





# RID HIJACKING:

Maintaining Access on Windows Machines.

Sebastián Castro

 sebastian.castro@csl.com.co

 @r4wd3r

 r4wd3r



CYBERSECURITY CONVENTION

Rome, Italy

2018

romhack.io

```
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

- 0x01. Exposing the RID Hijacking Attack.
- 0x02. A Windows Logon Story.
- 0x03. Hijacking the RID.
- 0x04. Demo.
- 0x05. Conclusions.

# Agenda

0x01. Exposing the RID Hijacking Attack.

0x02. A Windows Logon Story.

0x03. Hijacking the RID.

0x04. Demo.

0x05. Conclusions.

# 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

RID HIJACKING



# 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.

# How does it look like?

```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Guest>whoami
rh-demo\guest 1

C:\Users\Guest>net user Guest
User name           Guest
Full Name           Guest
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. 2

C:\Users\Guest>echo "hacked" > c:\Windows\System32\rindhijack.txt
C:\Users\Guest>type c:\Windows\System32\rindhijack.txt
"hacked" 3
```

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\rindhijack.txt
C:\Users\Guest>type c:\Windows\System32\rindhijack.txt
"hacked" 3
```

# Agenda

0x01. Exposing the RID Hijacking Attack.

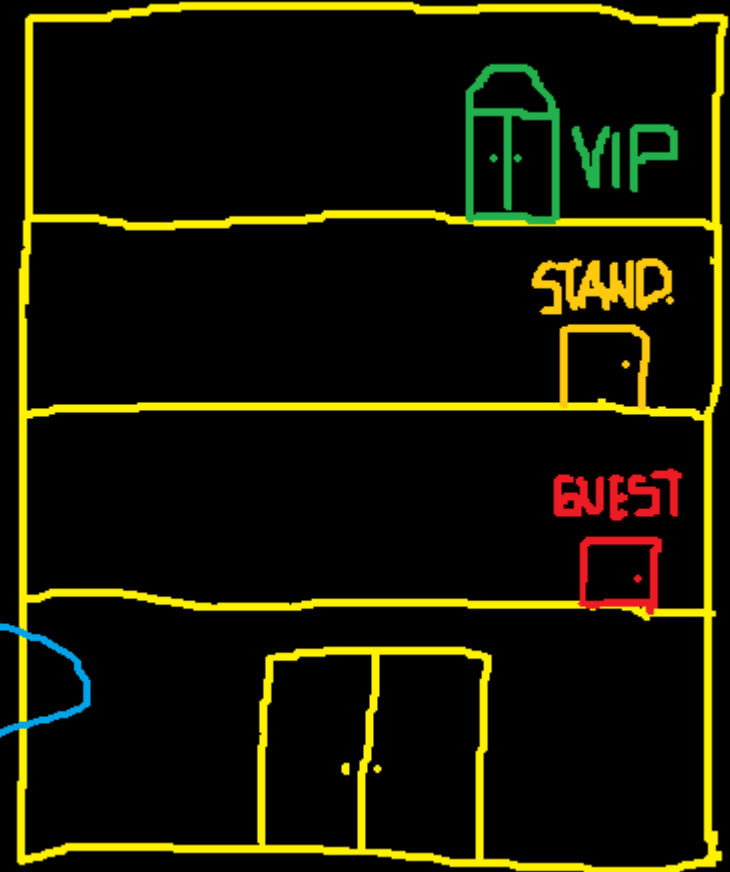
0x02. A Windows Logon Story.

0x03. Hijacking the RID.

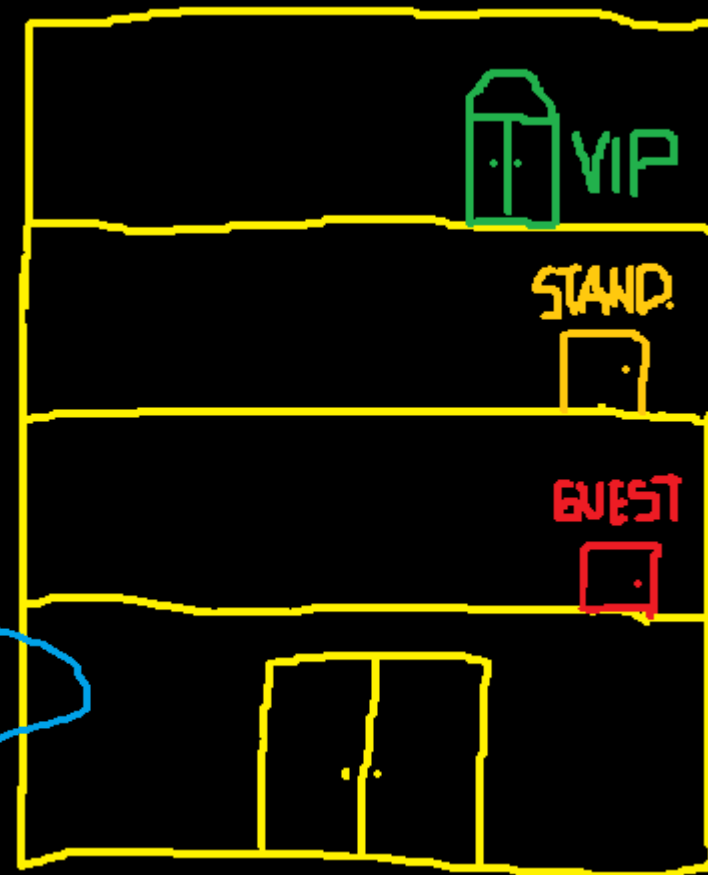
0x04. Demo.

0x05. Conclusions.

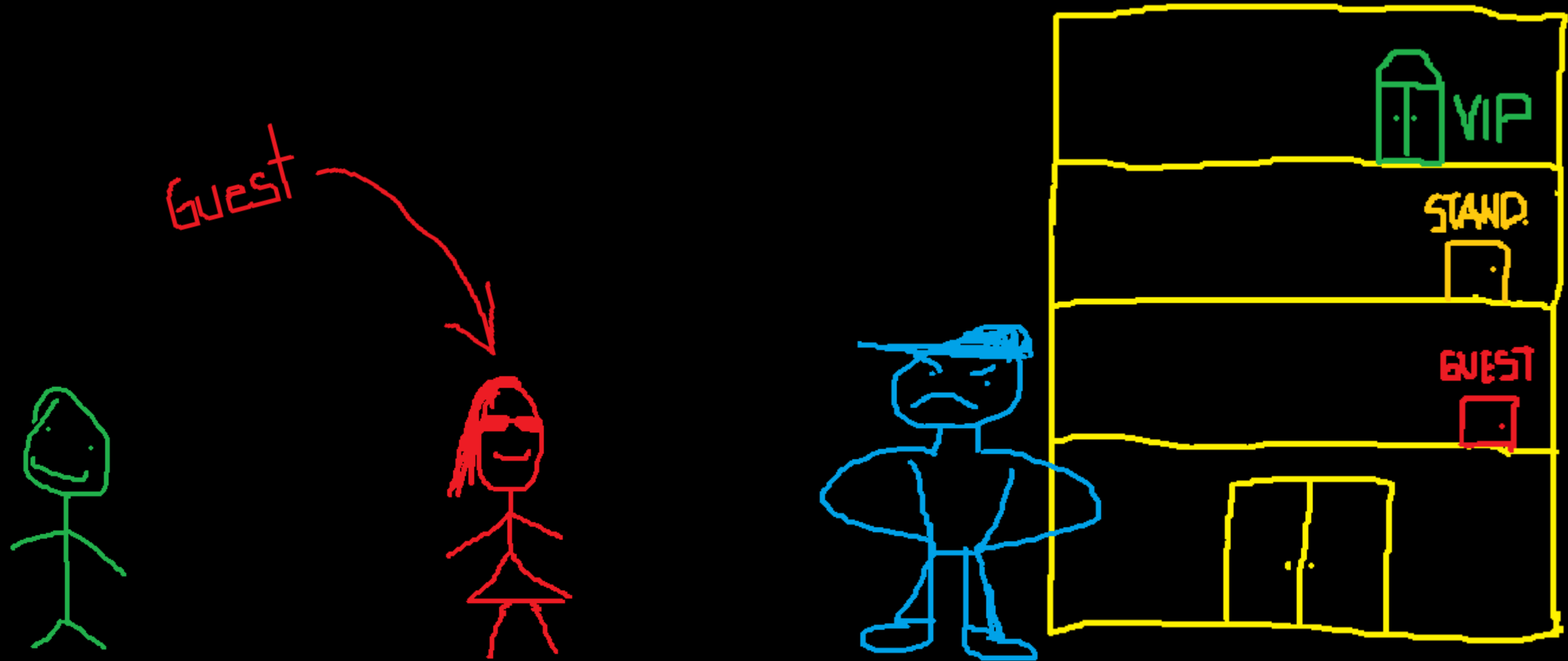
# A Windows Logon Story...



# A Windows Logon Story...



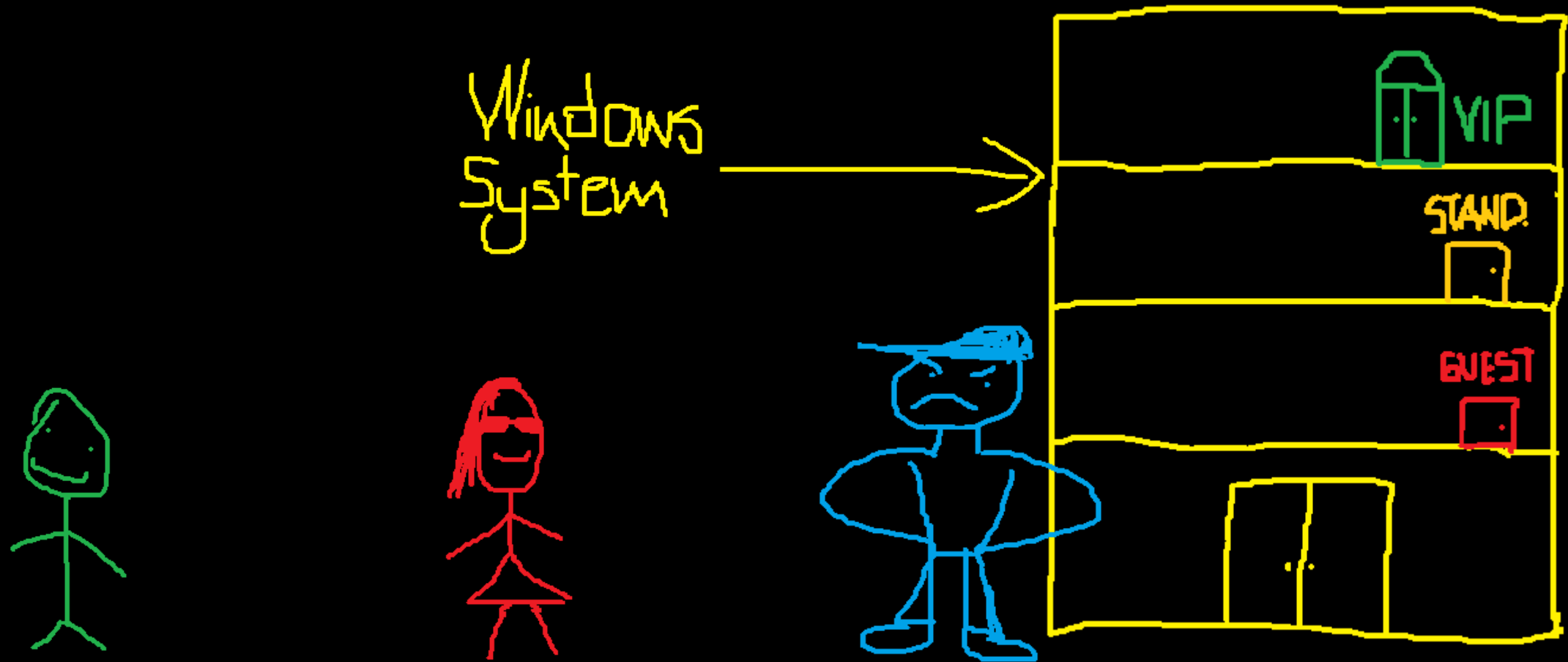
# A Windows Logon Story...



