Hunting for Bugs in the Windows Authentication Stack

James Forshaw
@tiraniddo@infosec.exchange

BlueHat IL
2023

Who am I?

- Researcher @ Google Project Ze
- Specialize in Windows
- Writer of Tools!

“Never met a logical vulnerability I didn’t like.”

Agenda

- Overview of the Windows Authentication Stack
- Review Methodology
- Reverse Engineering LSA Providers
- Example Vulnerabilities

Windows Authentication Stack?

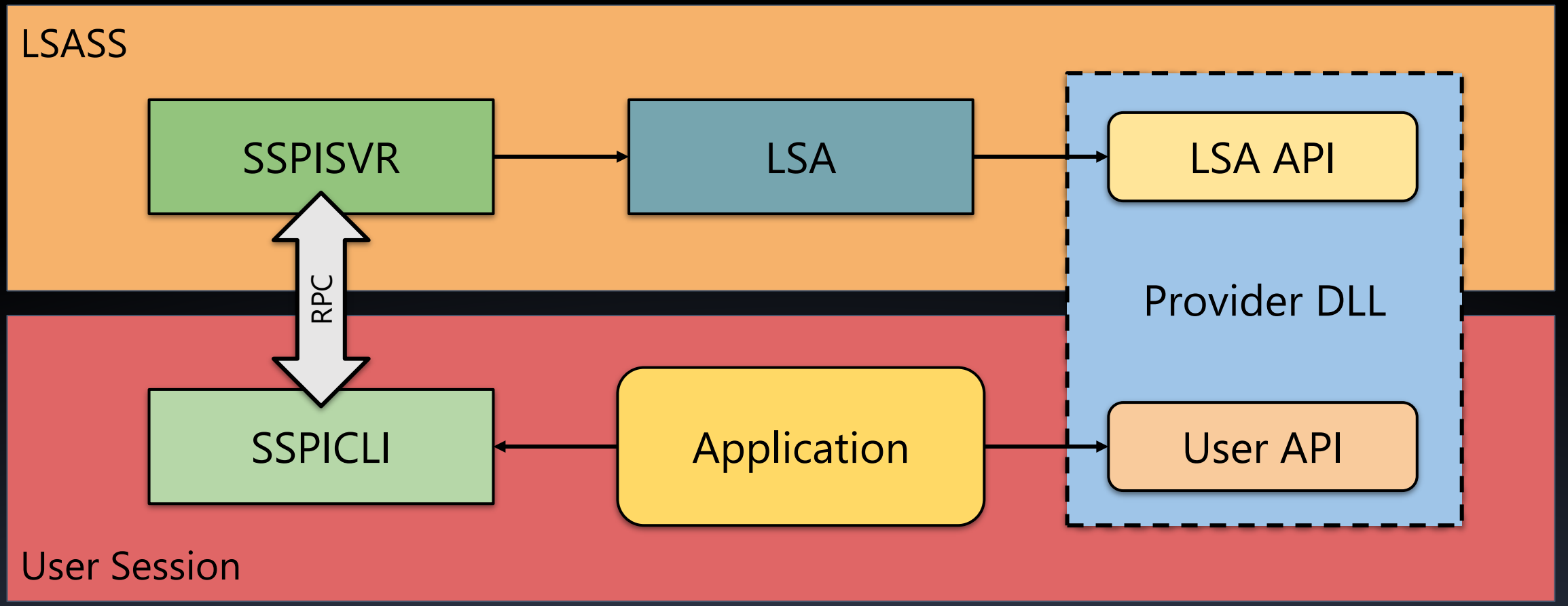
LSA Authentication Model

The *Local Security Authority* (LSA) authentication model has the following features:

- LSA authentication supports custom authentication packages. This allows end-customers and independent software vendors (ISVs) to customize or replace authentication routines to meet requirements beyond those provided by the standard Microsoft authentication packages. While the authentication packages provided by Microsoft require a user name and password logon data, a custom authentication package can take other forms of logon data, such as ATM card information and a personal identification number (PIN). A custom authentication package can also be used to implement a new *security protocol*.

https://learn.microsoft.com/en-us/windows/win32/secauthn/lsa-authentication-model

Security Support Provider Interface (SSPI)



