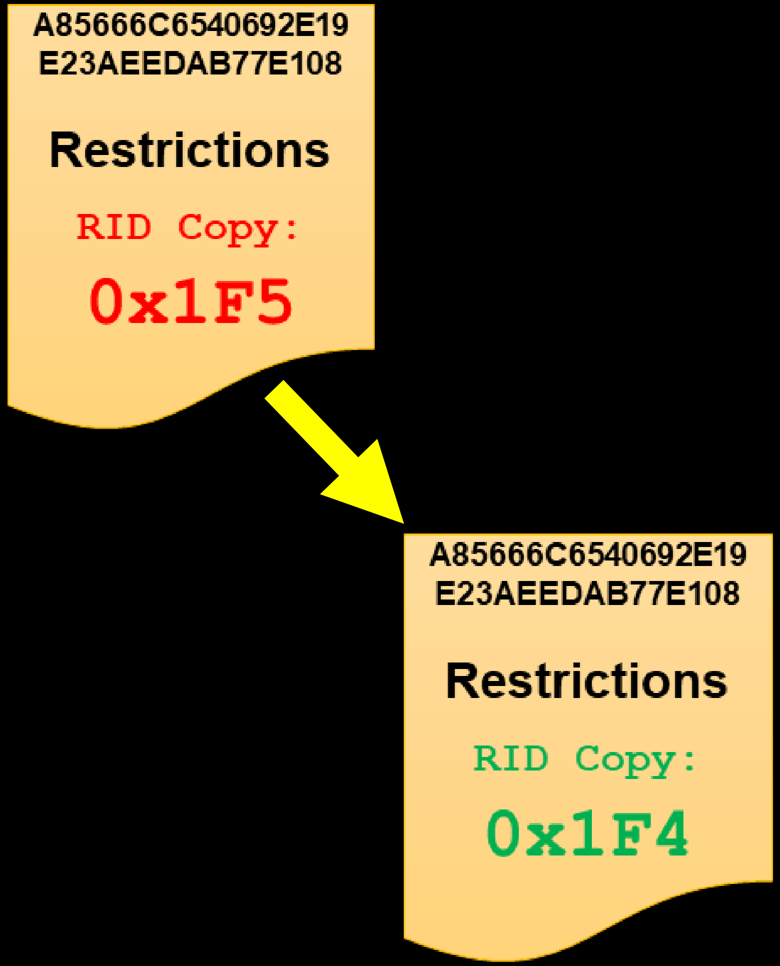
0x1F5

What if…?

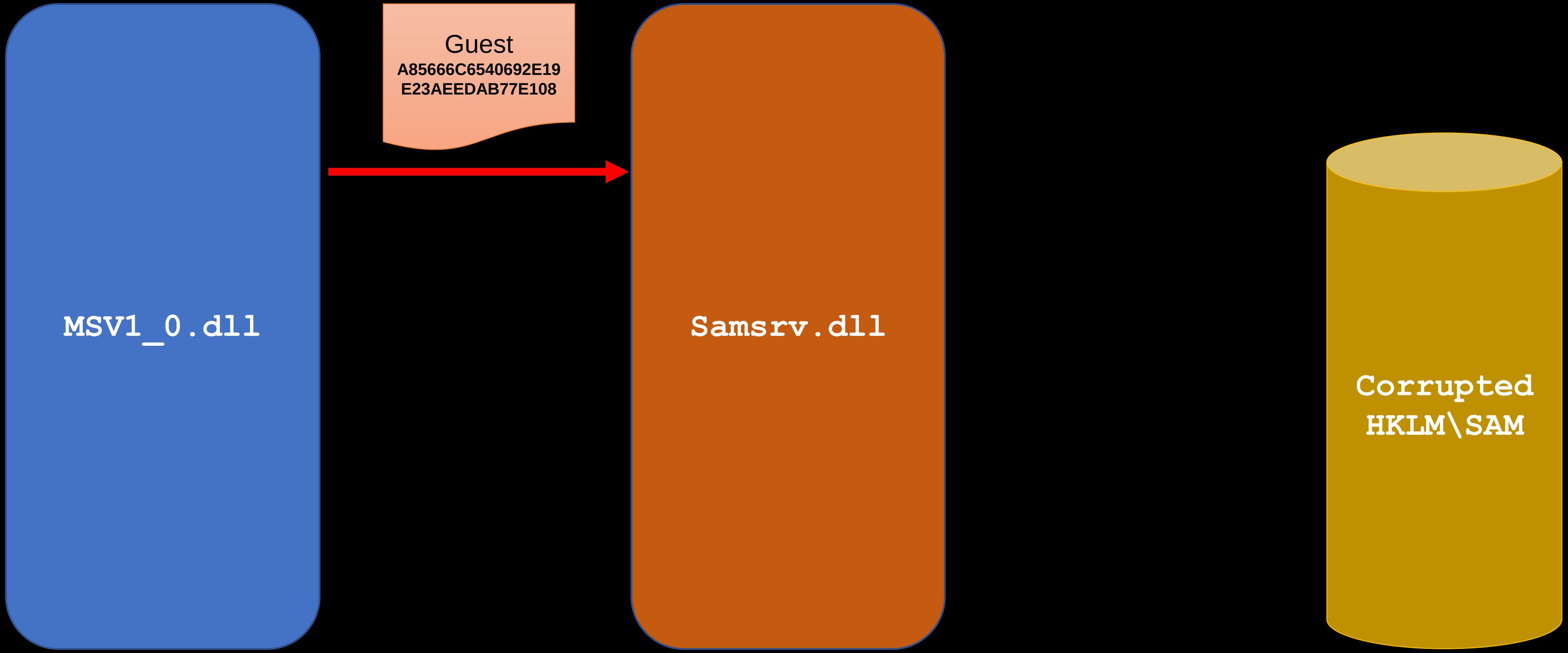
What would happen if the RID COPY is changed to another value?