# A Windows Logon Story...

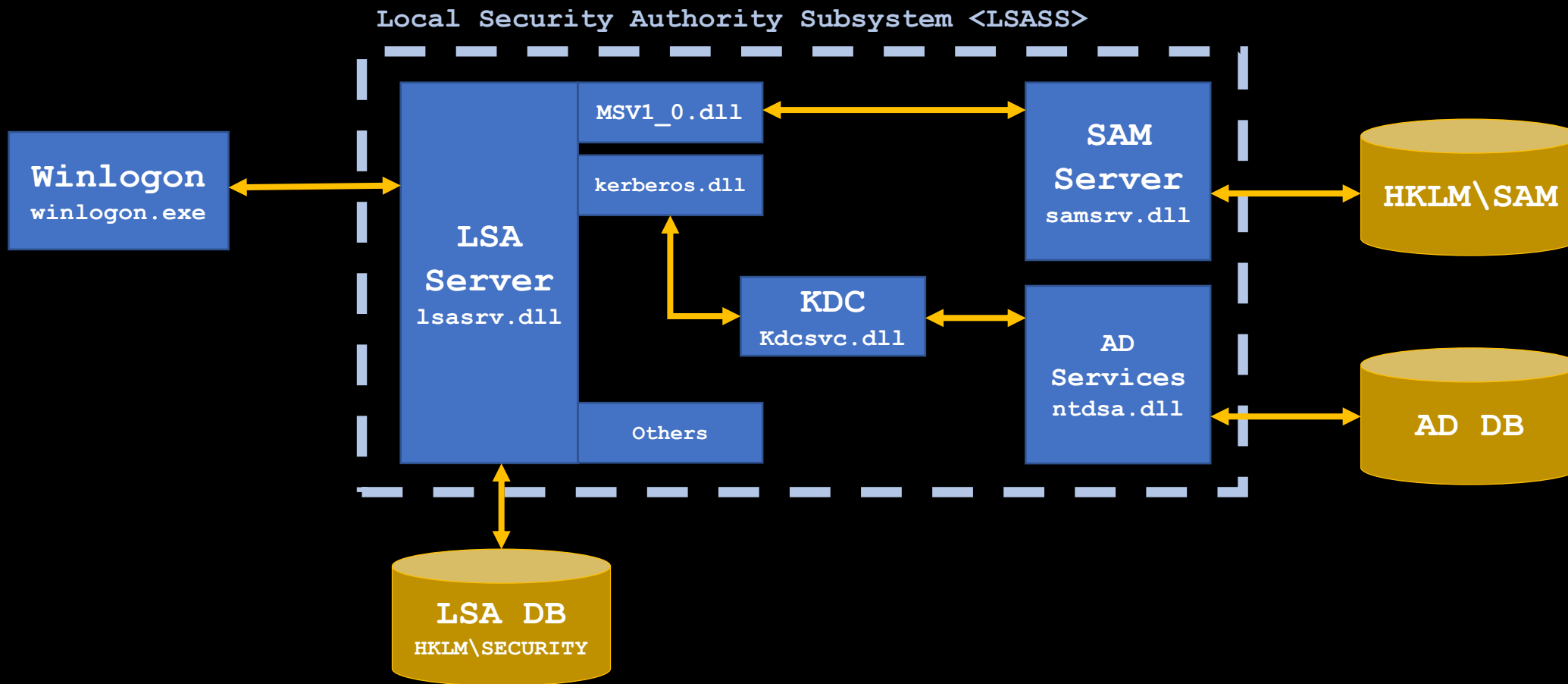


# A Windows Logon Story...





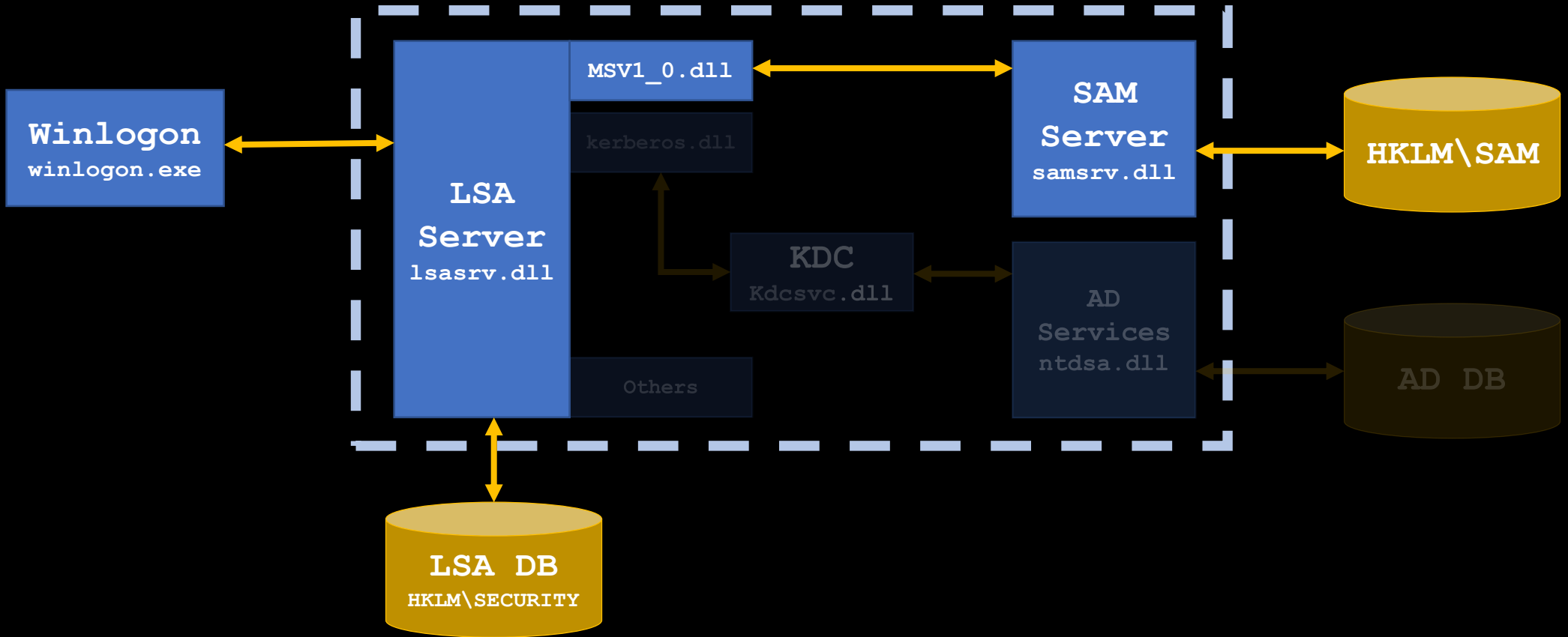
# Windows Security Architecture



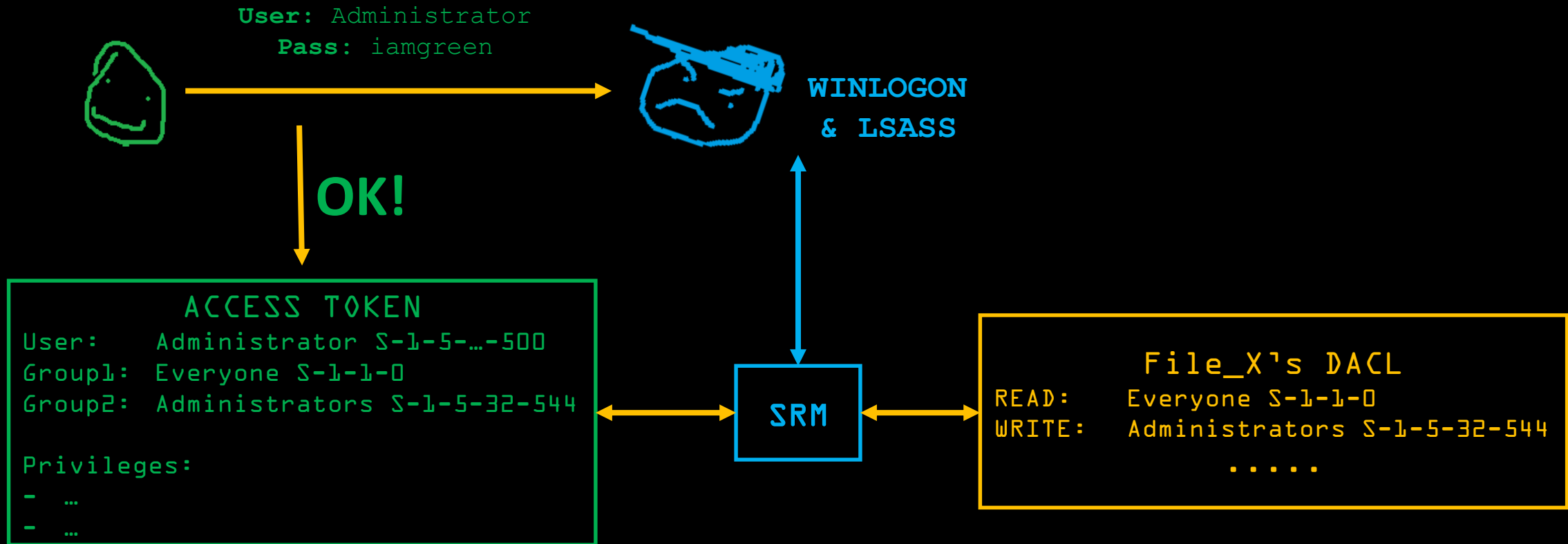
# Windows Security Architecture



Local Security Authority Subsystem <LSASS>



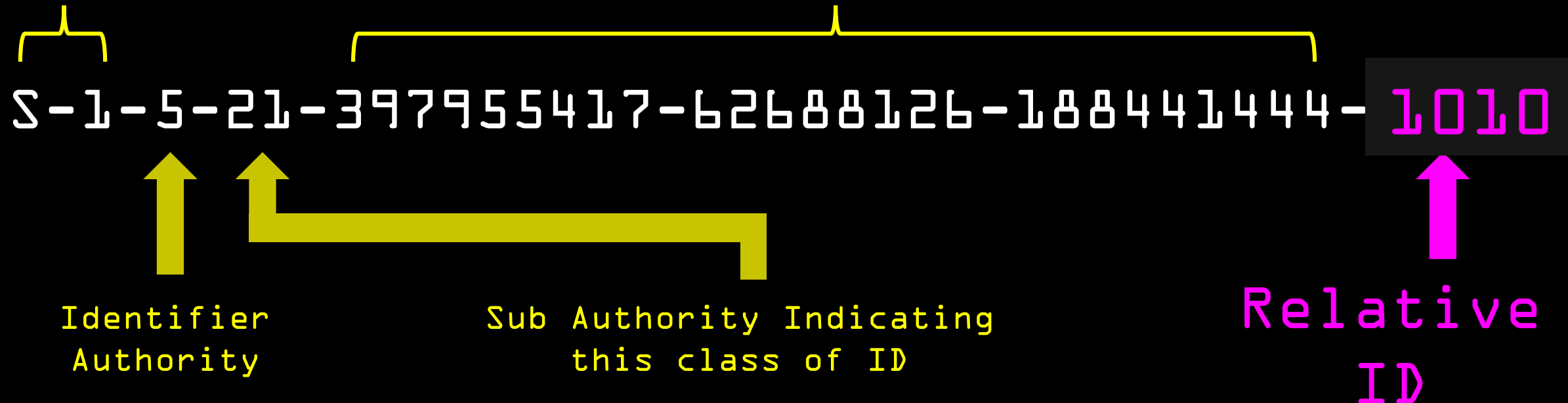
# Quick Logon Overview



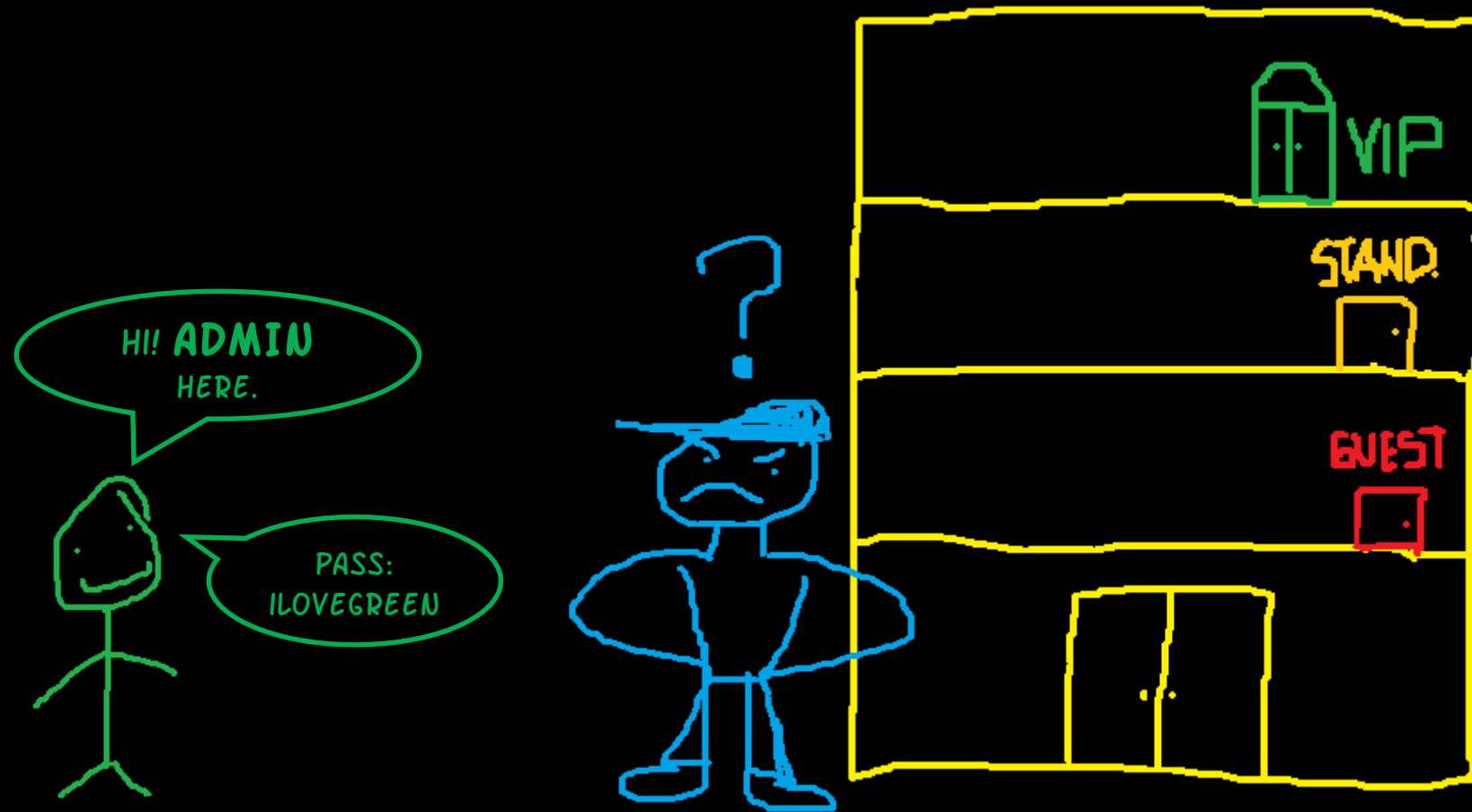
# Security Identifiers <SID>

Literal  
prefix

Three Sub Authorities for Uniqueness



# 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 Packages



List: HKLM\SYSTEM\CurrentControlSet\Control\Lsa

For interactive logons:

- <MSV1\_0.dll>: Standalone Authentication.
- <Kerberos.dll>: Domain Kerberos Authentication.