Entry Points to SSPI

Local Authentication

LsaLogonUser

Call LSA Provider

LsaCallAuthenticationPackage

Network Authentication

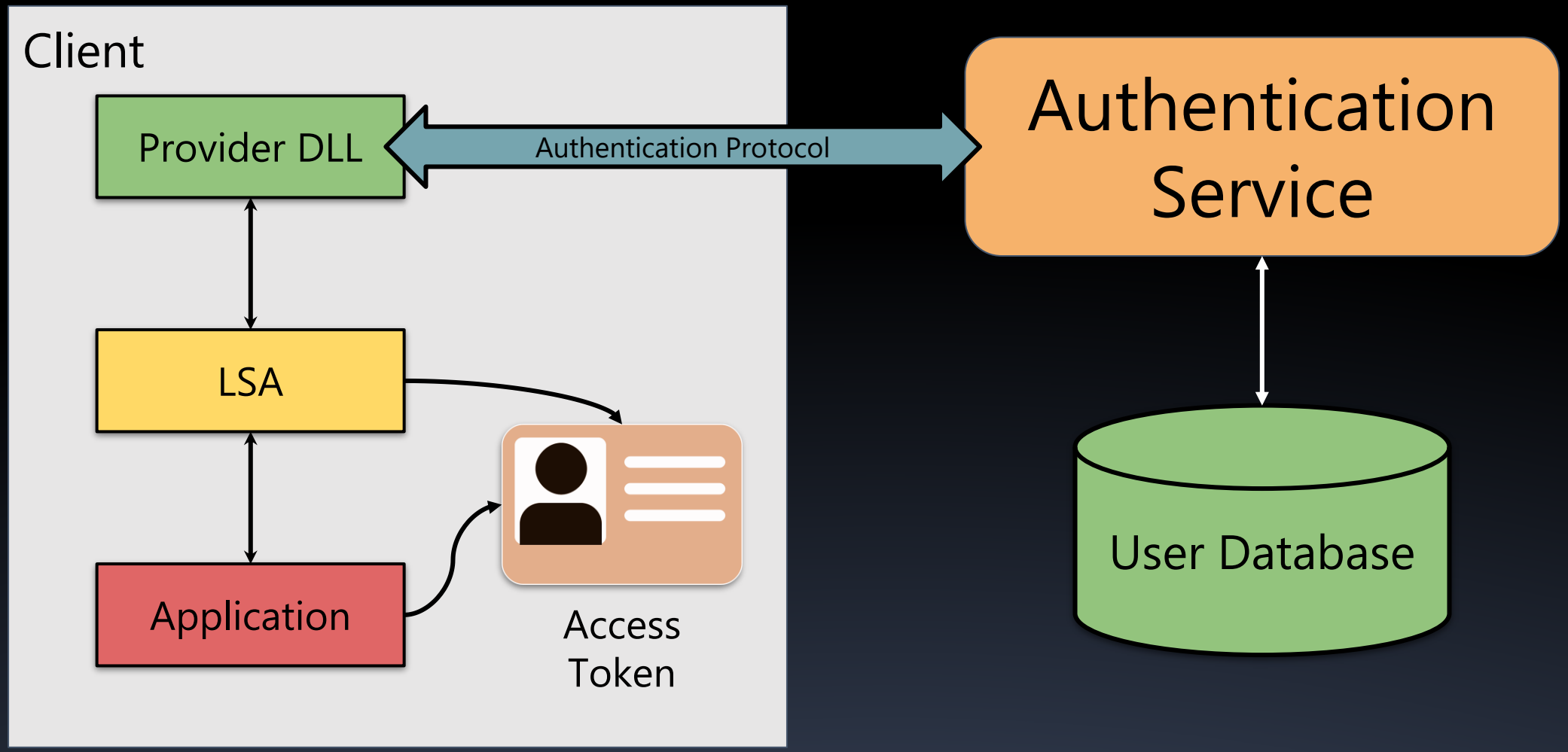
AcquireCredentialsHandle

AcceptSecurityContext

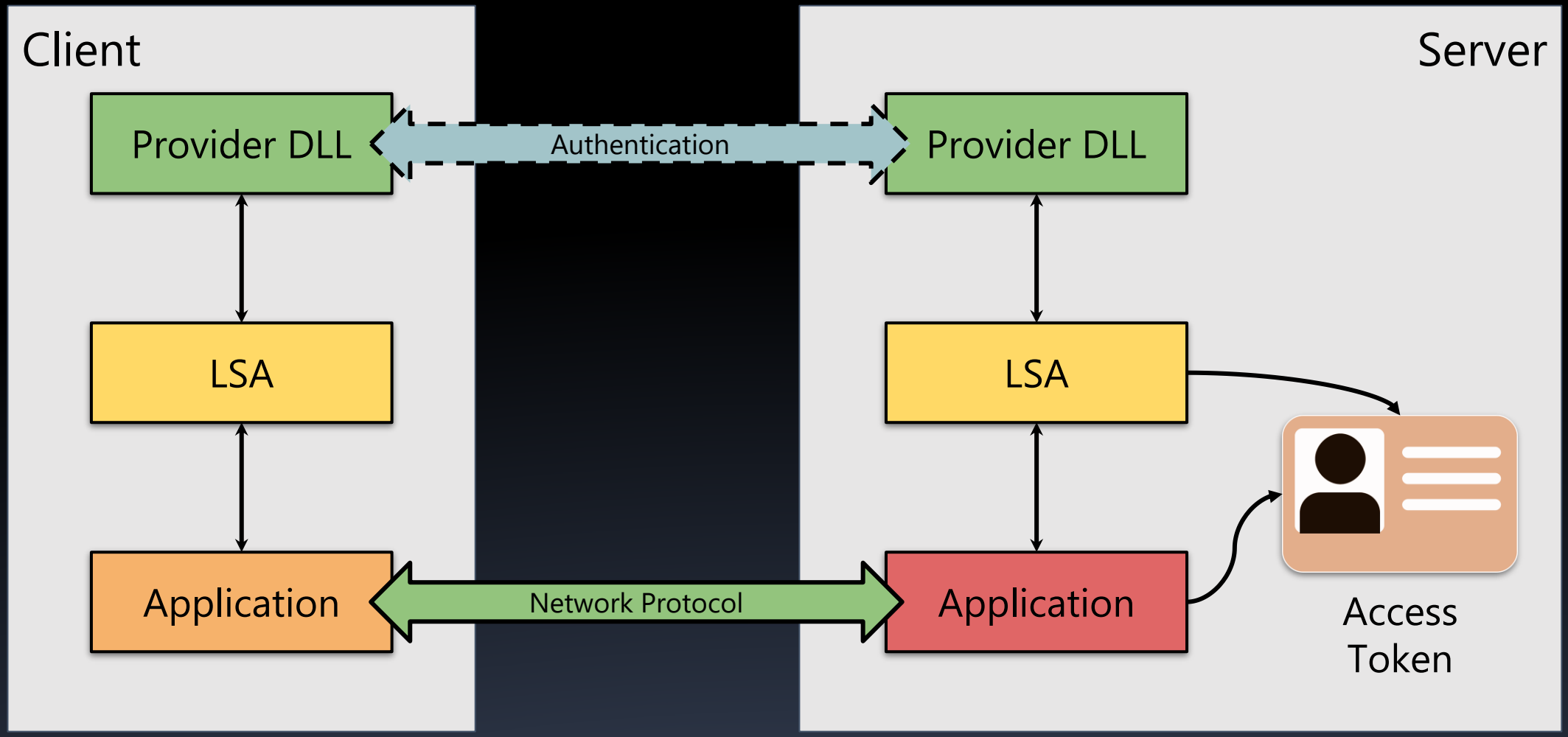
SetContextAttributes

InitializeSecurityContext

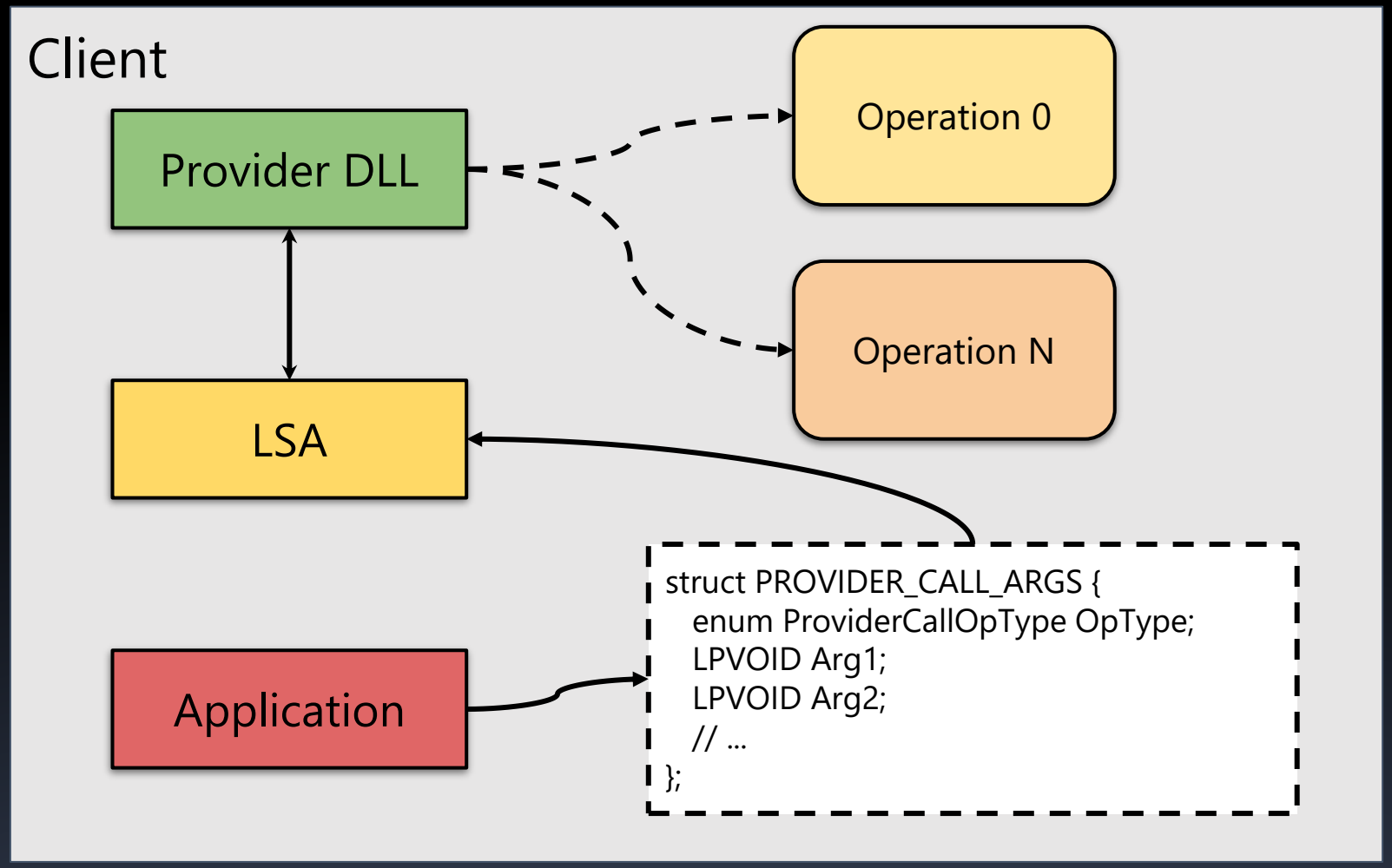
Local Authentication



Network Authentication



Call LSA Provider



Review Methodology

Reverse Engineering the Authentication Stack

Bug Classes

Arbitrary Code Execution

Authentication Bypass

Authorization Bypass

Information Disclosure

Enumerating Security Packages

EnumerateSecurityPackagesW function (sspi.h)

The EnumerateSecurityPackages function returns an array of SecPkgInfo structures that provide information about the security packages available to the client.

Syntax

```
SECURITY_STATUS SEC_ENTRY EnumerateSecurityPackagesW(
  [in] unsigned long *pcPackages,
  [in] PSecPkgInfoW  *ppPackageInfo
);
```

https://learn.microsoft.com/en-us/windows/win32/api/sspi/nf-sspi-enumeratesecuritypackagesw

Where's the Package DLL?

```
struct SecPkgInfoW {
  unsigned long  fCapabilities;
  unsigned short wVersion;
  unsigned short wRPCID;
  unsigned long  cbMaxToken;
  SEC_WCHAR      *Name;
  SEC_WCHAR      *Comment;
};
```

No DLL path here?