RID(Administrator) = 500

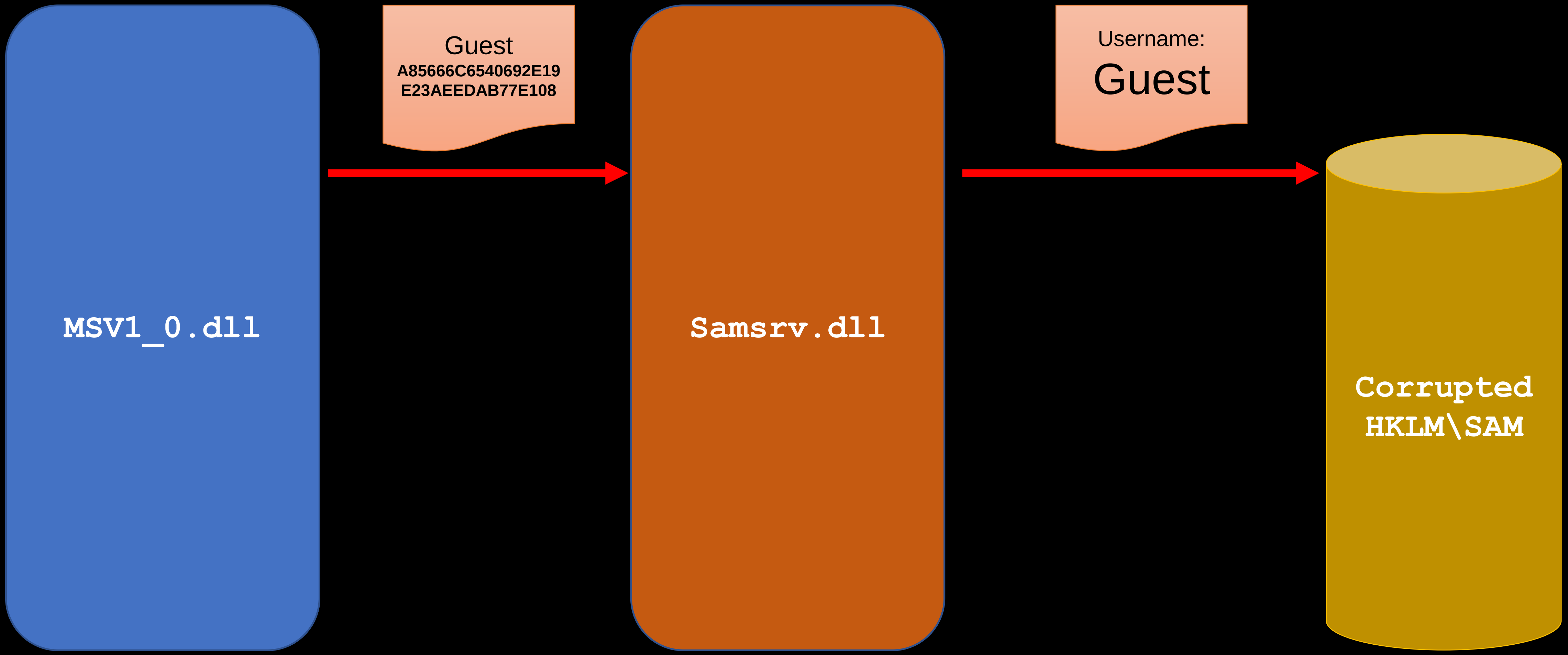
500d = 0x1F4



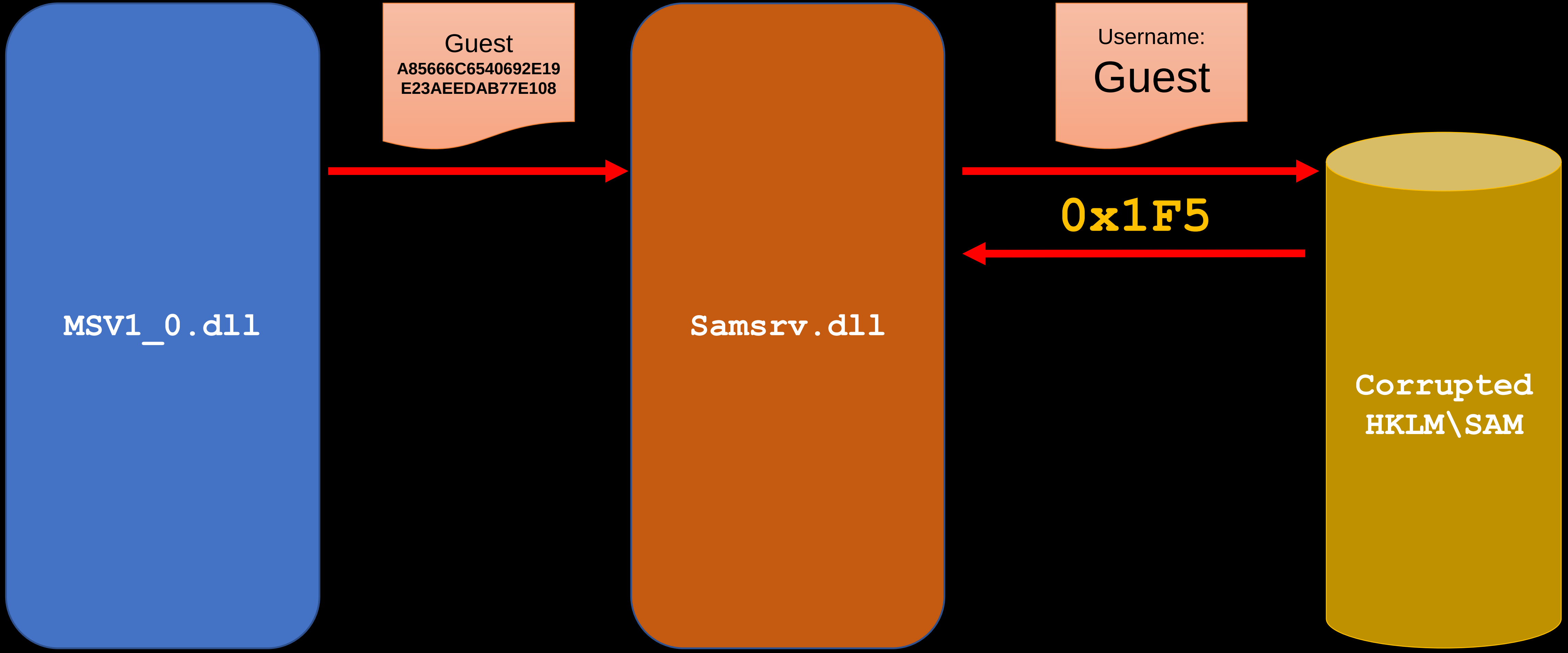