Kerberos authentication package will be ignored by now.

# 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

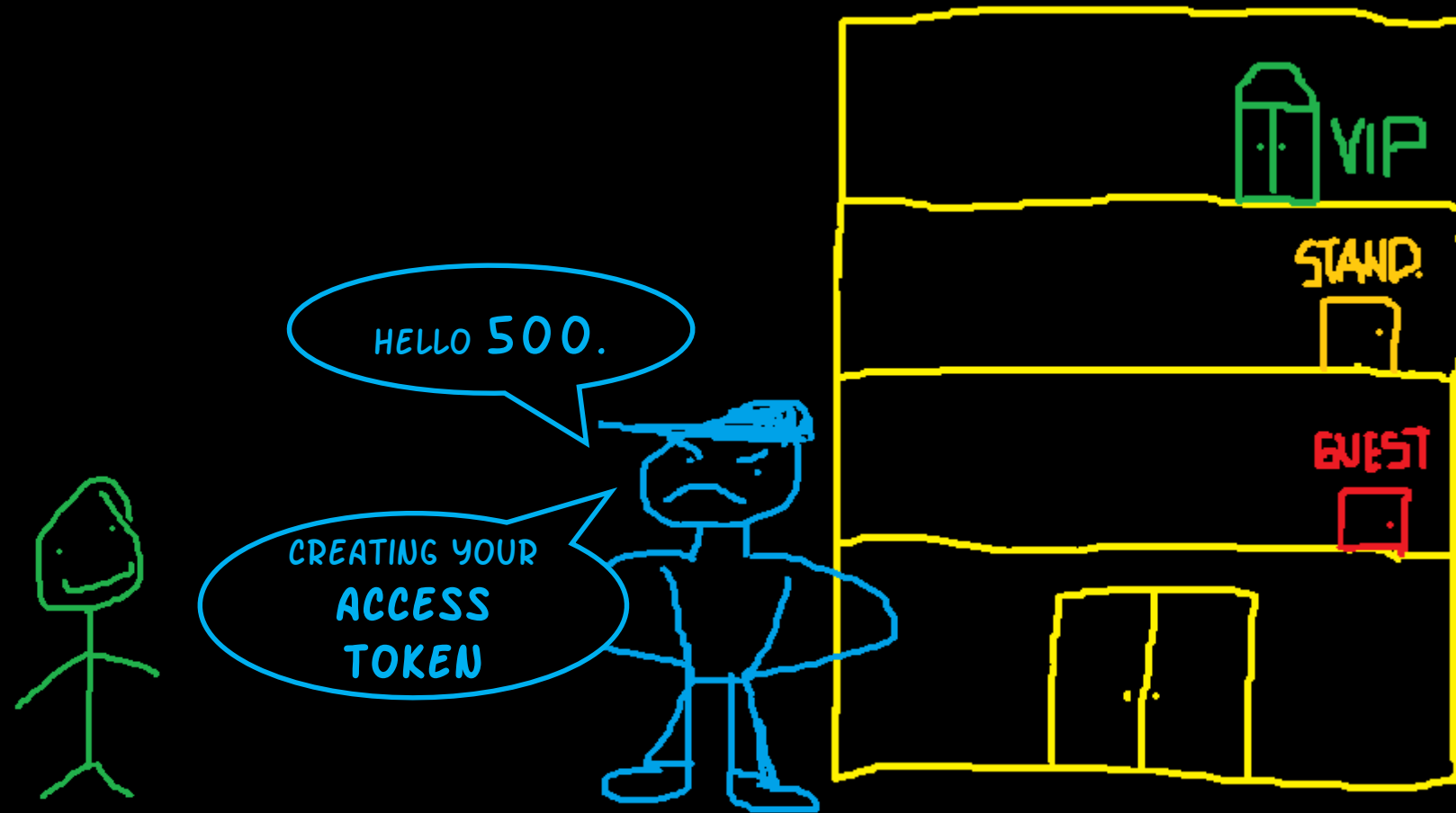


- 0x08. `MSV1_0` checks the information obtained from the `SAMSRV` response.
- 0x09. If OK, `MSV1_0` generates a `LUID` for the session.
- 0x0A. `MSV1_0` sends the login information (including `LUID`) to `LSASS`.



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

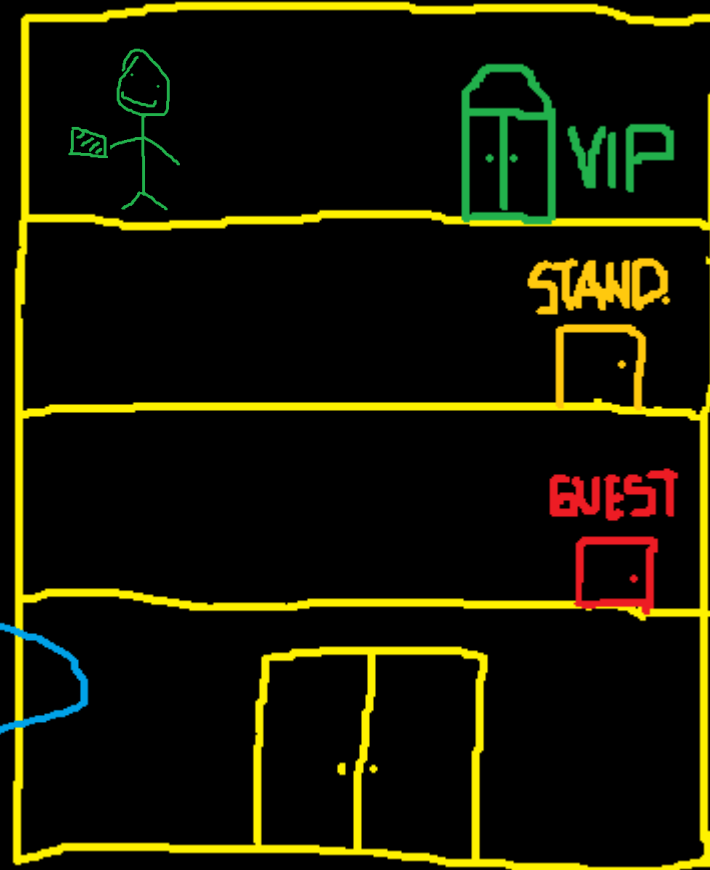




# Authorization

ACCESS GRANTED

TOKEN



# Agenda

- 0x01. Exposing the RID Hijacking Attack.
- 0x02. A Windows Authorization Story.
- 0x03. Hijacking the RID.
- 0x04. Demo.
- 0x05. Conclusions.

# 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-1-5-2196653972-2908857710-5094559845-500

# 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` sends `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 sends username and hashed password to the

SAMSRV. **How is the username associated with the SID?**

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

**What security info is retrieved?**

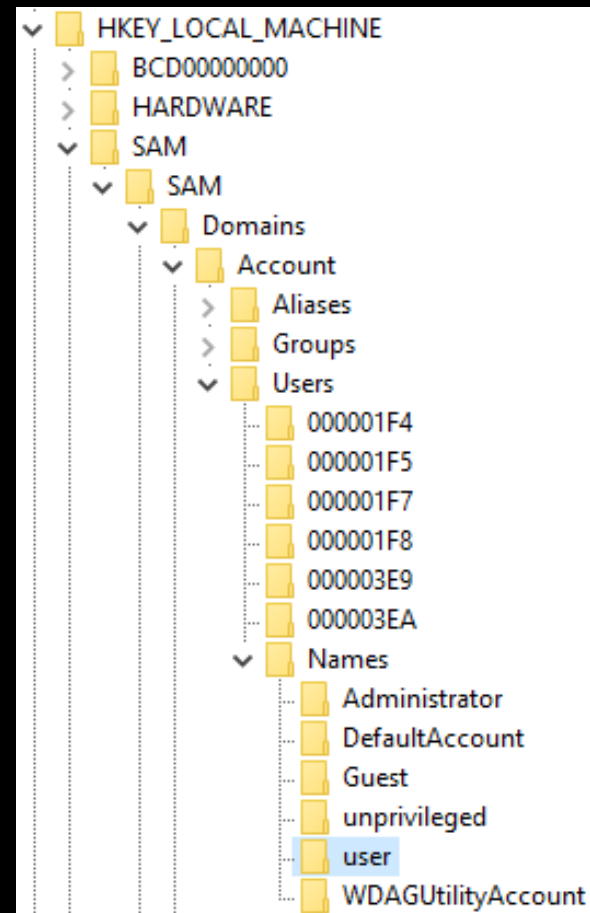




# Samsrv.dll and SAM

SAMSRV looks for the username at the SAM database.

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





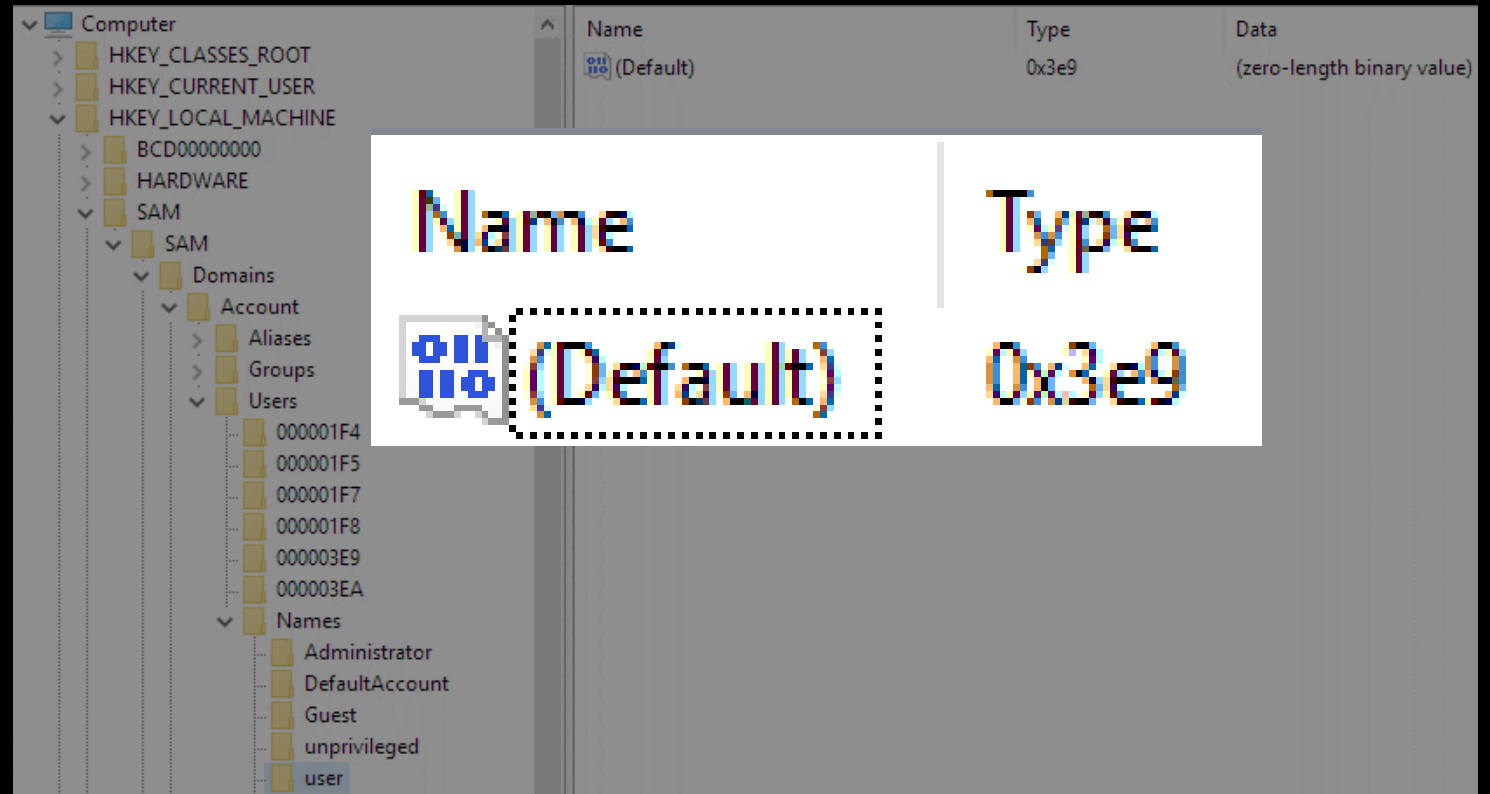
# Samsrv.dll and SAM

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.