DEMO

Enumerating packages.

DLL Initialization in LSA

SpLsaModeInitializeFn callback function (ntsecpkg.h)

The SpLsaModeInitialize function is called once by the Local Security Authority (LSA) for each registered security support provider/authentication package (SSP/AP) DLL it loads. This function provides the LSA with pointers to the functions implemented by each security package in the SSP/AP DLL.

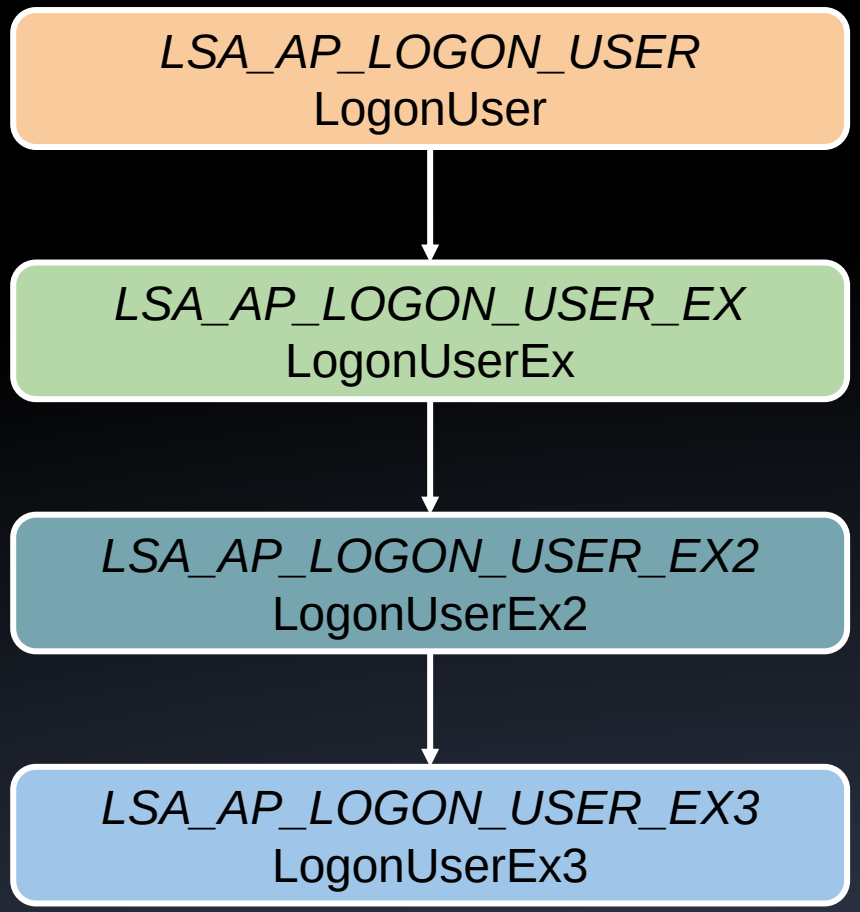
Syntax

```
NTSTATUS SpLsaModeInitializeFn(
  [in]  ULONG LsaVersion,
  [out] PULONG PackageVersion,
  [out] PSECPKG_FUNCTION_TABLE *ppTables,
  [out] PULONG pcTables
)
```

```
struct SECPKG_FUNCTION_TABLE {
  PLSA_AP_INITIALIZE_PACKAGE  InitializePackage;
  PLSA_AP_LOGON_USER          LogonUser;
  PLSA_AP_CALL_PACKAGE        CallPackage;
  ...
};
```

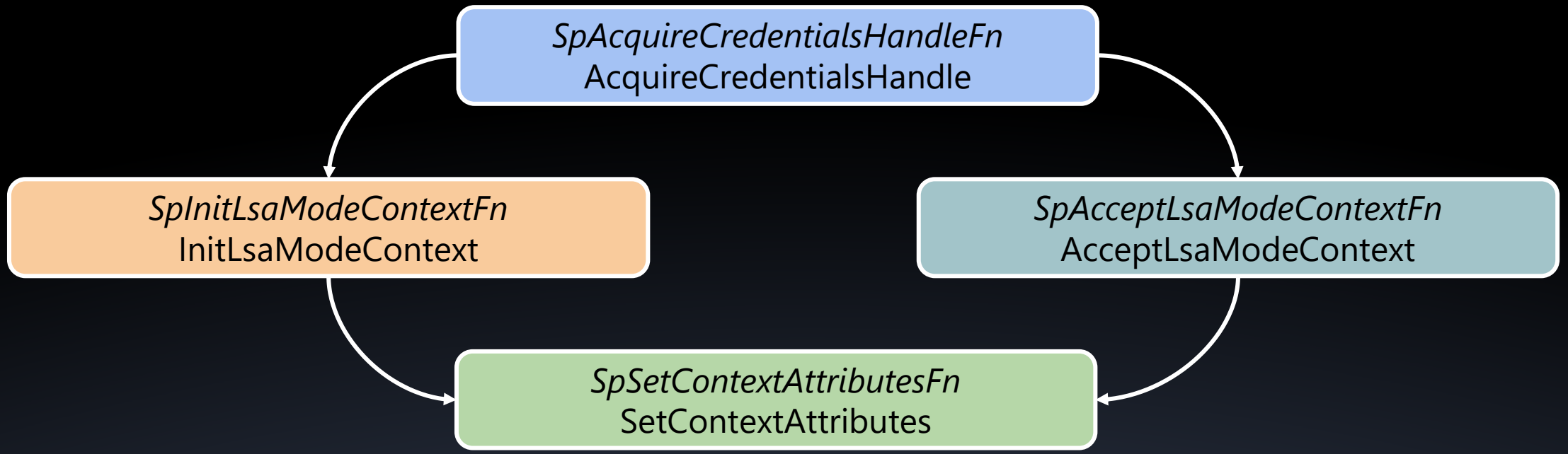
https://learn.microsoft.com/en-us/windows/win32/api/ntsecpkg/nc-ntsecpkg-splsamodeinitializefn

Local Authentication LSA Entry Points



LSA calls the highest supported version

Network Authentication LSA Entry Points



Call Provider LSA Entry Points

PLSA_AP_CALL_PACKAGE
CallPackage

Caller has SeTcbPrivilege enabled.

PLSA_AP_CALL_PACKAGE_UNTRUSTED
CallPackageUntrusted

Caller is untrusted.

PLSA_AP_CALL_PACKAGE_PASSTHROUGH
CallPackagePassthrough

Caller is another package in LSASS

Common Provider Code

```
mov     rcx, rbx
mov     rdi, [rbx]
mov     rax, LsaFunctions
mov     rax, [rax+30h]
call    __guard_dispatch_icall_fptr
```

What are the *LsaFunctions*?

Package Initialization

SpInitializeFn callback function (ntsecpkg.h)

The SpInitialize function is called once by the Local Security Authority (LSA) to provide a security package with general security information and a dispatch table of support functions. The security package should save the information and do internal initialization processing, if any is needed.