Login as Guest (the comeback)



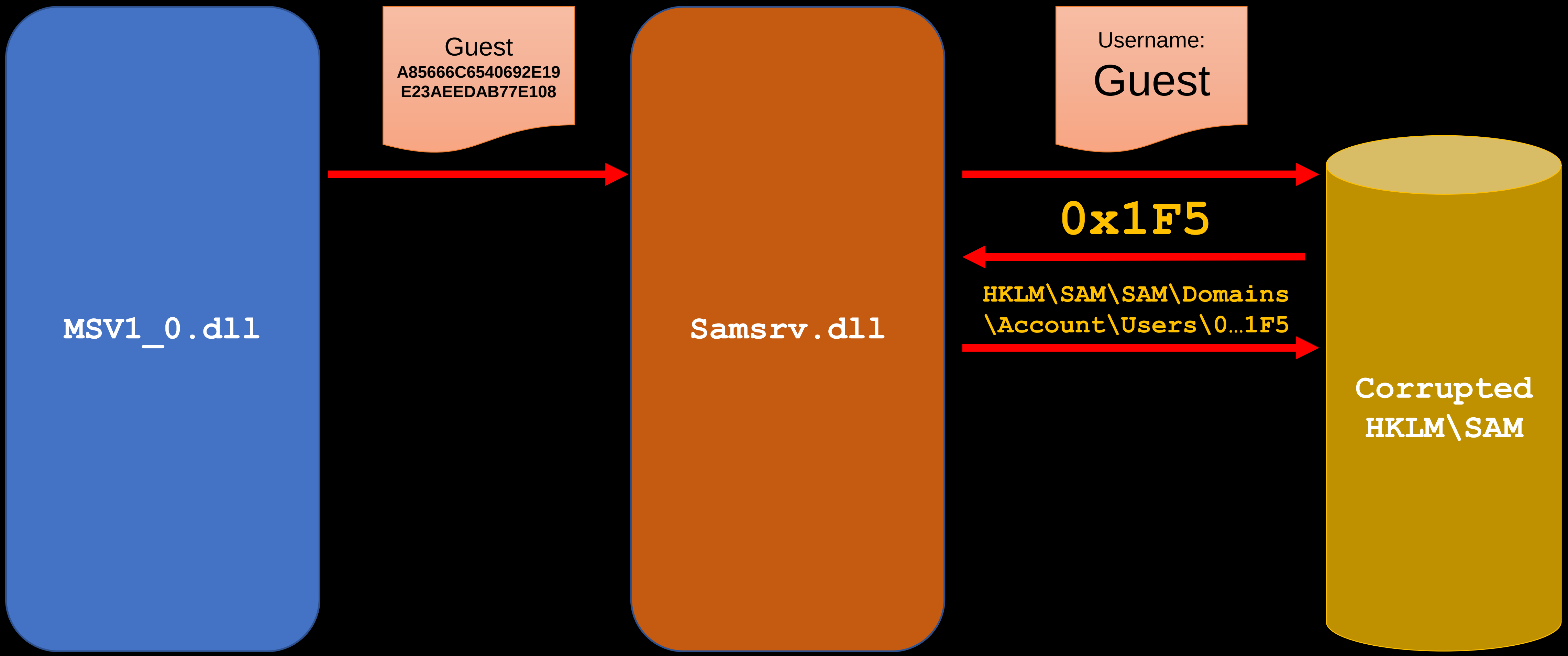
Login as Guest (the comeback)