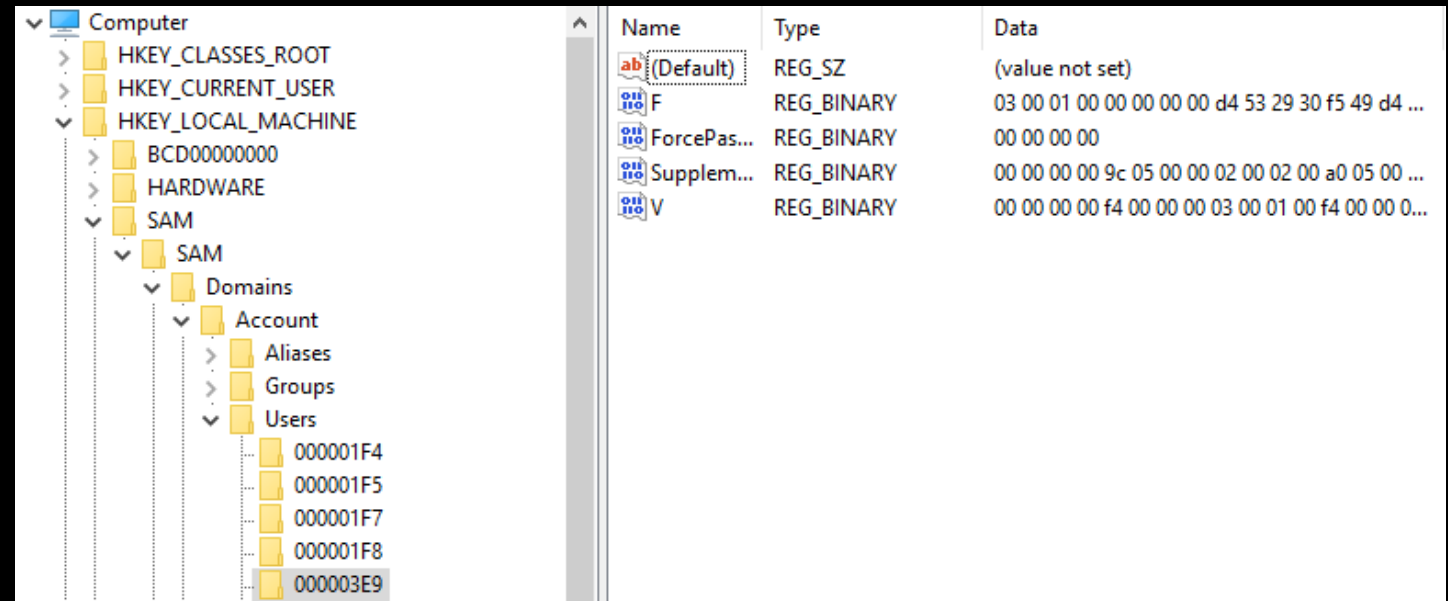
HKLM\SAM\SAM\Domains\Account\Users\Names



# Samsrv.dll and MSV1\_0.dll

HKLM\SAM\SAM\Domains\Account\Users

SAMSRV looks for the key associated with the RID.



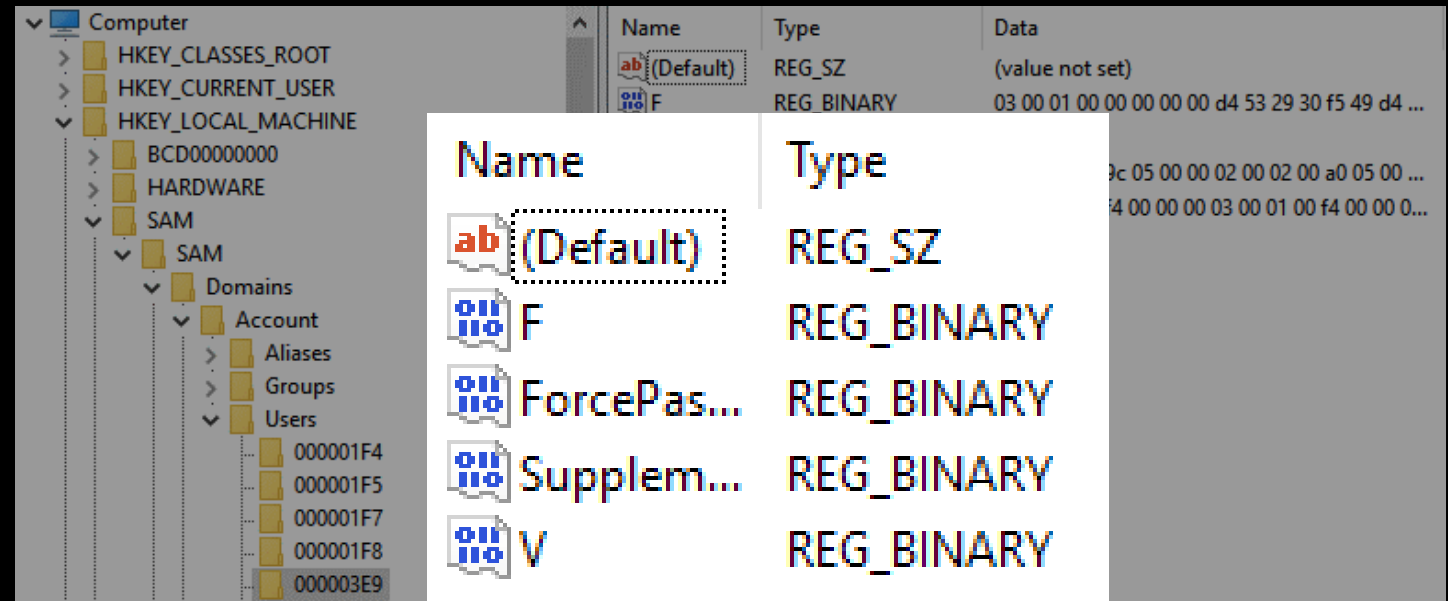
Name	Type	Data
(Default)	REG_SZ	(value not set)
F	REG_BINARY	03 00 01 00 00 00 00 d4 53 29 30 f5 49 d4 ...
ForcePas...	REG_BINARY	00 00 00 00
Supplem...	REG_BINARY	00 00 00 00 9c 05 00 00 02 00 02 00 a0 05 00 ...
V	REG_BINARY	00 00 00 00 f4 00 00 00 03 00 01 00 f4 00 00 0...

# Samsrv.dll and MSV1\_0.dll

HKLM\SAM\SAM\Domains\Account\Users

SAMSRV looks for the key associated with the RID.

SAMSRV grabs all the data stored in the referenced key.



Name	Type	Data
(Default)	REG_SZ	(value not set)
F	REG_BINARY	03 00 01 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	4 00 00 00 03 00 01 00 f4 00 00 0...
V	REG_BINARY	

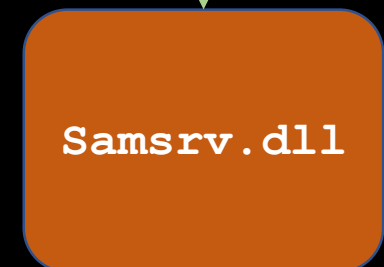
# Samsrv.dll and MSV1\_0.dll

**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
ab (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?

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

Consistent for all local users SIDs

Relative



# Understanding the attack

Why does the SAM store only the RID?

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

Consistent for all local users SIDs

Relative

What info is retrieved from the SAM?

# Understanding the attack

Why does the SAM store only the RID?

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

Consistent for all local users SIDs

Relative

What info is retrieved from the SAM?

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

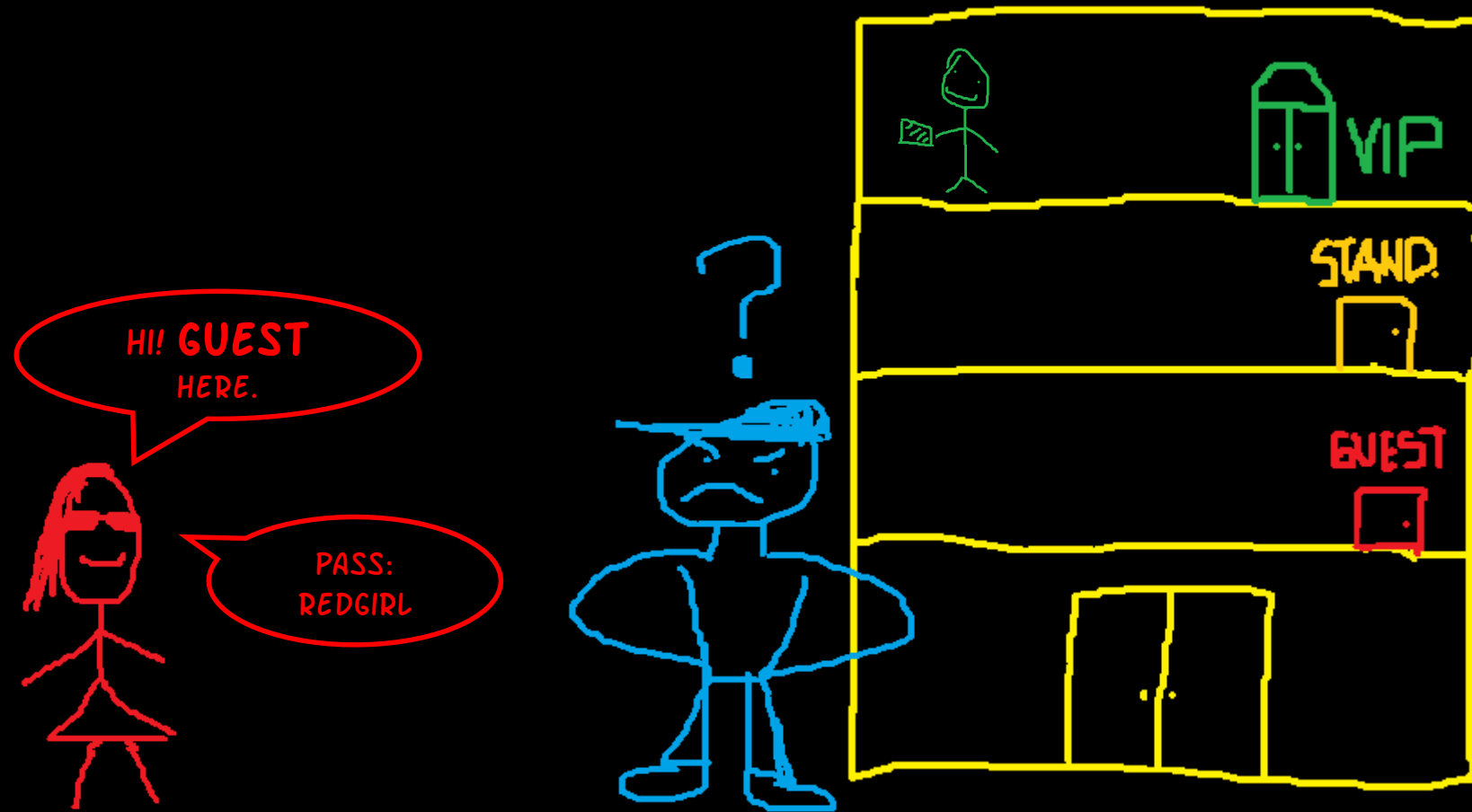
Password's Hash.

Account status (Active: Y/N).