Syntax

```
NTSTATUS SpInitializefn(
  [in] ULONG_PTR PackageId,
  [in] PSECPKG_PARAMETERS Parameters,
  [in] PLSA_SECPKG_FUNCTION_TABLE FunctionTable
)
{...}
```

```
typedef struct _LSA_SECPKG_FUNCTION_TABLE {
  PLSA_CREATE_LOGON_SESSION CreateLogonSession;
  PLSA_DELETE_LOGON_SESSION DeleteLogonSession;
  PLSA_ADD_CREDENTIAL       AddCredential;
  PLSA_GET_CREDENTIALS      GetCredentials;
  PLSA_DELETE_CREDENTIAL    DeleteCredential;
...
};
```

https://learn.microsoft.com/en-us/windows/win32/api/ntsecpkg/nc-ntsecpkg-spinitializefn

Getting Client Information

LSA_GET_CLIENT_INFO callback function (ntsecpkg.h)

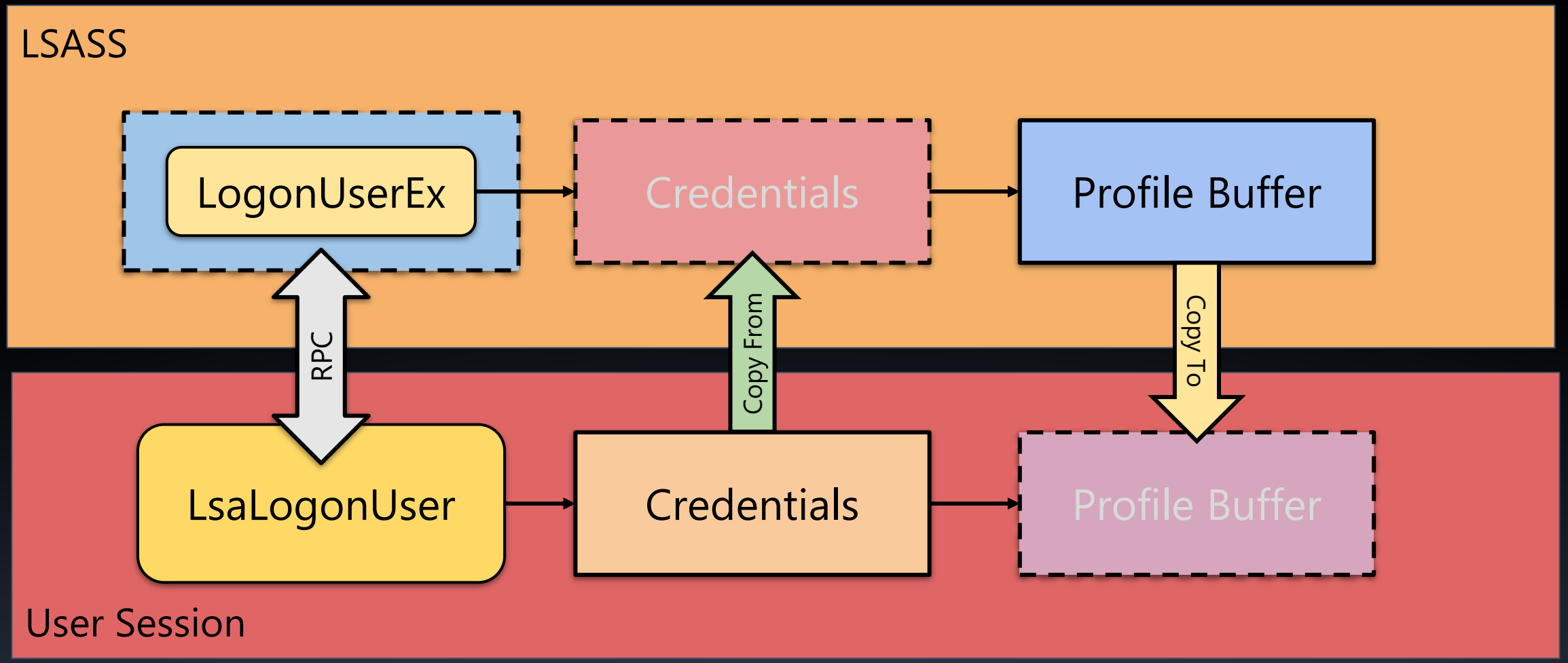
The GetClientInfo function gets information about the client process, such as thread and process ID, and flags indicating the client's state and privileges.

Syntax

```
NTSTATUS LsaGetClientInfo(
  [out] PSECPKG_CLIENT_INFO ClientInfo
)
```

```
struct SECPKG_CLIENT_INFO {
  LUID                        LogonId;
  ULONG                         ProcessID;
  ULONG                         ThreadID;
  BOOLEAN                        HasTcbPrivilege;
  BOOLEAN                        Impersonating;
  BOOLEAN                        Restricted;
  UCHAR                       ClientFlags;
  SECURITY_IMPERSONATION_LEVEL ImpersonationLevel;
  HANDLE                       ClientToken;
};
```

Copy Parameters to/from Client



Copy Parameters to/from Client

```
NTSTATUS LsaCopyFromClientBuffer(
  [
  [
  [
  [
)
```

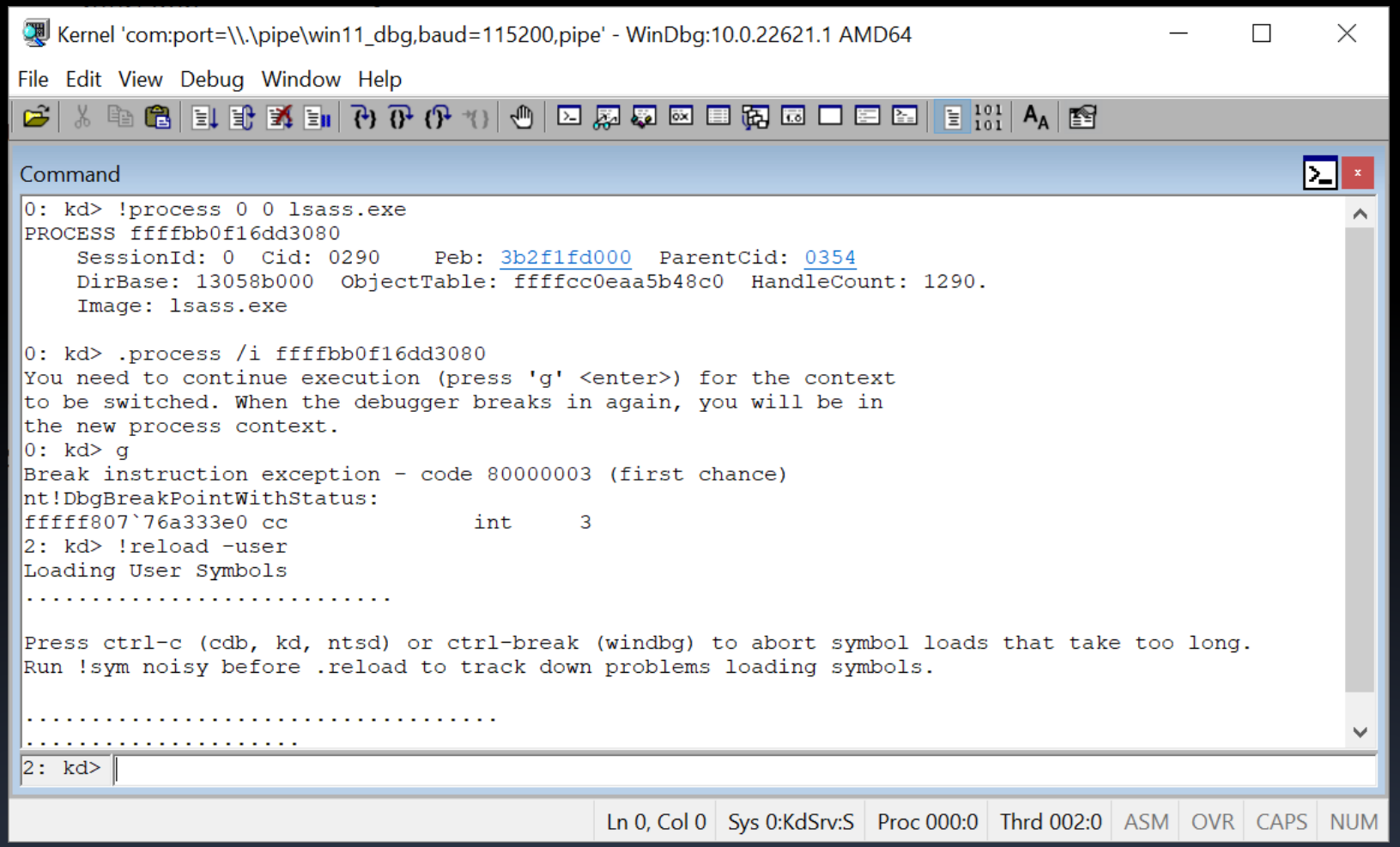
```
NTSTATUS LsaCopyToClientBuffer(
  [in] PLSA_CLIENT_REQUEST ClientRequest,
  [in] ULONG Length,
  [in] PVOID ClientBaseAddress,
  [in] PVOID BufferToCopy
)
```

Used for LogonUser and CallPackage

Used for Network Authentication

```
NTSTATUS LsaMapBuffer(
  [in]  PSecBuffer InputBuffer,
  [out] PSecBuffer OutputBuffer
)
```