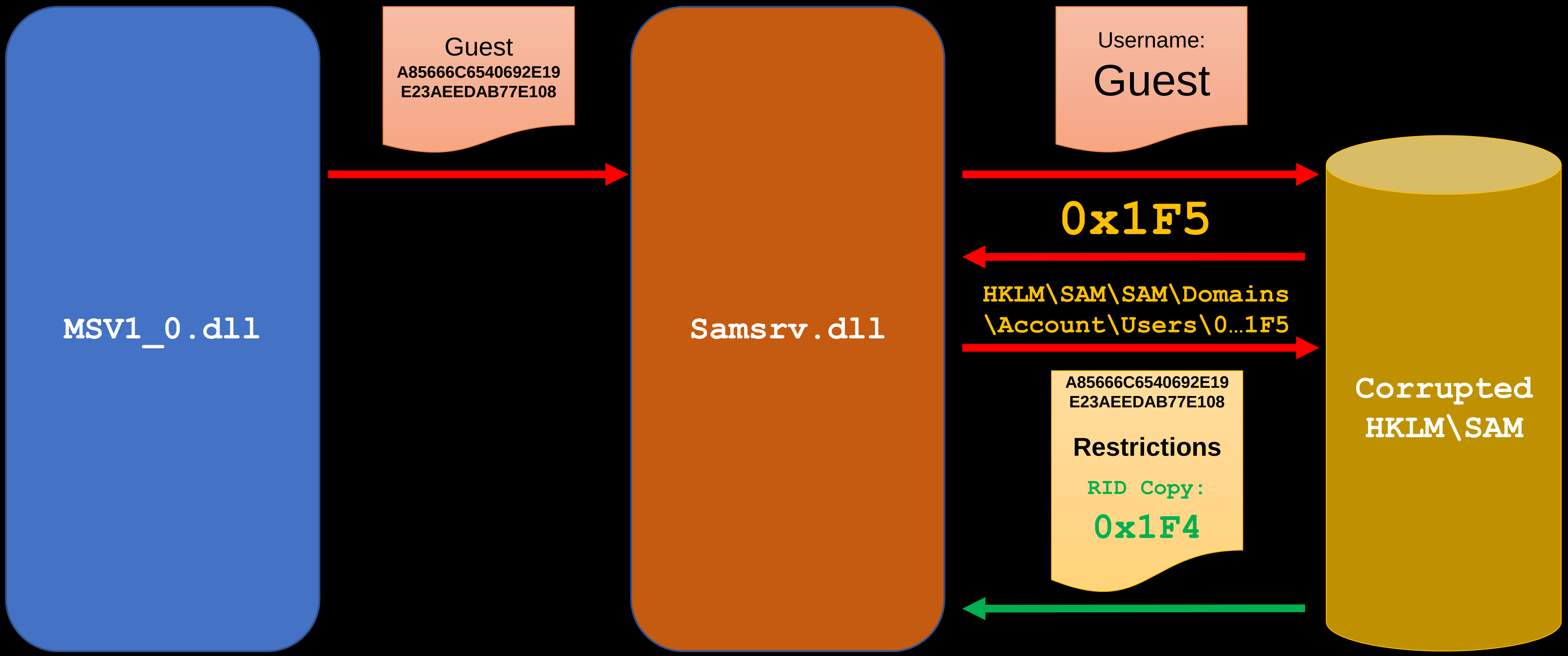
Login as Guest (the comeback)



Login as Guest (the comeback)