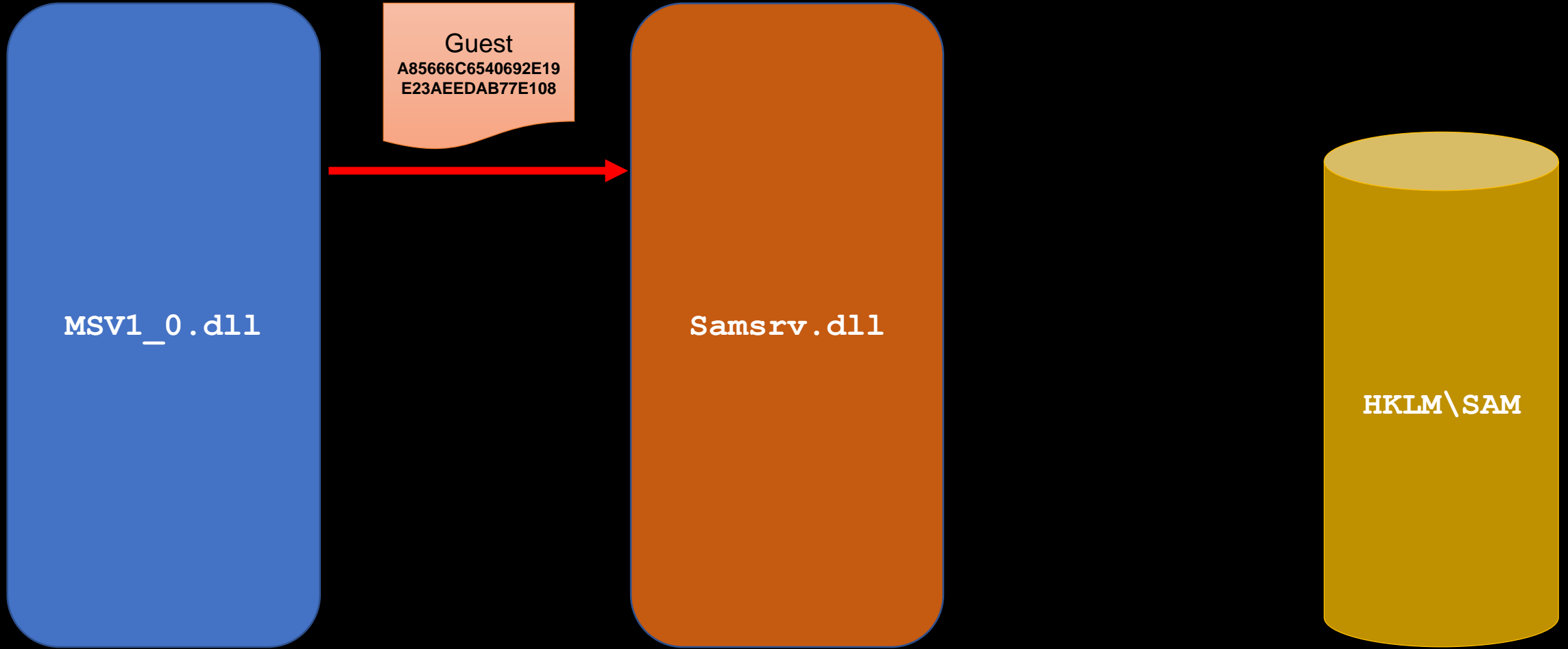
Some account restrictions.

A copy of the user's RID.