Debugging LSASS



Research Results

Vulnerabilities Reported and Fixed

Bug Class	*Count*
Arbitrary Code Execution	3
Authentication Bypass	3
Authorization Bypass	4
Information Disclosure	4

CVE-2022-30164

Kerberos AppContainer Capability Bypass EoP (Authorization Bypass)

AppContainer Network Authentication

```
PS> $token = Get-NtToken -AppContainer -PackageSid ABC
PS> Invoke-NtToken $token {
>>   $cred = New-LsaCredentialHandle -Package 'Kerberos' -UseFlag Outbound
>>   $ctx = New-LsaClientContext -CredHandle $cred -Target CIFS/FILESERVER
>> }
Exception calling "CreateClient" with "5" argument(s):
"(0xC000005F) - A specified logon session does not exist. It may already have been
terminated."
```

Windows domain credentials enable a user to log into remote resources using their credentials, and act as if a user provided their user name and password. The ***enterpriseAuthentication*** capability is typically used in line-of-business apps that connect to servers within an enterprise.

https://learn.microsoft.com/en-us/windows/uwp/packaging/app-capability-declarations#restricted-capability-list

Kerberos Provider Messages

KERB_PROTOCOL_MESSAGE_TYPE enumeration (ntsecapi.h)

The KERB_PROTOCOL_MESSAGE_TYPE enumeration lists the types of messages that can be sent to the Kerberos authentication package by calling the LsaCallAuthenticationPackage function.

Syntax

```
enum KERB_PROTOCOL_MESSAGE_TYPE {
  KerbDebugRequestMessage = 0,
  KerbQueryTicketCacheMessage,
  KerbChangeMachinePasswordMessage,
  KerbVerifyPacMessage,
  KerbRetrieveTicketMessage,
  KerbUpdateAddressesMessage,
...
```

https://learn.microsoft.com/en-us/windows/win32/api/ntsecapi/ne-ntsecapi-kerb_protocol_message_type

Retrieve Encoded Ticket

```
struct KERB_RETRIEVE_TKT_REQUEST {
  KERB_PROTOCOL_MESSAGE_TYPE MessageType;
  LUID                       LogonId;
  UNICODE_STRING             TargetName;
  ULONG                      TicketFlags;
  ULONG                      CacheOptions;
  LONG                       EncryptionType;
  SecHandle                  CredentialsHandle;
};
```