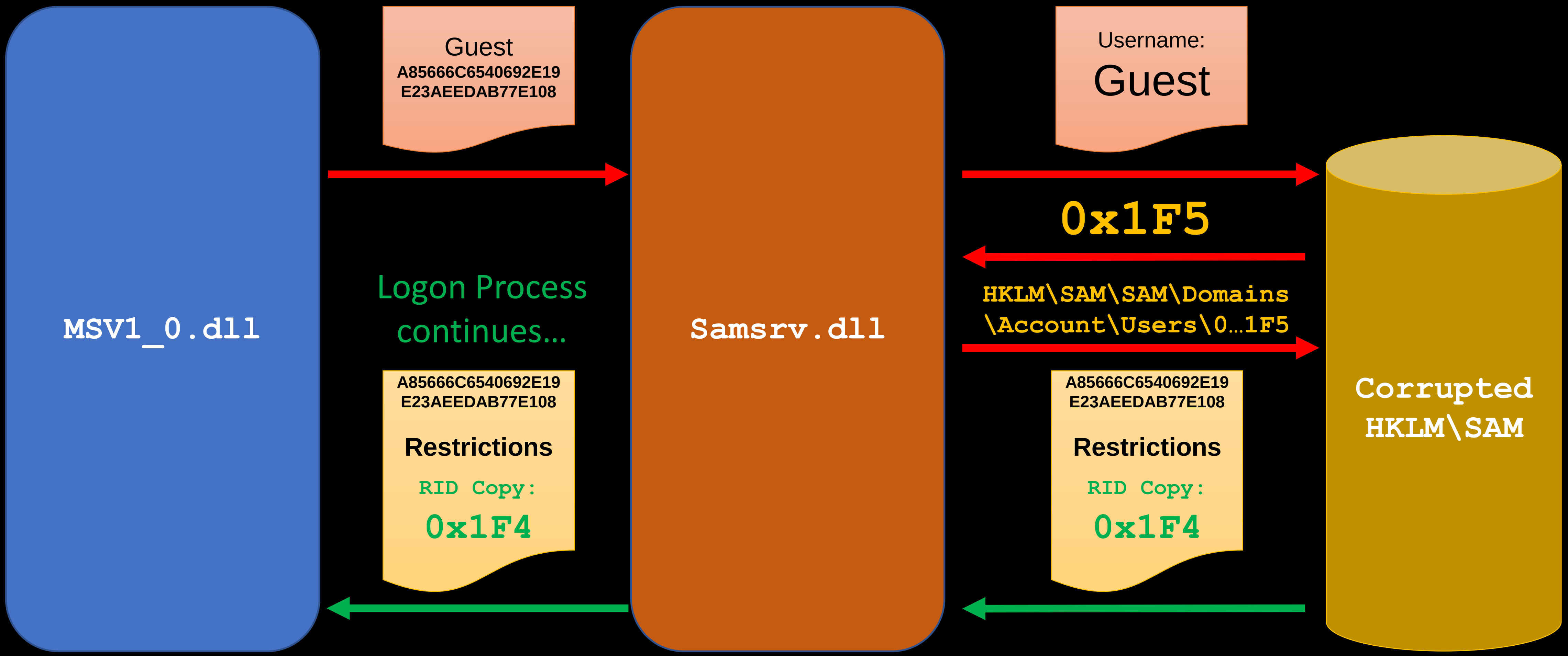


Login as Guest (the comeback)



RID HIJACKING

Login as Guest (the comeback)



Login as Guest (the comeback)

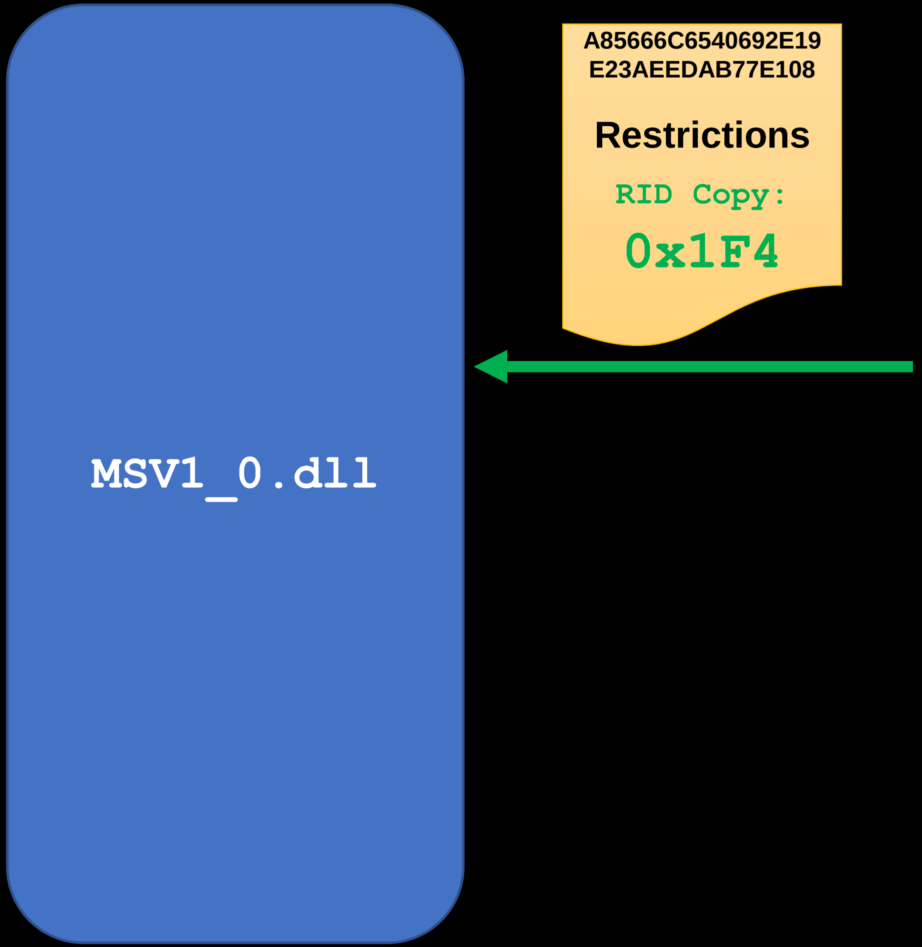
MSV1_0 checks the account restrictions provided from SAMSRV.

If allowed, then compares:

SAMSRV response password hash

VS

User entered hashed password



Login as Guest (the comeback)

MSV1_0 checks the account restrictions provided from SAMSRV.