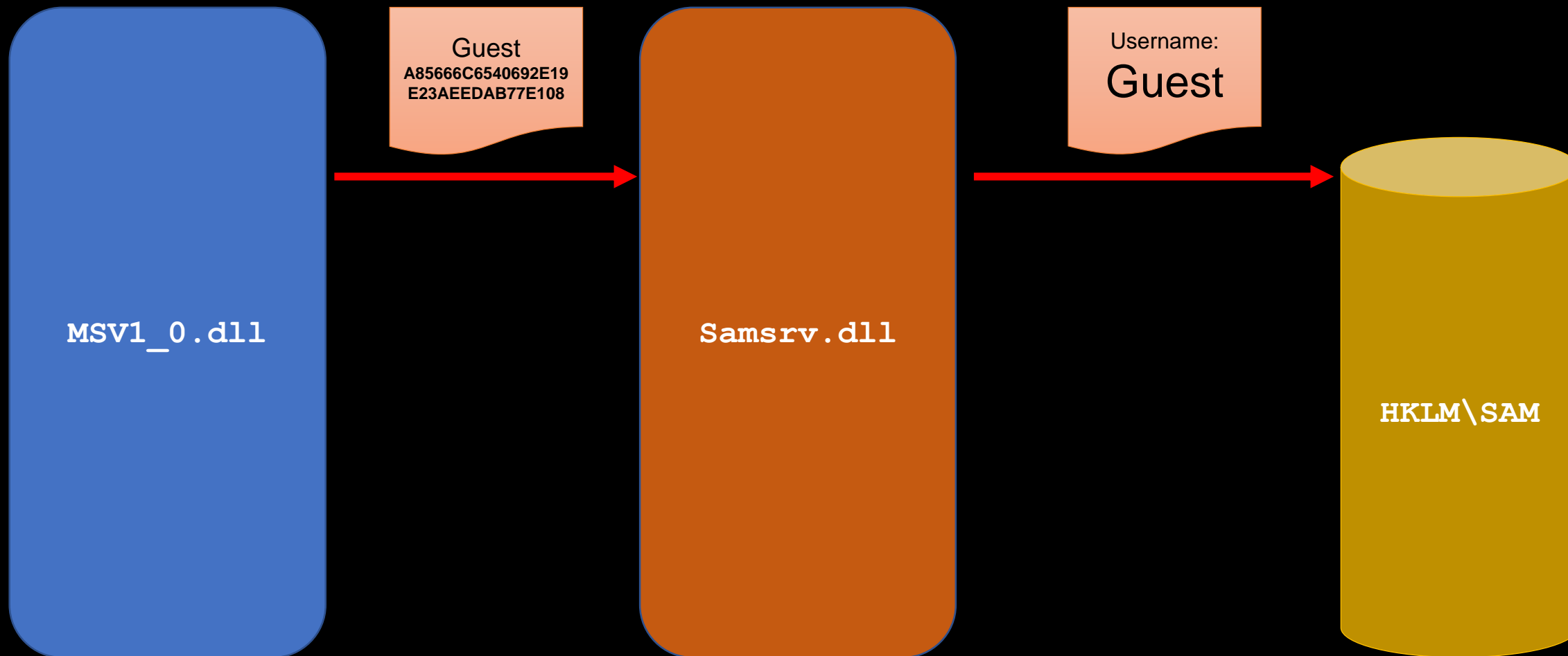
# Login as Guest



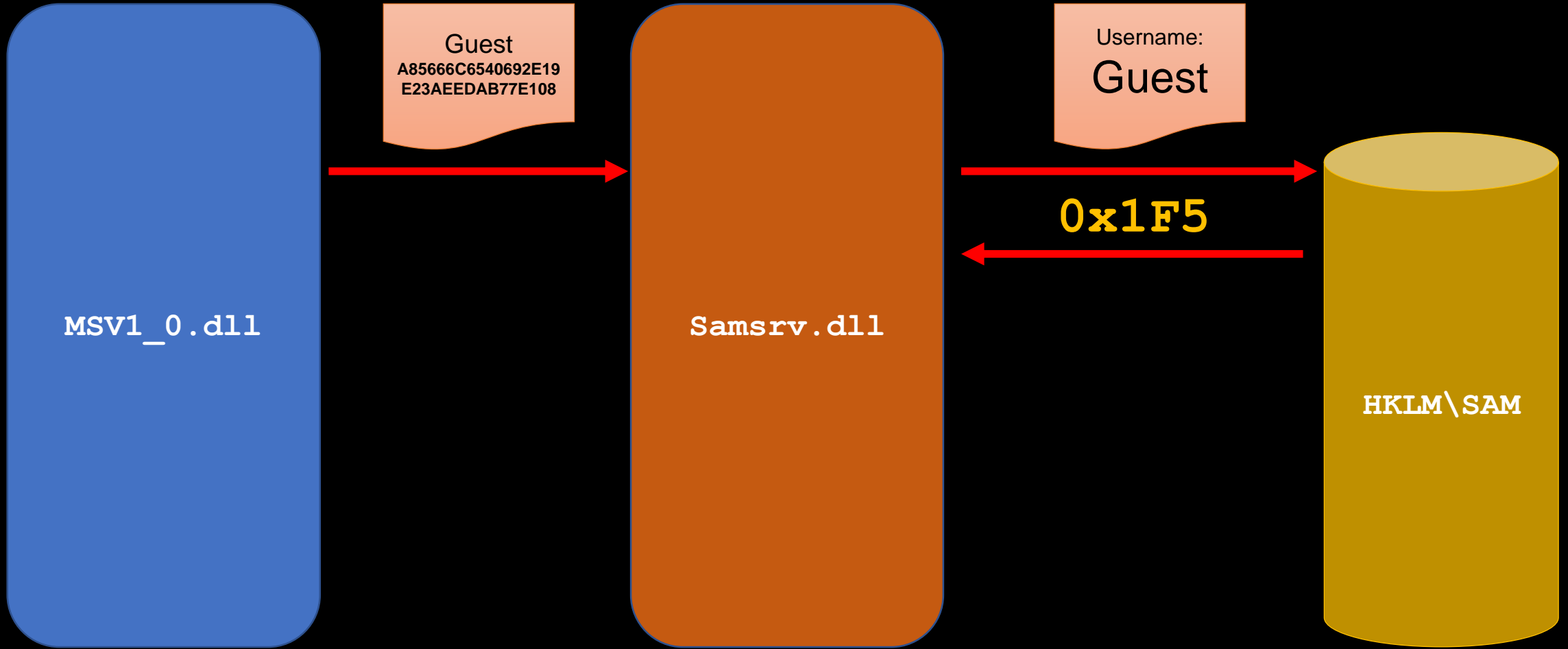
# Login as Guest



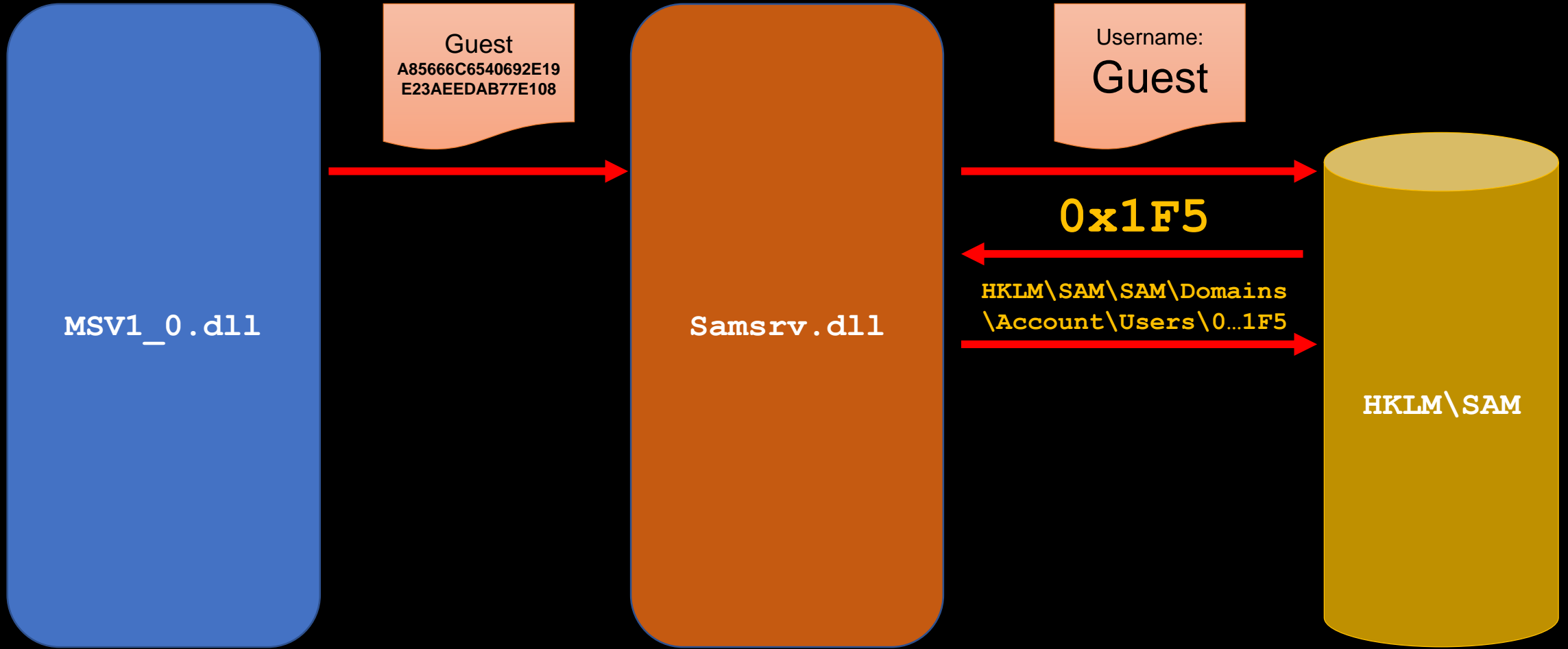
# Login as Guest



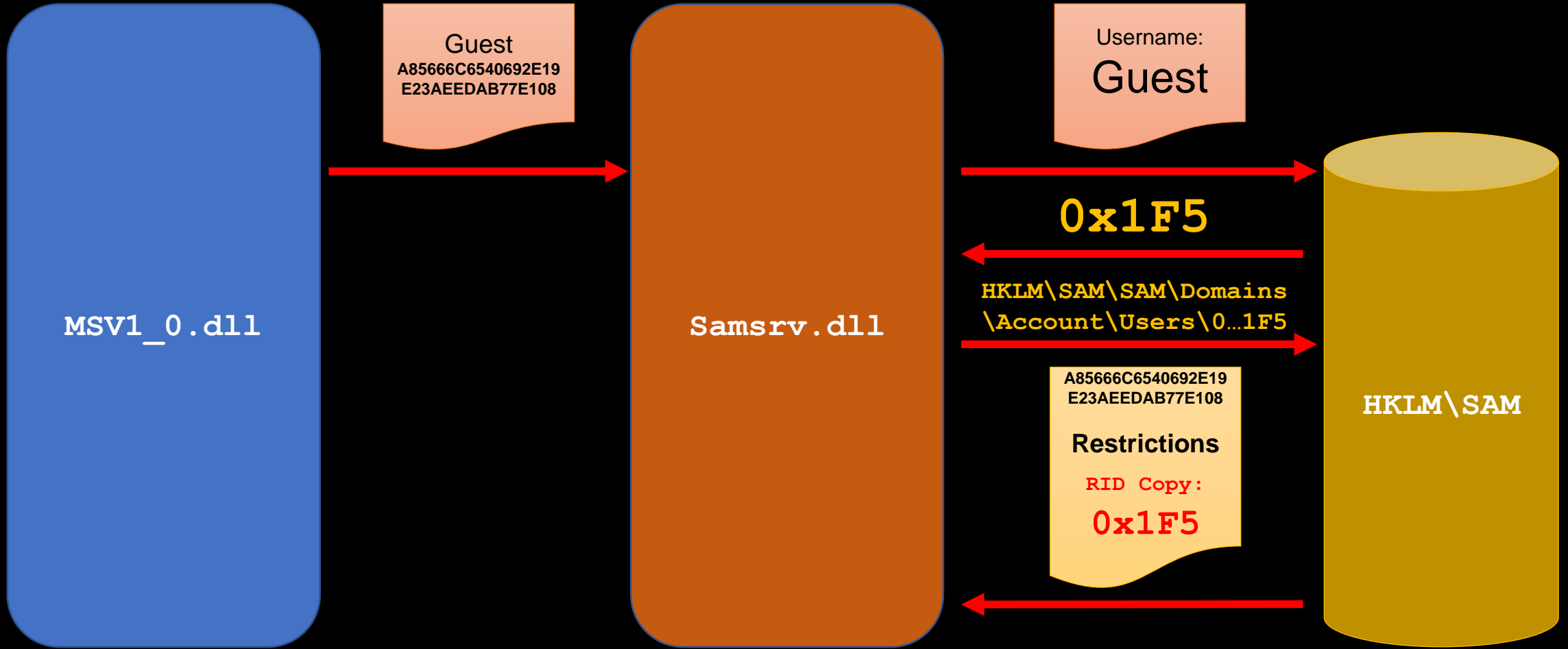
# Login as Guest



# Login as Guest

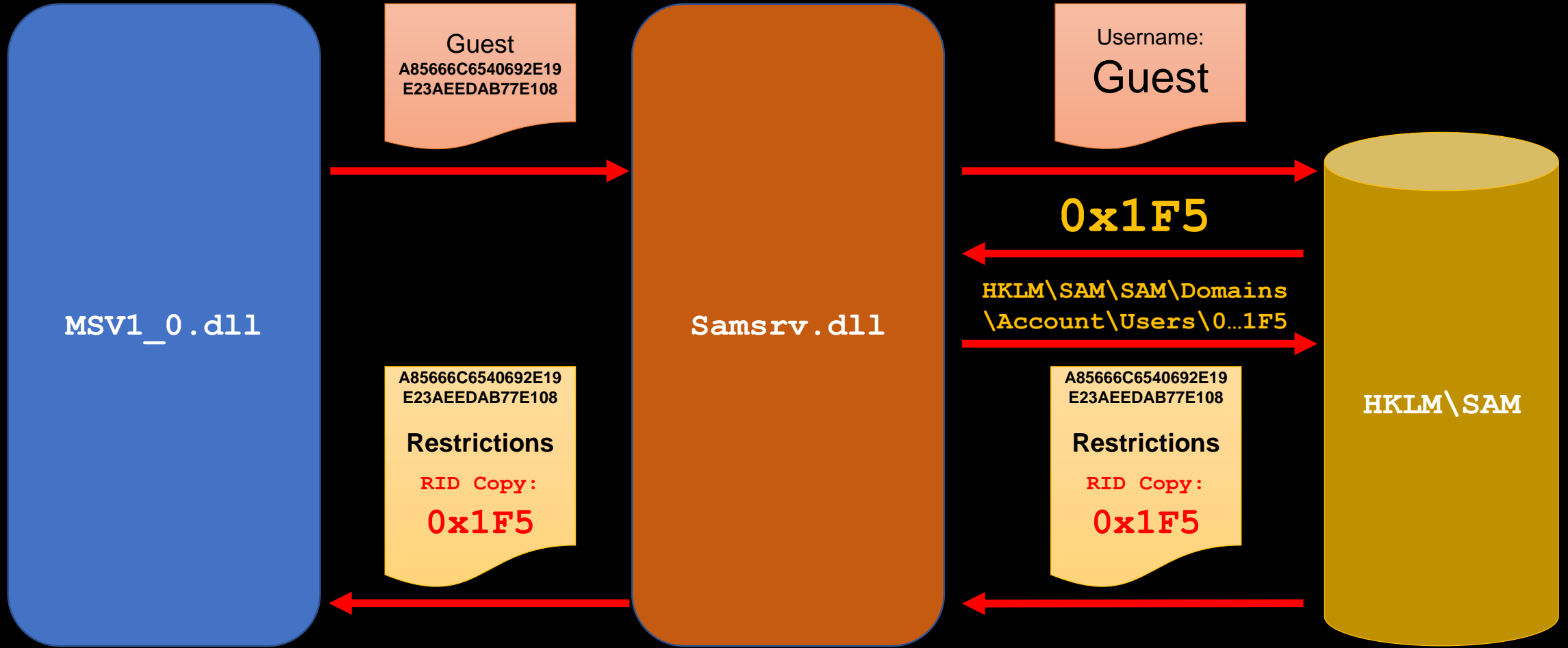


# Login as Guest





# Login as Guest



# Login as Guest (Case 1)



Login as Guest (Case 1)