Set to KerbRetrieveEncodedTicketMessage

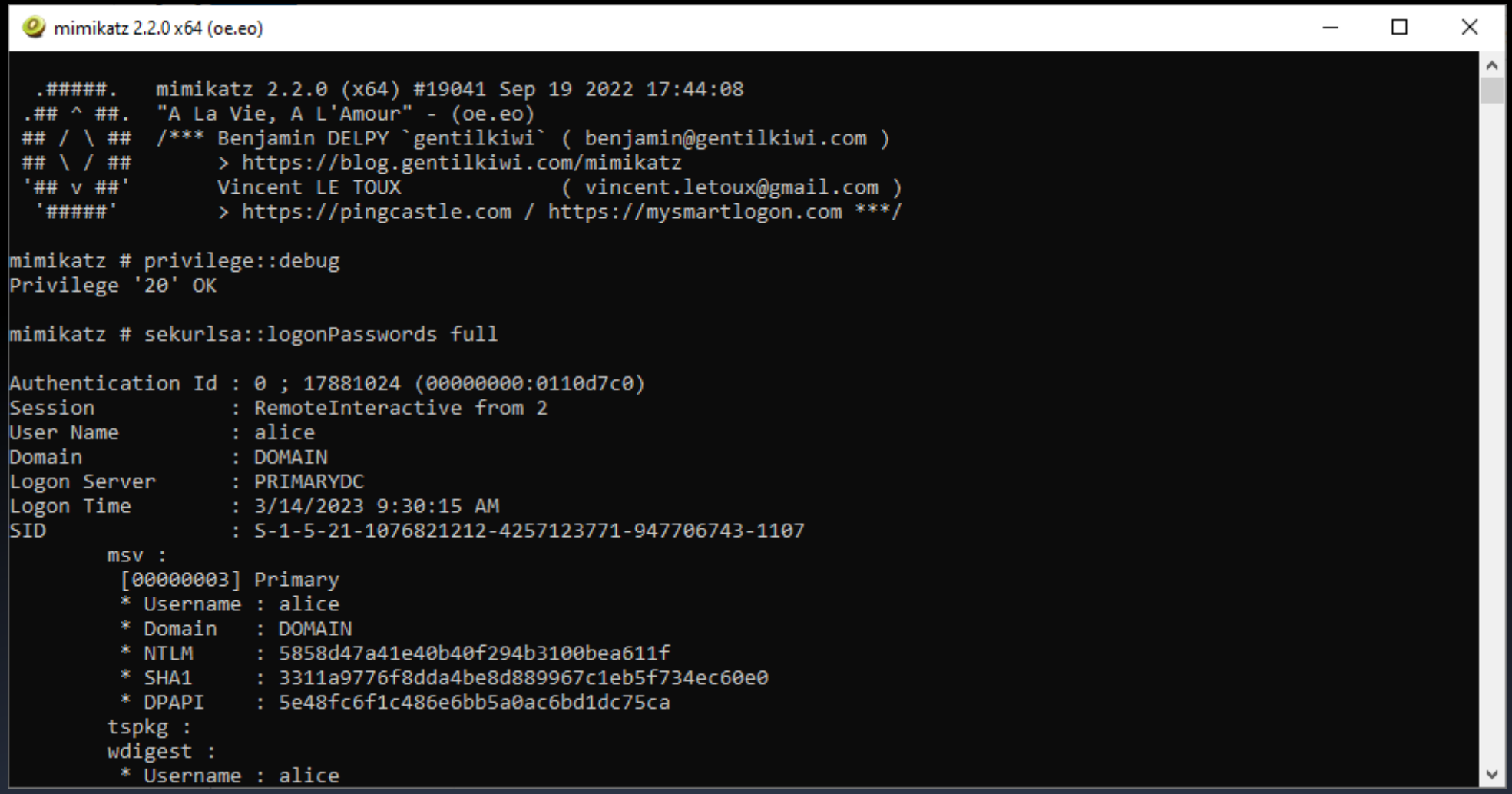
Set to arbitrary SPN

DEMO

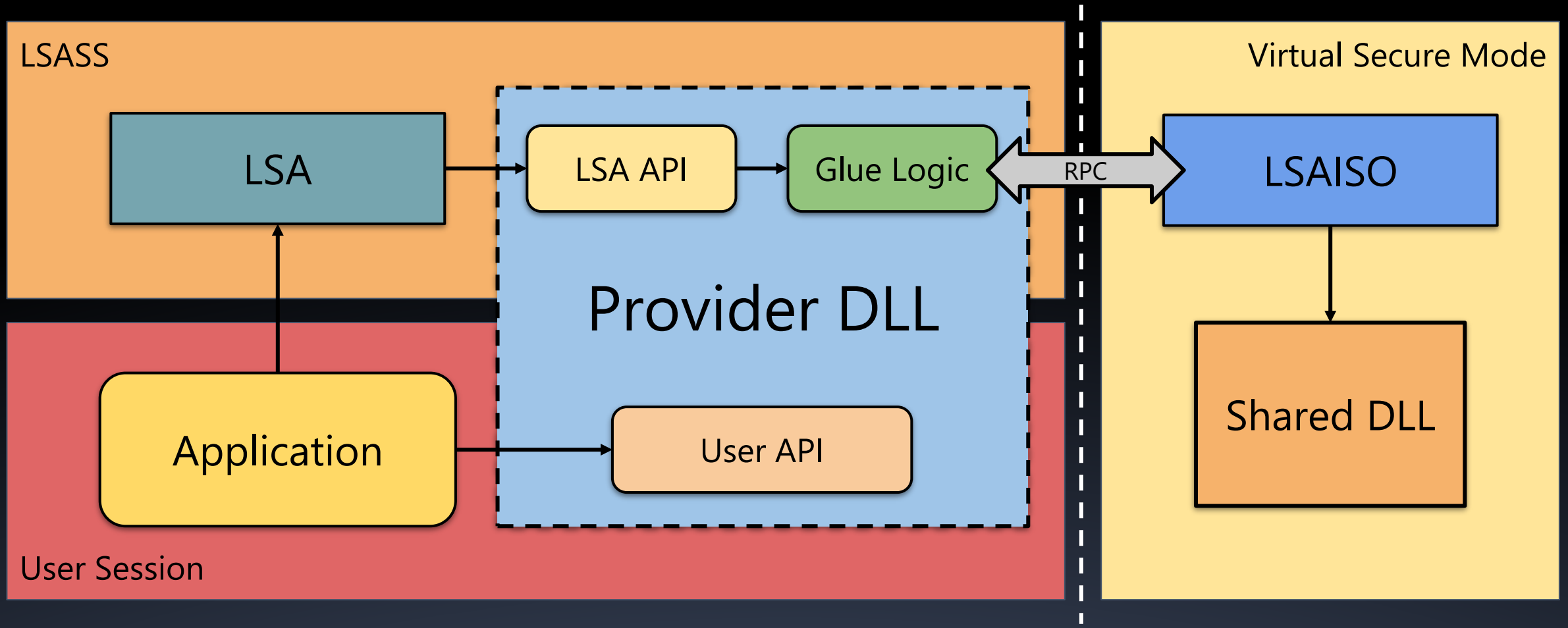
CVE-2022-34705

Credential Guard BCrypt Context Use-After-Free

Everyone Loves Mimikatz



Credential Guard



Credential Guard RPC Interfaces

```
PS> Get-RpcServer C:\windows\system32\LsaIso.exe
Name        UUID                                 Ver Procs EPs Service Running
----        ----                                 --- ----- --- ------- -------
LsaIso.exe a3e5af3e-8a33-4737-af6e-bc1f8ecee4bf 1.0 5     0           False
LsaIso.exe 39730ec4-82ea-4fdf-8a45-c408e393e212 1.0 2     0           False
LsaIso.exe eda3c9e4-0d4c-4bb7-b612-0e89d4f0607d 1.0 1     0           False
LsaIso.exe 57cce375-4430-47a6-bb96-2cad0d2fd140 1.0 26    0           False
LsaIso.exe 9cfeead6-6135-4fcf-831a-fd3b236023f8 1.0 33    0           False
LsaIso.exe 45527ae0-2a7d-4cec-b214-739f4159c392 1.0 19    0           False
LsaIso.exe 1707e621-44e3-4f54-bb7d-c537eabb55a5 1.0 3     0           False

PS> Get-Item 'NtObject:\RPC Control\LSA_ISO_RPC_SERVER'
Name                 TypeName
----                 --------
LSA_ISO_RPC_SERVER ALPC Port
```

BCrypt RPC Implementation

```
NTSTATUS BCryptIumOpenAlgorithmProvider(BCRYPT_ISO_OBJECT **obj,
  LPCWSTR pszAlgId, ULONG dwFlags) {
  BCRYPT_ALG_HANDLE hAlgorithm;
  BCryptOpenAlgorithmProvider(&hAlgorithm, pszAlgId,
                              NULL, dwFlags);
  *obj = AllocateBCryptIsoObject(hAlgorithm);
  return STATUS_SUCCESS;
}
```

BCrypt RPC Implementation

```
NTSTATUS BCryptIumOpenAlgorithmProvider(BCRYPT_ISO_OBJECT **obj,
  LPCWSTR pszAlgId, ULONG dwFlags) {
  BCR
  BCr

  *ob
  ret
}
```

```
BCRYPT_ISO_OBJECT *AllocateBCryptIsoObject(
            BCRYPT_ALG_HANDLE hAlgorithm) {
  BCRYPT_ISO_OBJECT* result = LocalAlloc(0,
                                sizeof(BCRYPT_ISO_OBJECT));
  result->Magic = 'BIOM';
  result->hAlgorithm = hAlgorithm;
  return result;
}
```

BCrypt RPC Implementation

```
NTSTATUS BCryptIumCloseAlgorithmProvider(BCRYPT_ISO_OBJECT *obj,
                                         ULONG dwFlags) {
  if (!obj || obj->Magic == 'BIOM') {
    return STATUS_INVALID_HANDLE;
  }
  BCryptCloseAlgorithmProvider(obj->hAlgorithm, dwFlags);
  LocalFree(obj);
  return STATUS_SUCCESS;
}
```

BCrypt RPC Interface Definition

```
typedef [context_handle] void* PBCRYPT_HANDLE_TYPE;

[uuid("57cce375-4430-47a6-bb96-2cad0d2fd140"), version(1.0)]
interface BCryptInterface {
    HRESULT BCryptIumGetClientContext([out] PBCRYPT_HANDLE_TYPE* p0);
    HRESULT BCryptIumReleaseContext([ref] PBCRYPT_HANDLE_TYPE* p0);
    HRESULT BCryptIumOpenAlgorithmProvider([in] handle_t p1, [out] int64_t* p1,
        [in] wchar_t* p2,  [in] wchar_t* p3,  [in] int p4);
    …
    HRESULT BCryptIumCloseAlgorithmProvider(handle_t p0, handle_t p1,
                                            [in] int64_t p1, [in] int p2);
    …
}
```

Context Handles

NOT Context Handles!

Debugging LSAISO

```
c:\windows\system32\lsaiso.exe -CredGuard
```

Command line to run as Credential Guard.

```
bp rpcrt4!RpcServerUseProtseqEpW "eb @r8 58; g"
```

Modify RPC endpoint from ***LSA_ISO_RPC_SERVER*** to ***XSA_ISO_RPC_SERVER***

```
eb LsaIso!StartDisableCredentialGuardWatcher 31 c0 c3
eb LsaIso!SignalLsaIsoReady 31 c0 c3
```

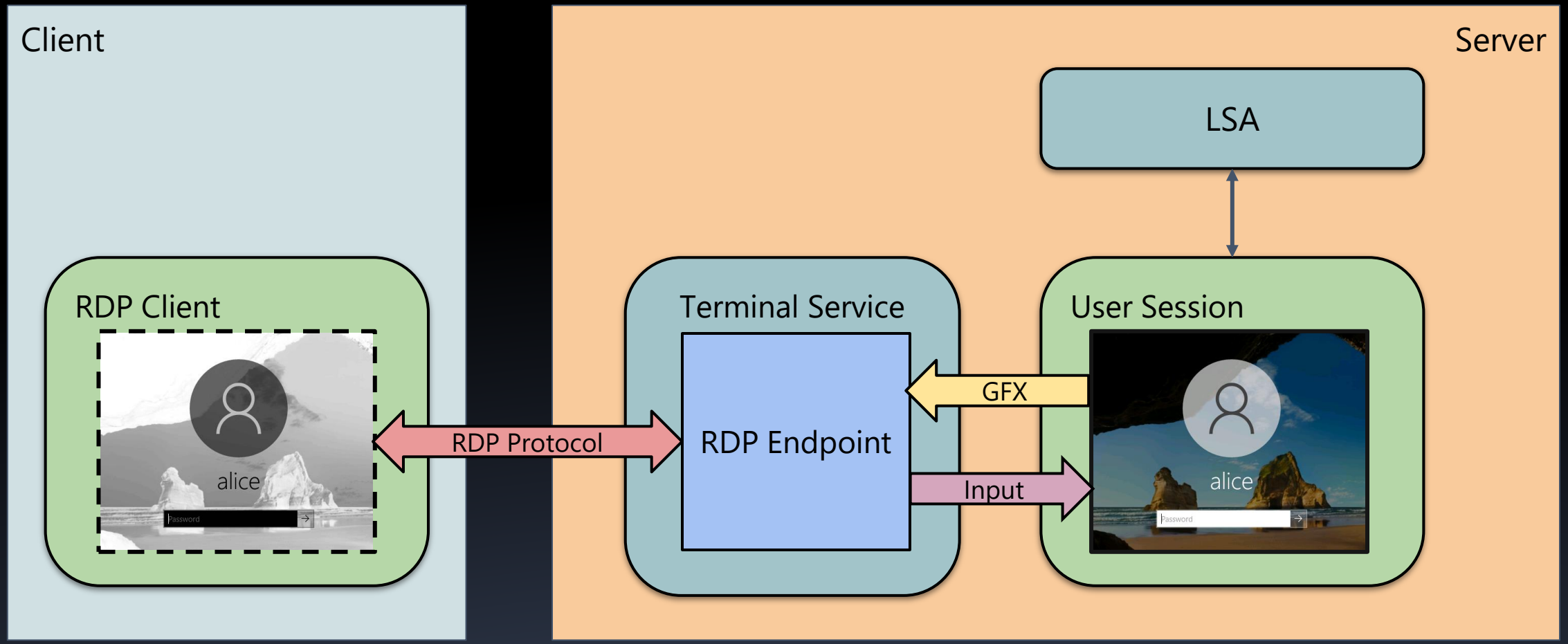
Patch out functions which won't work as non-admin.