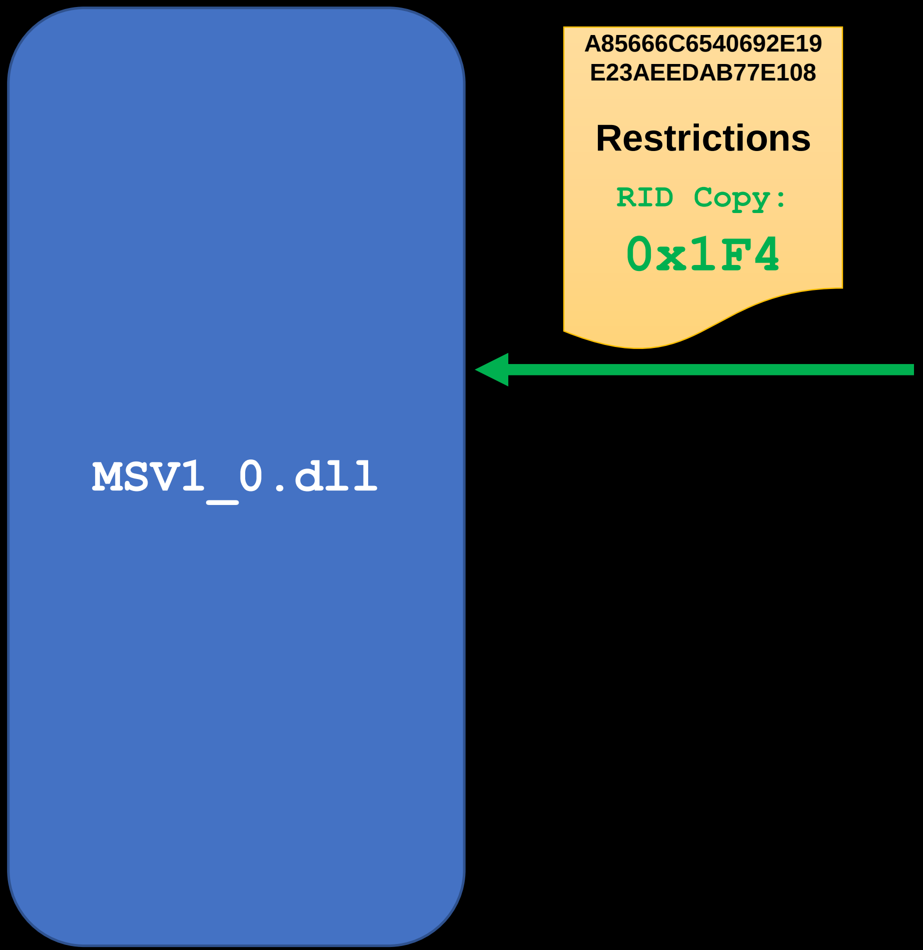
If allowed, then compares:

SAMSRV response password hash

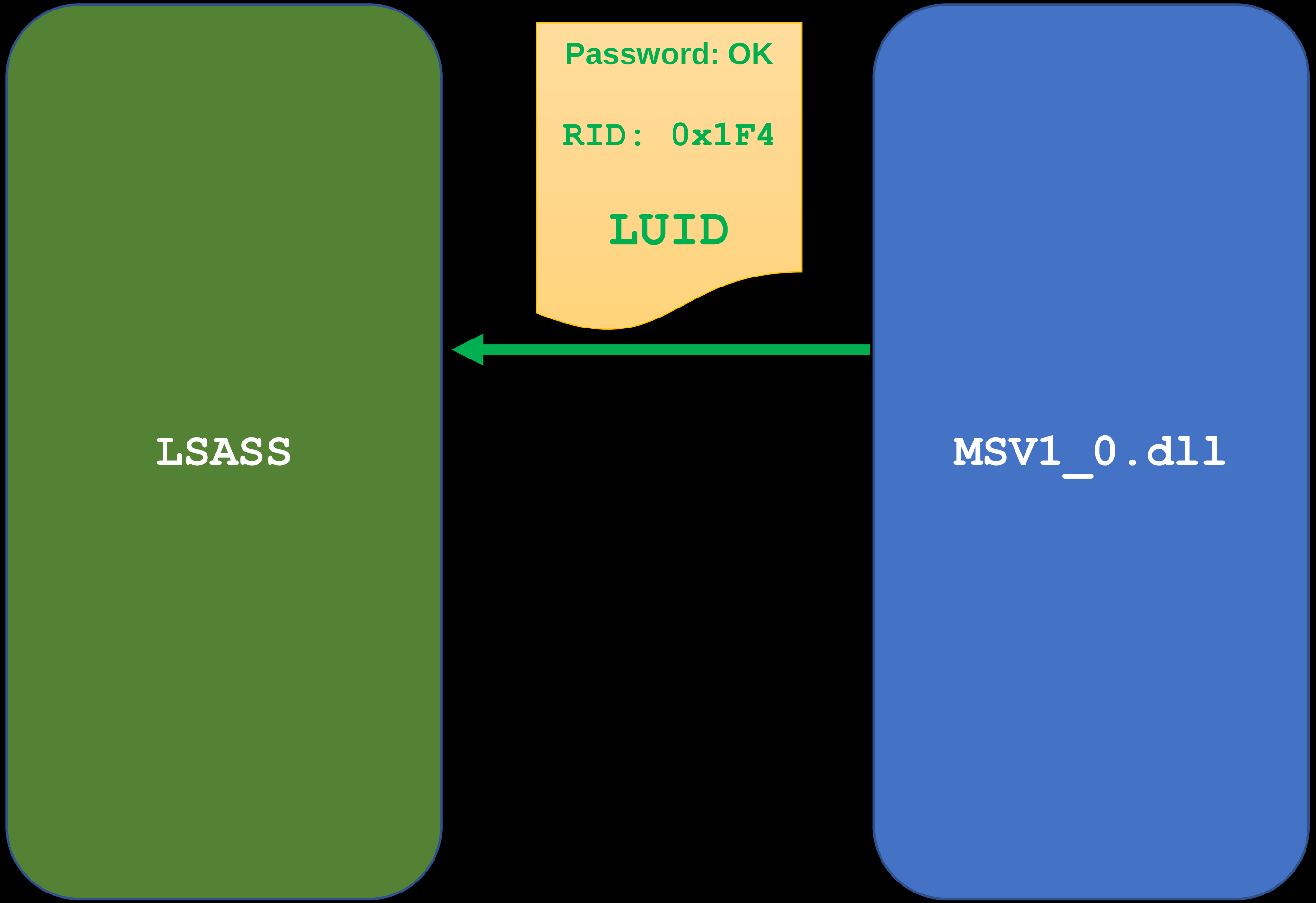
VS

User entered hashed password

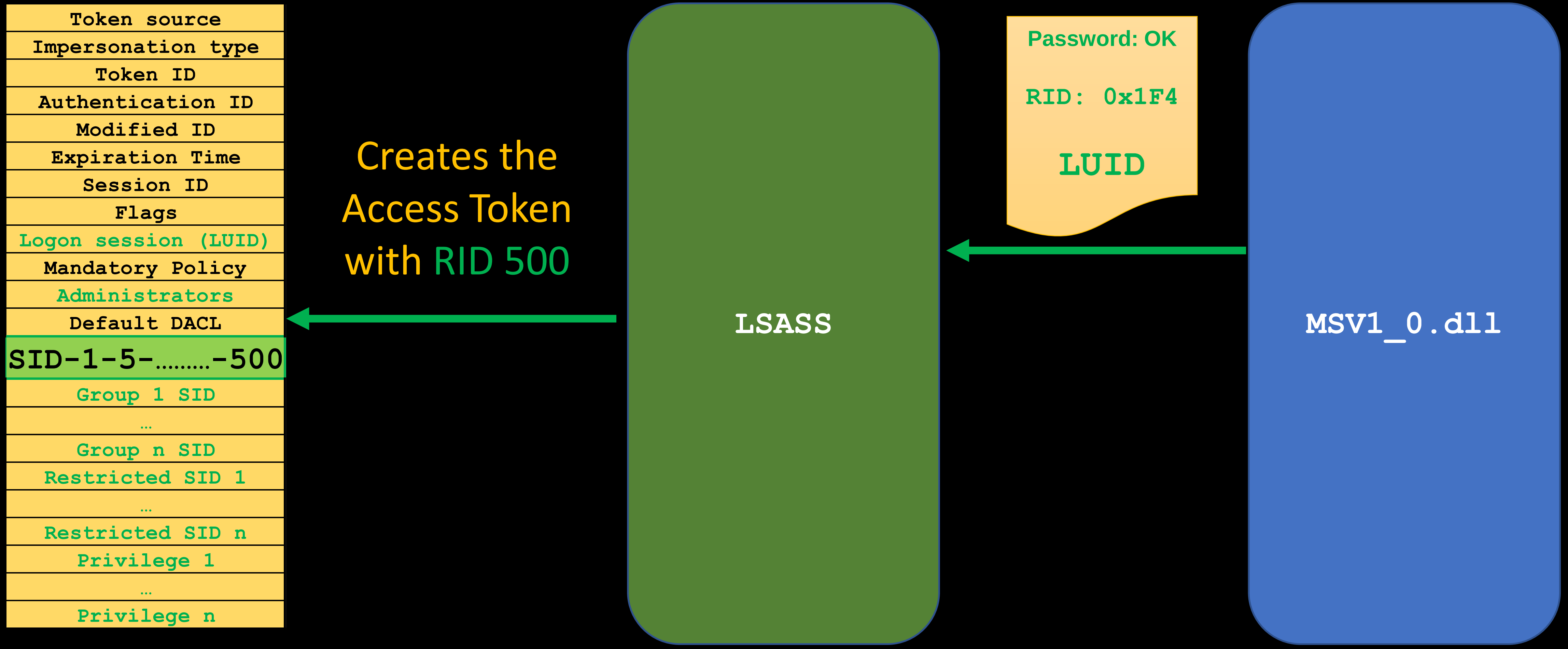
Hash will be the same



Login as Guest (the comeback)