**DENIED**

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

# Login as Guest (Case 2)

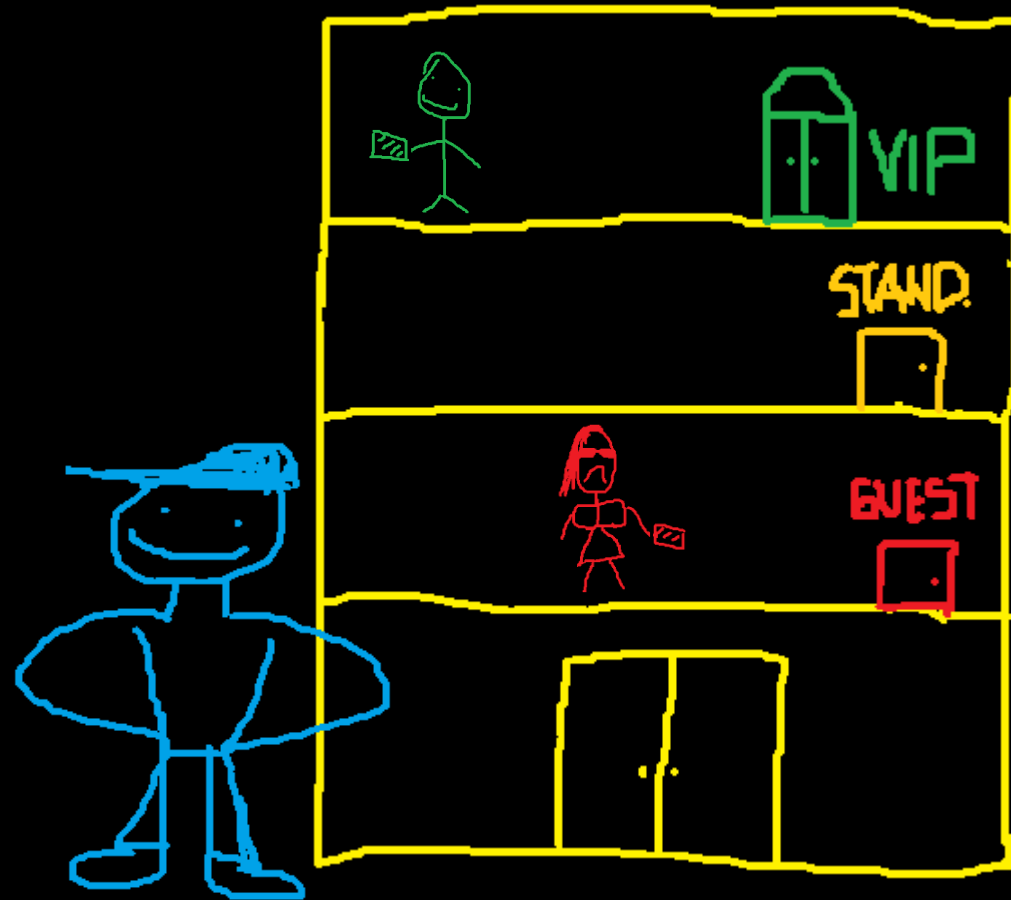


# Login as Guest (Case 2)

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`

A85666C6540692E19  
E23AEEDAB77E108

**Restrictions**

RID Copy:  
**0x1F5**

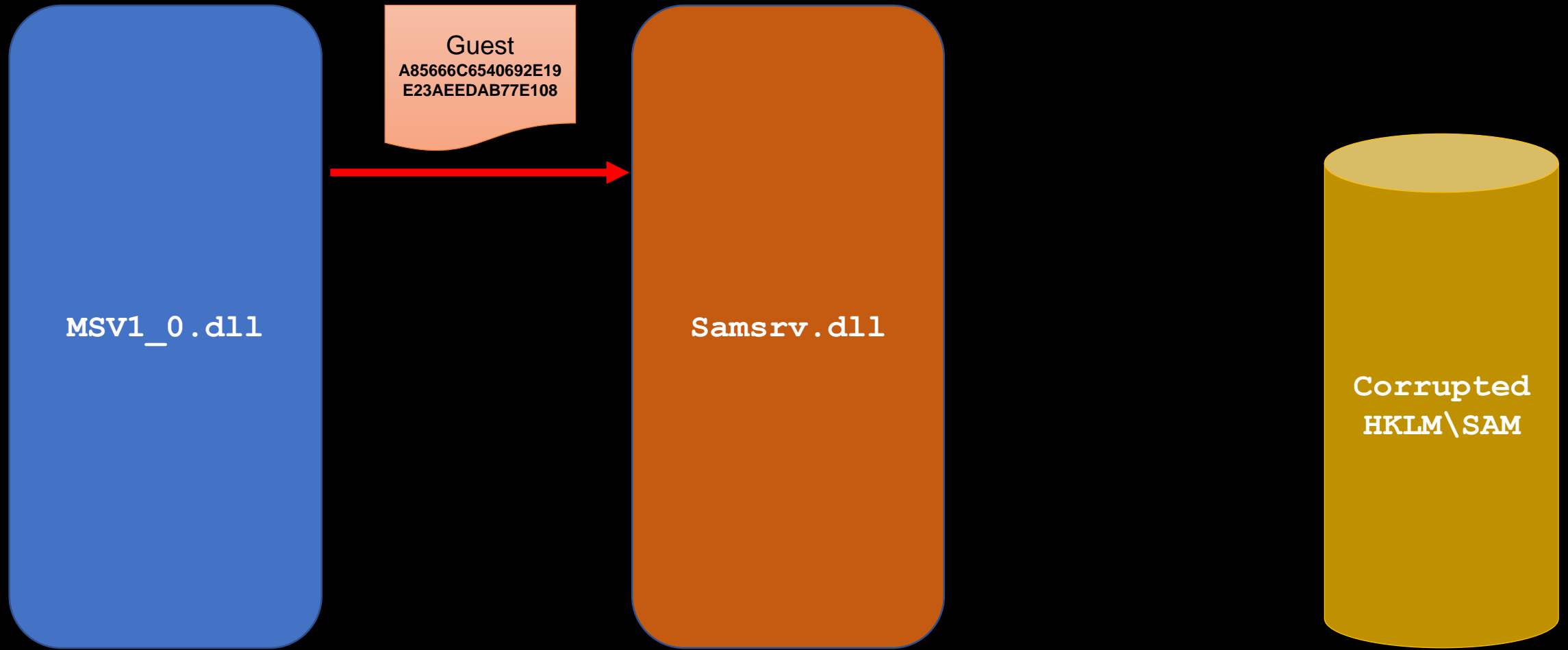


A85666C6540692E19  
E23AEEDAB77E108

**Restrictions**

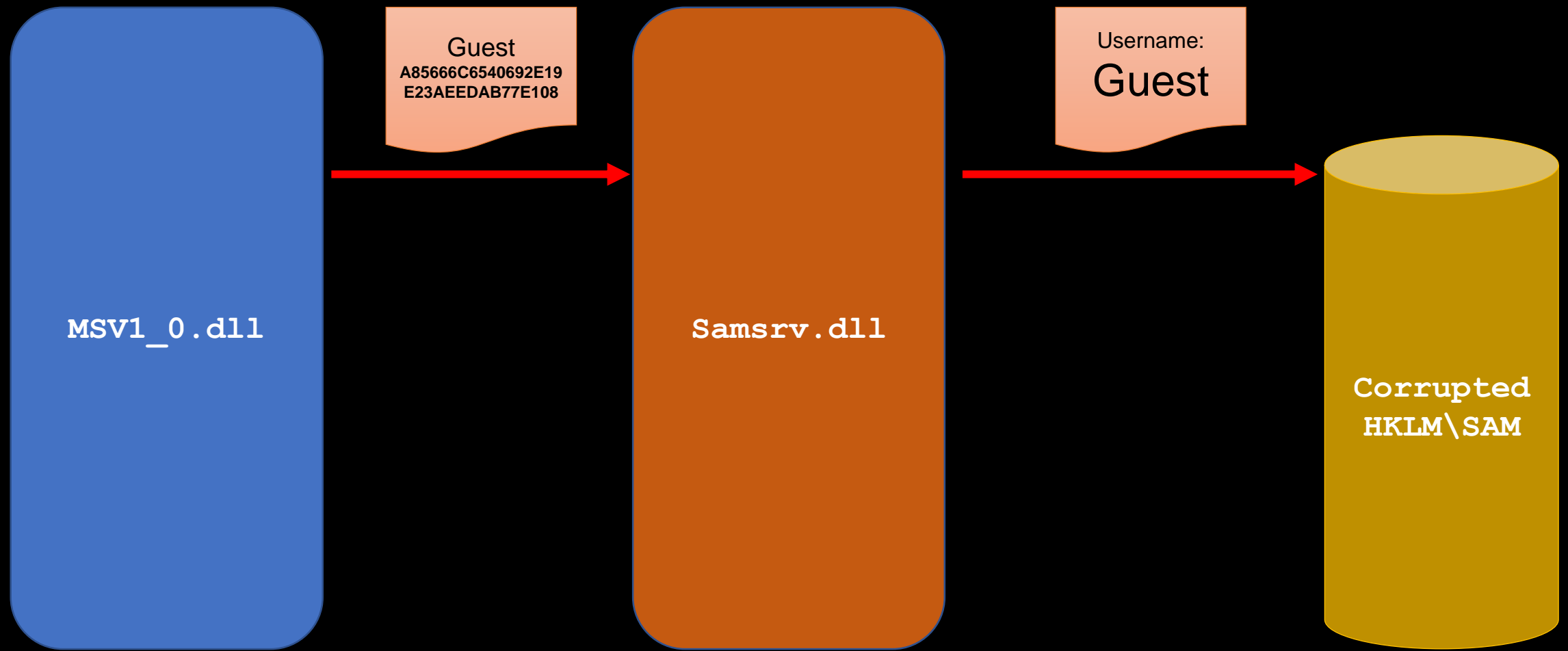
RID Copy:  
**0x1F4**

# Login as Guest (the comeback)

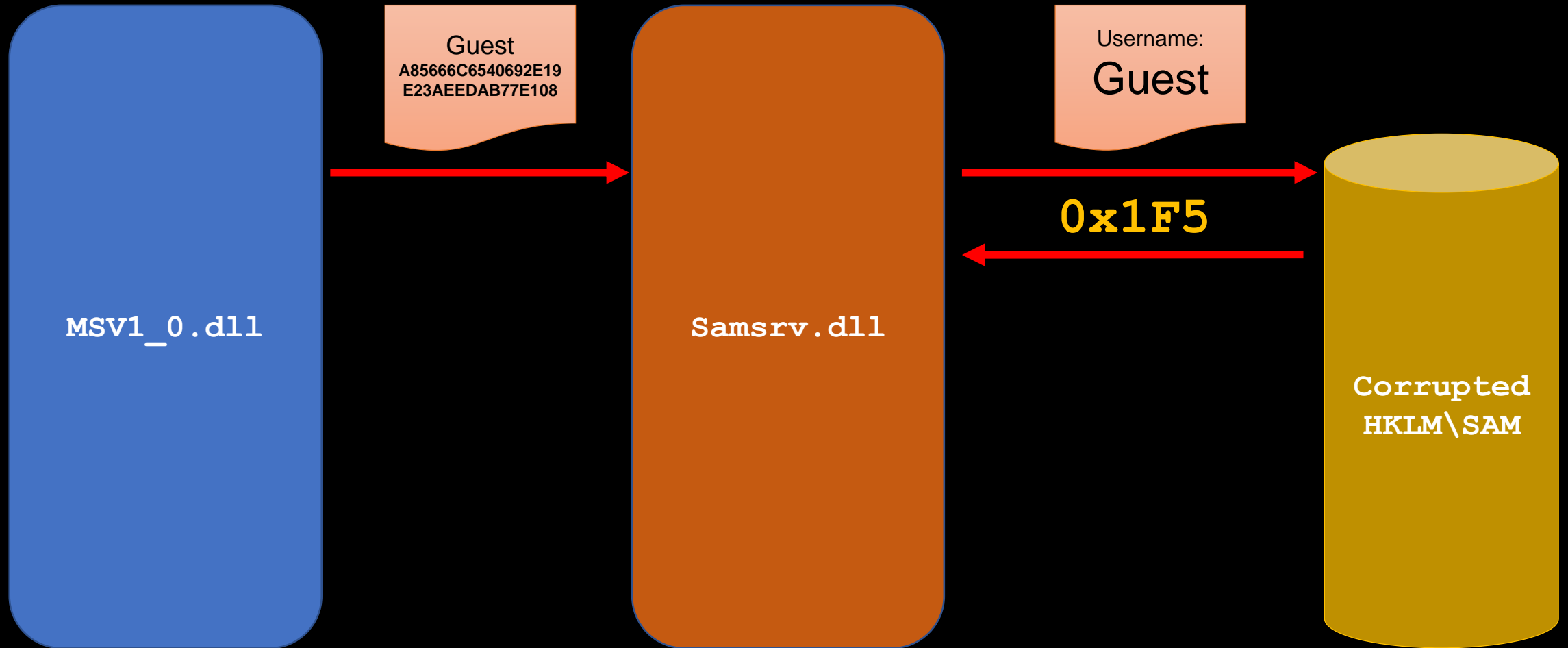




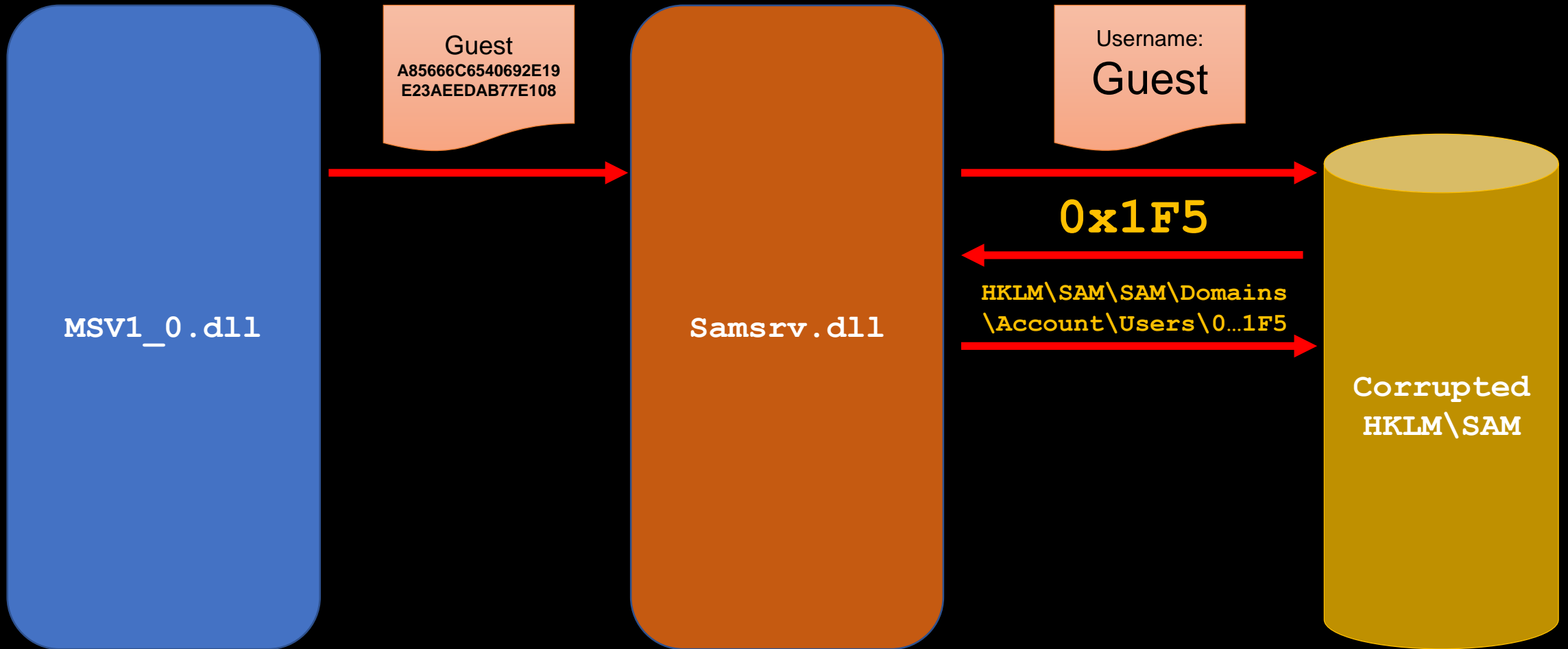
# Login as Guest (the comeback)



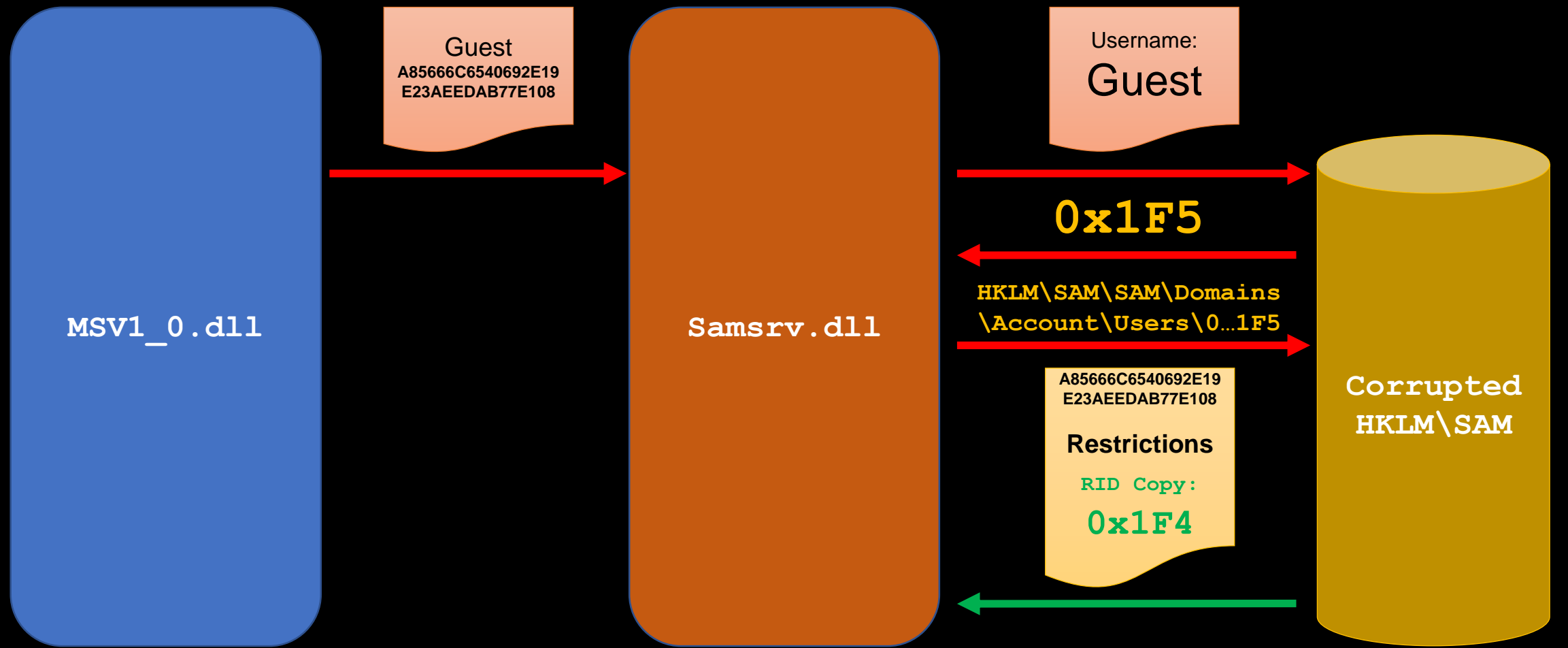
# Login as Guest (the comeback)