DEMO

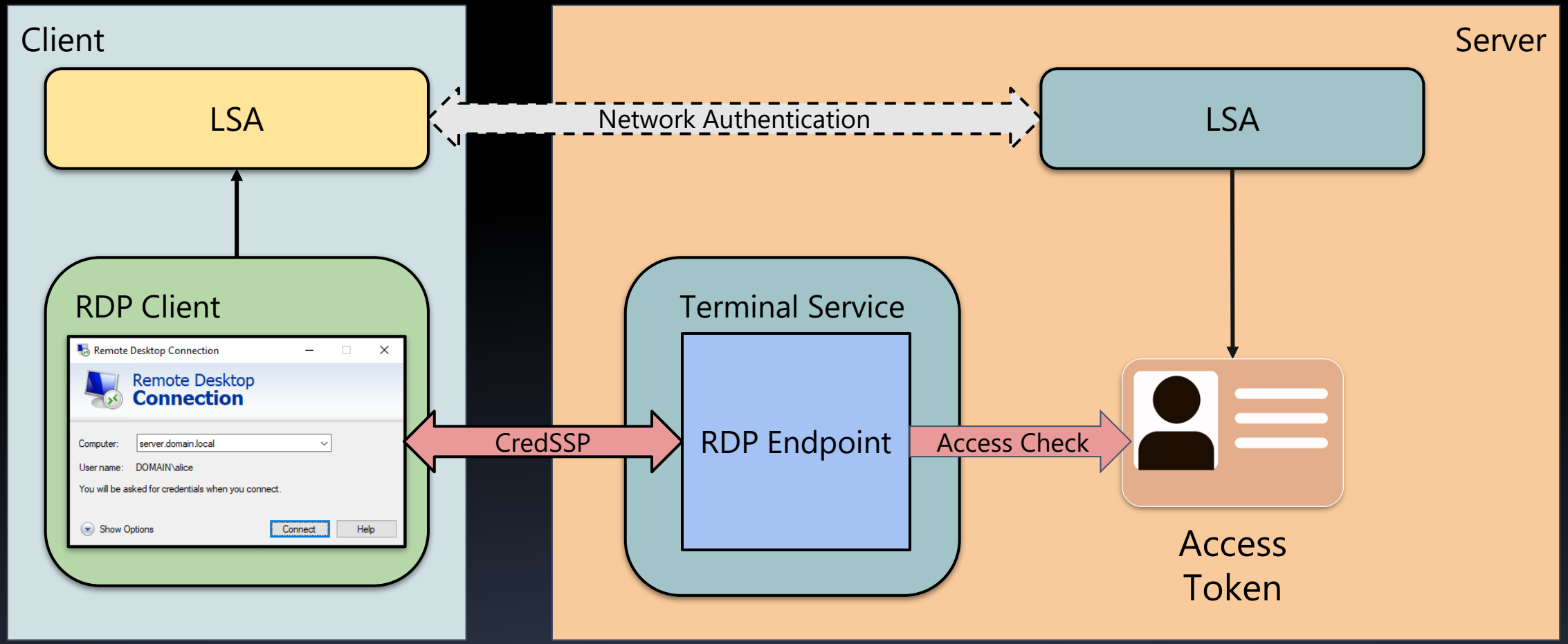
CVE-2022-30165

Remote Credential Guard Password Buffer (RCE)