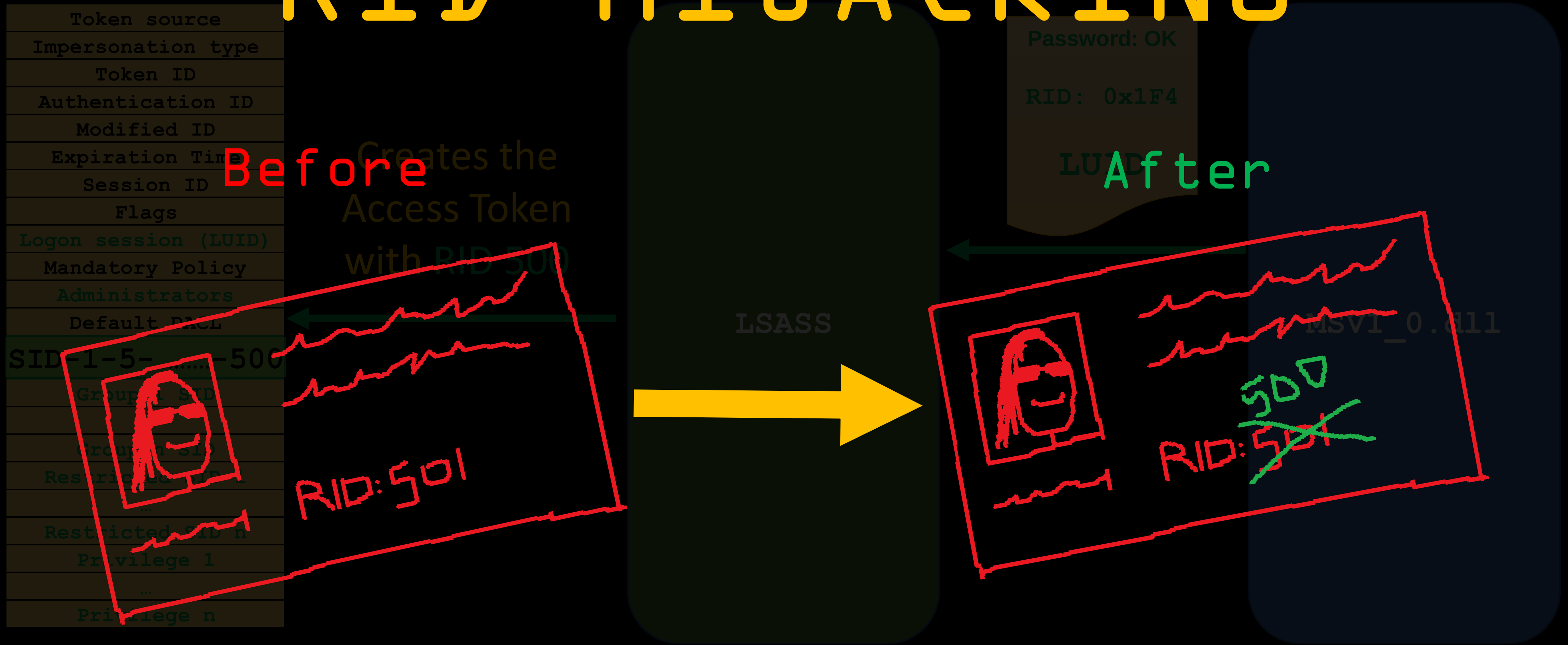
Login as Guest (the comeback)



RID HIJACKING

Login as Guest (the comeback)

RID HIJACKING



ITSECX @r4wd3r csl.com.co

Login as Guest

WELCOME GUEST
HERE'S YOUR
ACCESS TOKEN
VIP
STAND.
GUEST

Login as Guest

ACCESS GRANTED
TOKEN
500
RID: ~~501~~
VIP
STAND.
GUEST

SECURITY ISSUES

0x01. SAMSRV does not check if the RID associated with the user is consistent to the RID COPY.

SECURITY ISSUES

0x01. SAMSRV does not check if the RID associated with the user is consistent to the RID COPY.

0x02. LSASS does not corroborate the RID with the username before creating the access token.

SECURITY ISSUES

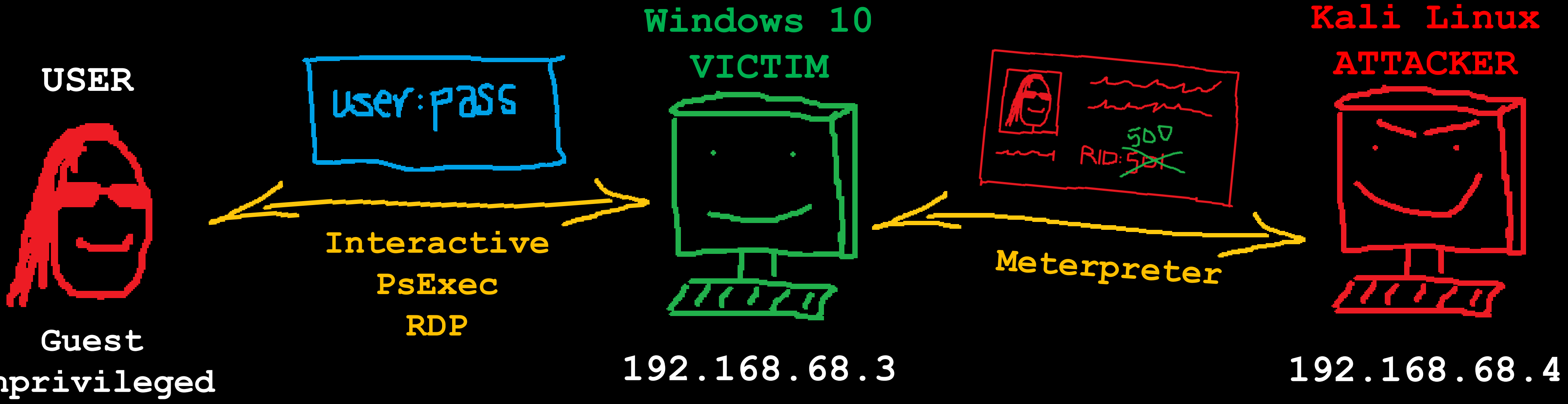
0x01. SAMSRV does not check if the RID associated with the user is consistent to the RID COPY.

0x02. LSASS does not corroborate the RID with the username before creating the access token.

0x03. LSASS never looks for RID inconsistencies during the user's session.

Agenda

Demonstration



Agenda

Conclusions

References

1. http://csl.com.co/rid-hijacking/

2. Russinovich, Mark. Solomon, David A. Ionescu, Alex. "Windows Internals". 6th Edition.

3. Scambray, Joel. McClure, Stuart. "Hacking Exposed: Windows Security Secrets & Solutions". 3rd Edition.

4. https://technet.microsoft.com/pt-pt/library/cc780332(v=ws.10).aspx

5. https://docs.microsoft.com/en-us/windows-server/security/windows-authentication/credentials-processes-in-windows-authentication