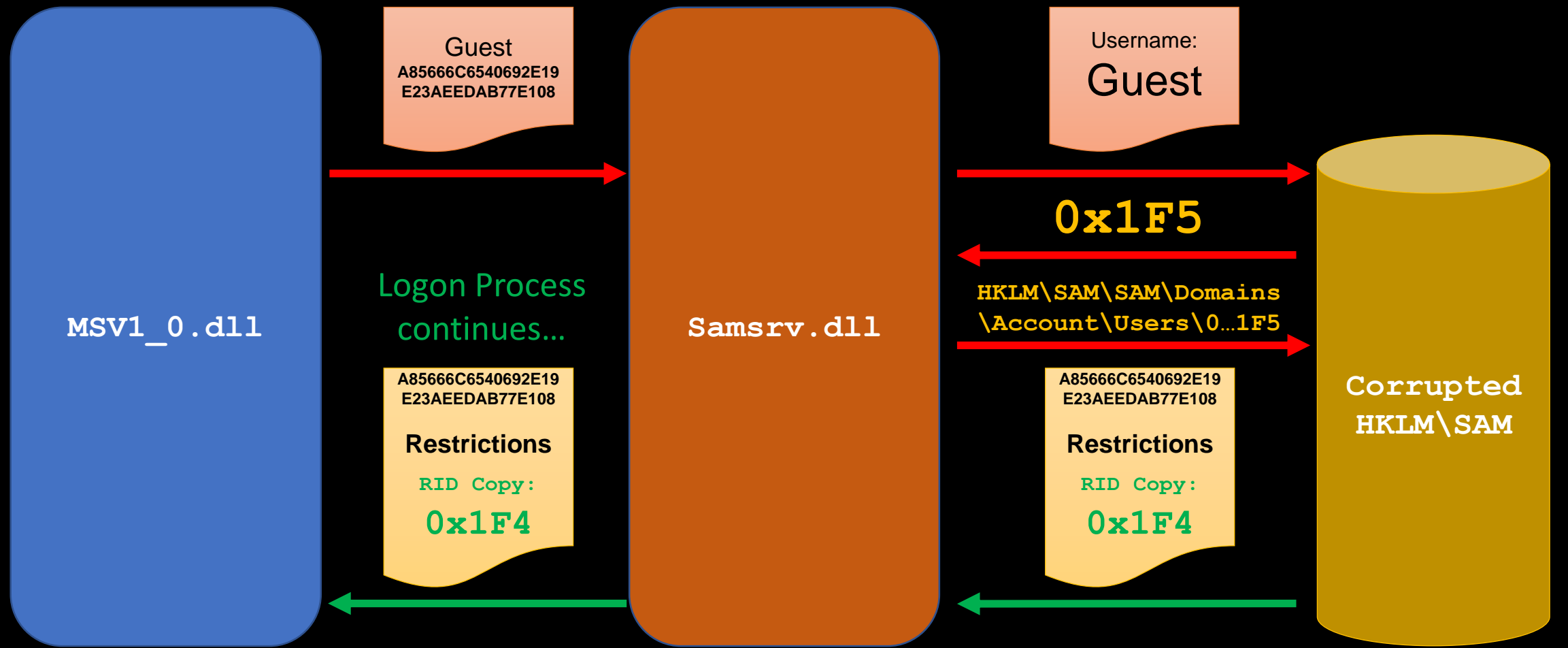
# Login as Guest (the comeback)



# Login as Guest (the comeback)



# 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

MSV1\_0.dll

A85666C6540692E19  
E23AEEDAB77E108

Restrictions

RID Copy:

0x1F4



# Login as Guest (the comeback)

MSV1\_0 checks the account restrictions provided from SAMSRV.

If allowed, then compares:

SAMSRV response password hash  
Hash will be the  
VS  
User entered password  
same

MSV1\_0.dll

A85666C6540692E19  
E23AEEDAB77E108

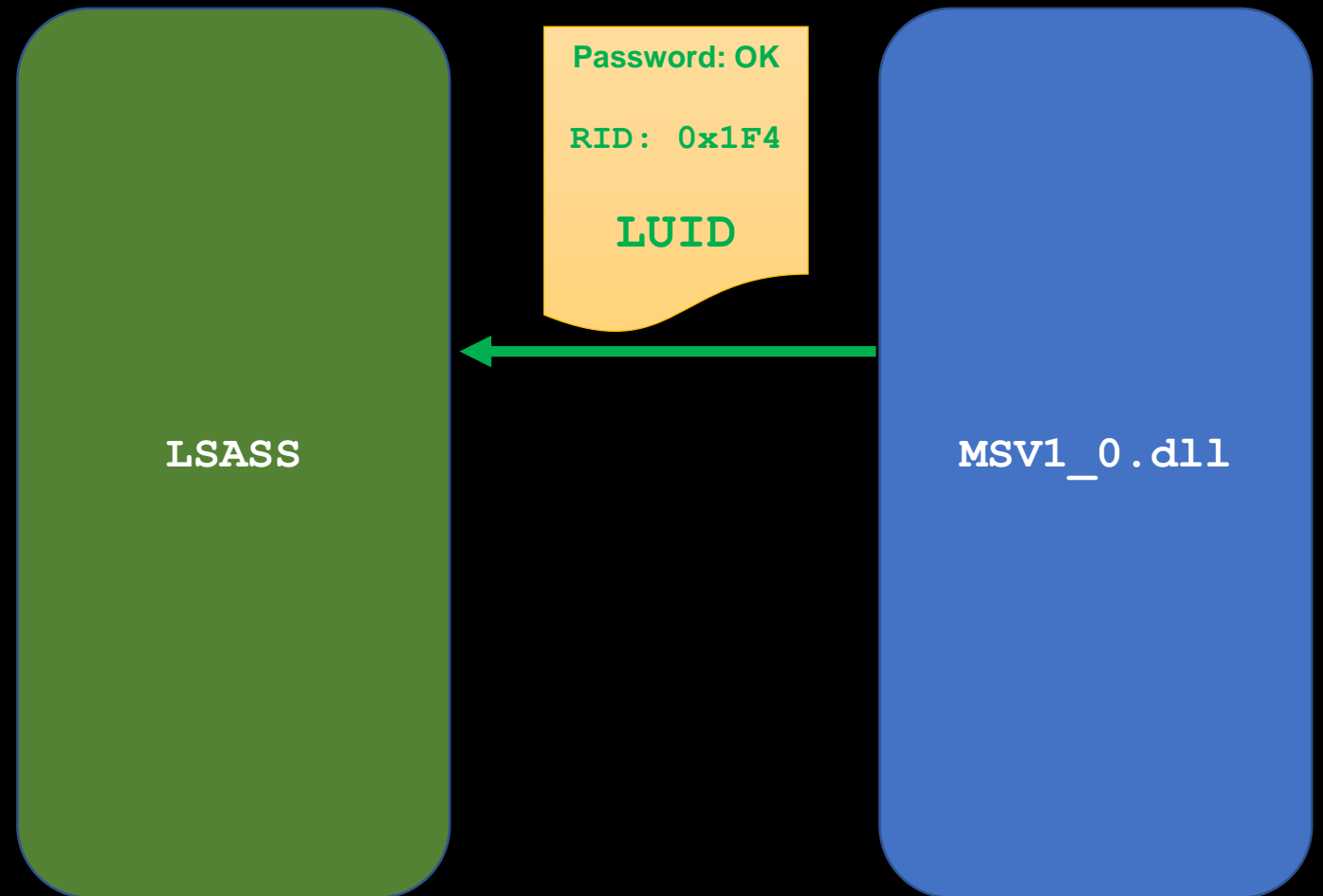
Restrictions

RID Copy:

0x1F4



# Login as Guest (the comeback)





# Login as Guest (the comeback)

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session (LUID)
Mandatory Policy
Administrators
Default DACL
SID-1-5-.....-500
Group 1 SID
...
Group n SID
Restricted SID 1
...
Restricted SID n
Privilege 1
...
Privilege n

Creates the Access Token with RID 500



LSASS

Password: OK  
 RID: 0x1F4  
 LUID



MSV1\_0.dll



# Login as Guest (the comeback)

# RID HIJACKING

Token source
Impersonation type
Token ID
Authentication ID
Modified ID
Expiration Time
Session ID
Flags
Logon session (LUID)
Mandatory Policy
Administrators
Default DACL
SID-1-5-...-500
Group SID
Group name
Restrictions
Restricted rights
Privilege 1
...
Privilege n

Before

Creates the Access Token with RID 500

LSASS

Password: OK

RID: 0x1F4

After



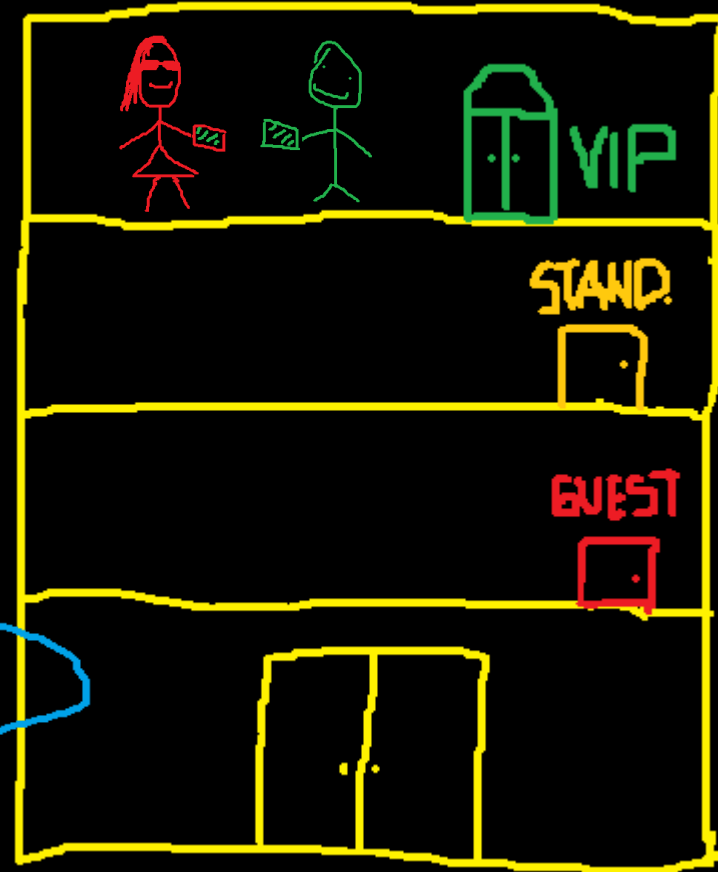
# Login as Guest



# Login as Guest

ACCESS GRANTED

TOKEN



# 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

- 0x01. Exposing the RID Hijacking Attack.
- 0x02. A Windows Authorization Story.
- 0x03. Hijacking the RID.
- 0x04. Demo.
- 0x05. Conclusions.



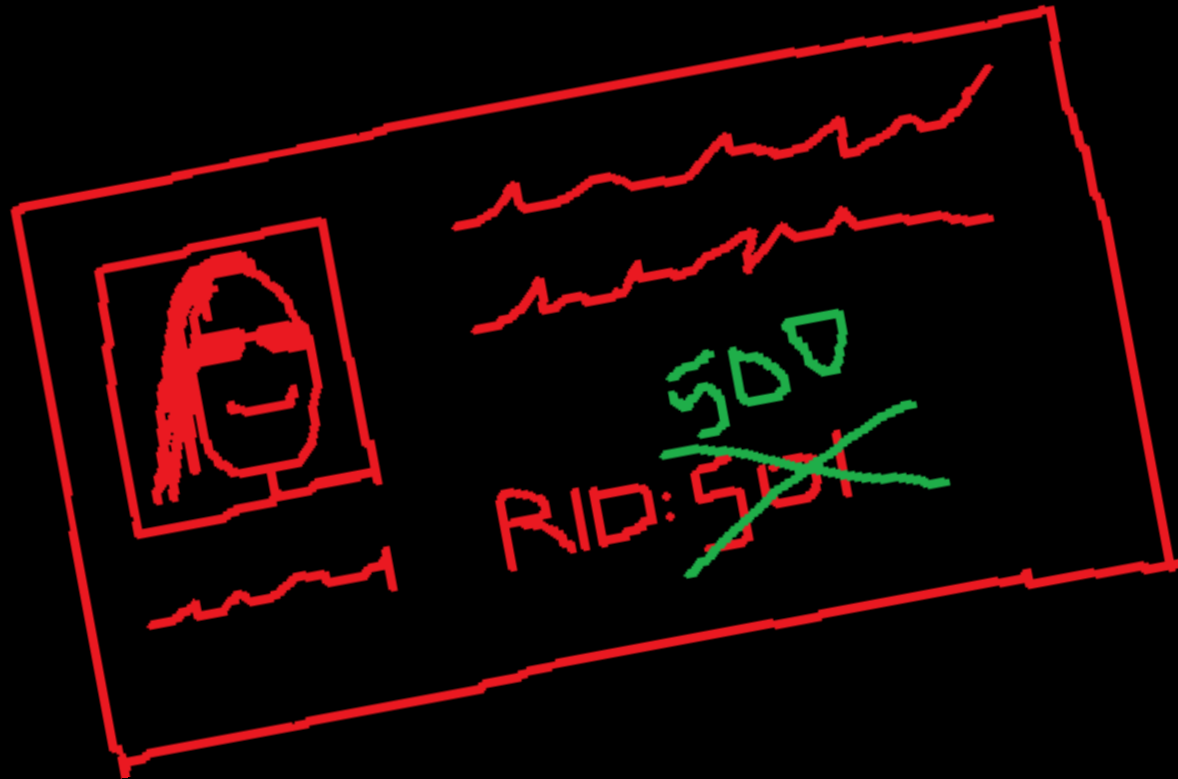
# Demonstration



# Agenda

- 0x01. Exposing the RID Hijacking Attack.
- 0x02. A Windows Authorization Story.
- 0x03. Hijacking the RID.
- 0x04. Demo.
- 0x05. Conclusions.**

# Conclusions



# References

1. <http://csl.com.co/rid-hijacking/>
2. Russinovich, Mark. Solomon, David A. Ionescu, Alex. "Windows Internals". 6<sup>th</sup> Edition.
3. Scambray, Joel. McClure, Stuart. "Hacking Exposed: Windows Security Secrets & Solutions". 3<sup>rd</sup> Edition.
4. [https://technet.microsoft.com/pt-pt/library/cc780332\(v=ws.10\).aspx](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>