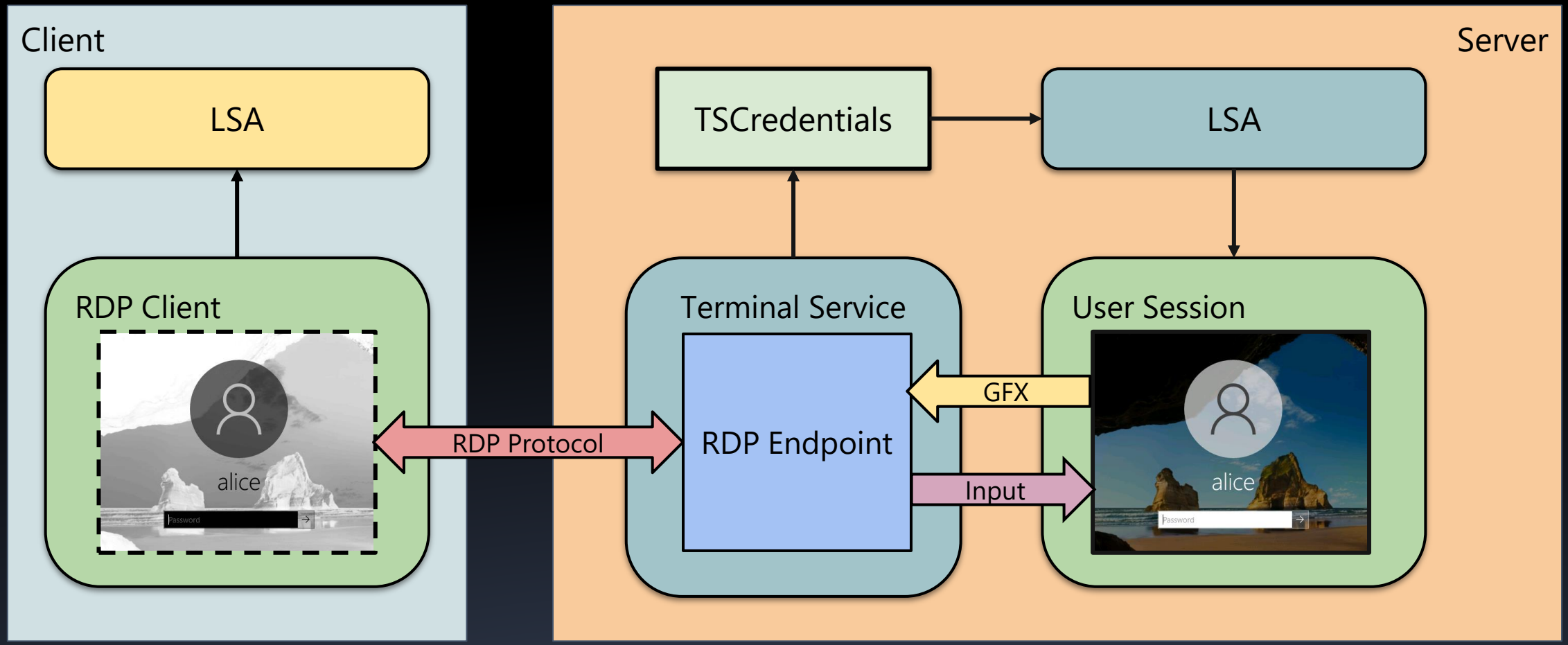
Remote Desktop in the Old Days



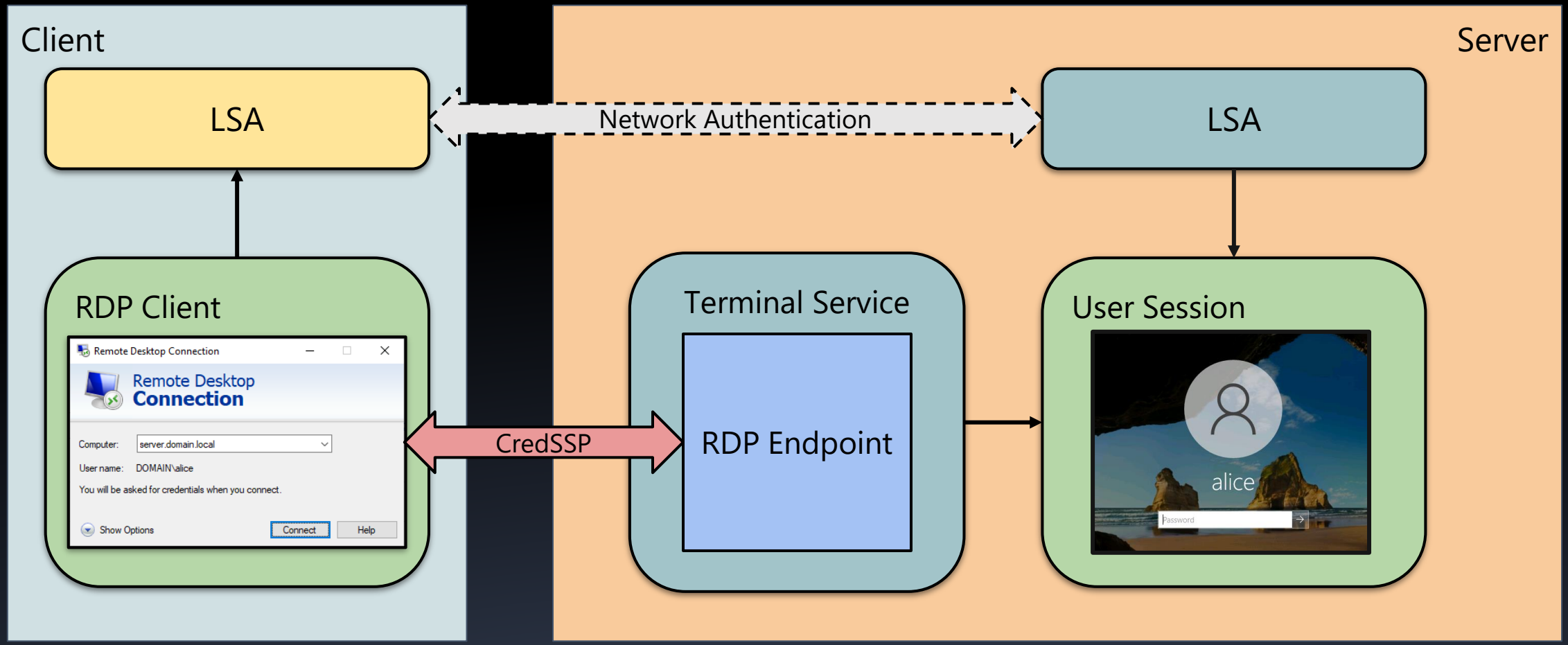
Network Level Authentication (CredSSP)



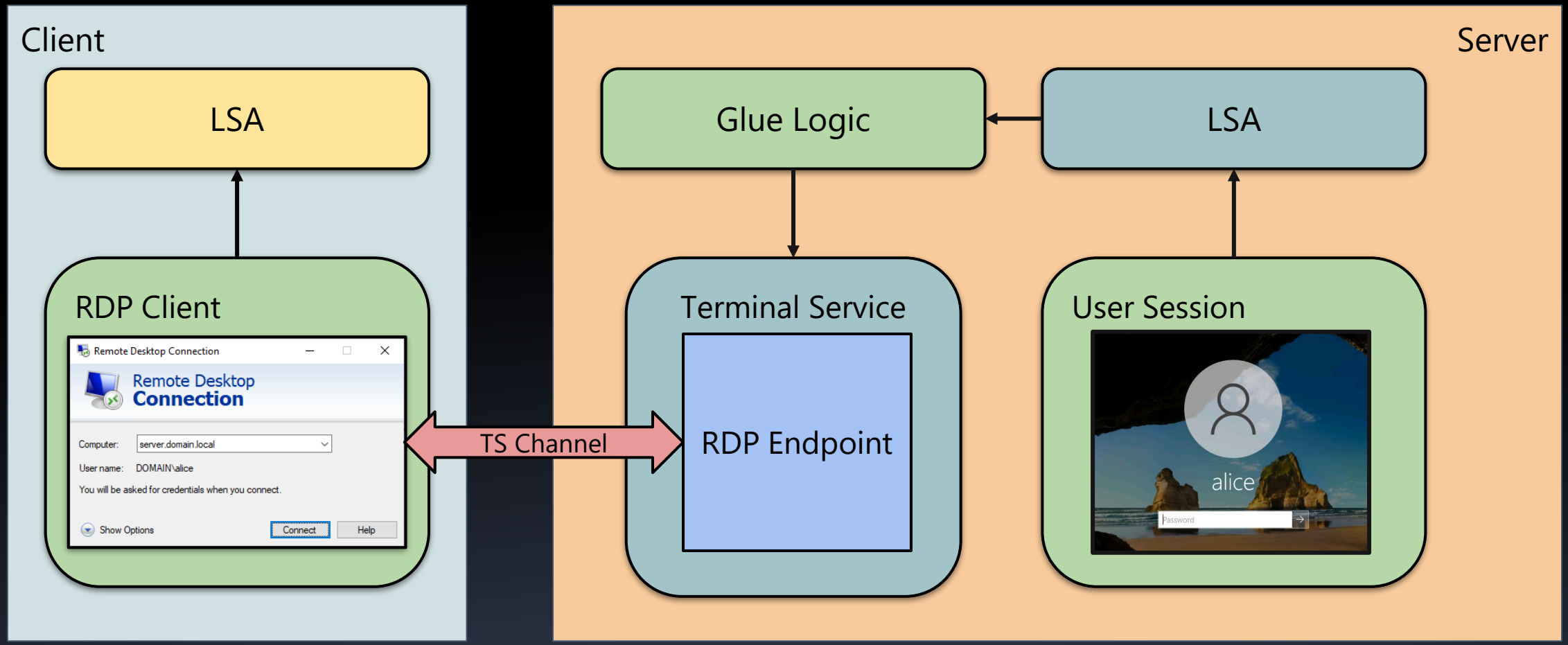
Network Level Authentication (CredSSP)



Restricted Admin Mode



Remote Credential Guard



Remote Guard Credentials

2.2.1.2.3.1 TSRemoteGuardPackageCred

The TSRemoteGuardPackageCred structure contains credentials for a specific security package.

```
TSRemoteGuardPackageCred ::= SEQUENCE{
        packageName [0] OCTET STRING,
        credBuffer  [1] OCTET STRING,
}
```

packageName: An ASN.1 OCTET STRING that contains the name of the package for which these credentials are intended.

credBuffer: An ASN.1 OCTET STRING byte buffer that contains the credentials in a format that SHOULD be specified by the CredSSP server operating system for the package that provided them.

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-cssp/173eee44-1a2c-463f-b909-c15db01e68d7

Accepting Remote Credentials (TSPKG)

```
SECPKG_REDIRECTED_LOGON_BUFFER data = {};
UNICODE_STRING Password { &data, sizeof(data),
                          sizeof(data) };

GUID RCG_MAGIC = {0xc2be5457,0x82eb,0x483e,
       {0xae,0x4e,0x74,0x68,0xef,0x14,0xd5,0x09}};
data.Guid = RCG_MAGIC;
data.Init = TSInitRedirectedLogon;
data.Cleanup = TSCleanupRedirectedLogon;

TSProtectPassword(&Password);
TSBuildCreds(&Result, L"User", L"Domain", &Password);
```

Build "password" using binary data including function pointers.

Encrypt buffer and build credentials

Authenticating with Remote Creds

```
if (Password.Length < sizeof(SECPKG_REDIRECTED_LOGON_BUFFER)) {
  return;
}

KerbUnpackLogonInfoPassword(&Password);
if (RtlCompareMemory(RCG_MAGIC, &Password.Buffer) != 16) {
  return;
}

SECPKG_REDIRECTED_LOGON_BUFFER *data = Password.Buffer;
data.Init();
```

Arbitrary function pointer dereference

Encrypting the Password

```
TSPasswordCreds ::= SEQUENCE {
  domainName   [0] OCTET STRING,
  userName     [1] OCTET STRING,
  password     [2] OCTET STRING
}
```

Uses the same encryption for a normal password

```
if (CredBuffer->Type == PasswordCreds) {
  TSPasswordCreds* creds = CredBuffer->Data;
  TSProtectPassword(&creds->password);
  TSBuildCreds(&Result, &creds->userName,
    &creds->domainName, &creds->password);
}
```

DEMO

Wrapping Up