Spooky authentication at a distance

Why I don’t care about your passwords

About me

Name: Tamas Jos

Day Job: SEC Consult, an Eviden business

Aliases: @skelsec, github.com/skelsec

Night Job:

- Open-source developer
- Creator of many security projects
- Member of Porchetta Industries

Bigger Projects-o-mine:

- pypyKatz
- minikerberos
- MsLDAP
- Aardwolf
- aioSMB
- antlmrelay
- OctoPwn
- Amurex

Other: Likes to go on tangents and rants about unrelated topics

The promise

By the end of this presentation you'll learn about a new technique that allows you to:

- Use the authentication context of a domain user, of a remote machine, from your own computer

Requirements:

- Small agent running on target machine
- No dumping of credentials, in fact we won't be seeing any plaintext credentials
- No administrator privileges required
- No special permissions required (e.g. special tokens etc.)
- No exploits, everything that will be shown here is "by-design"

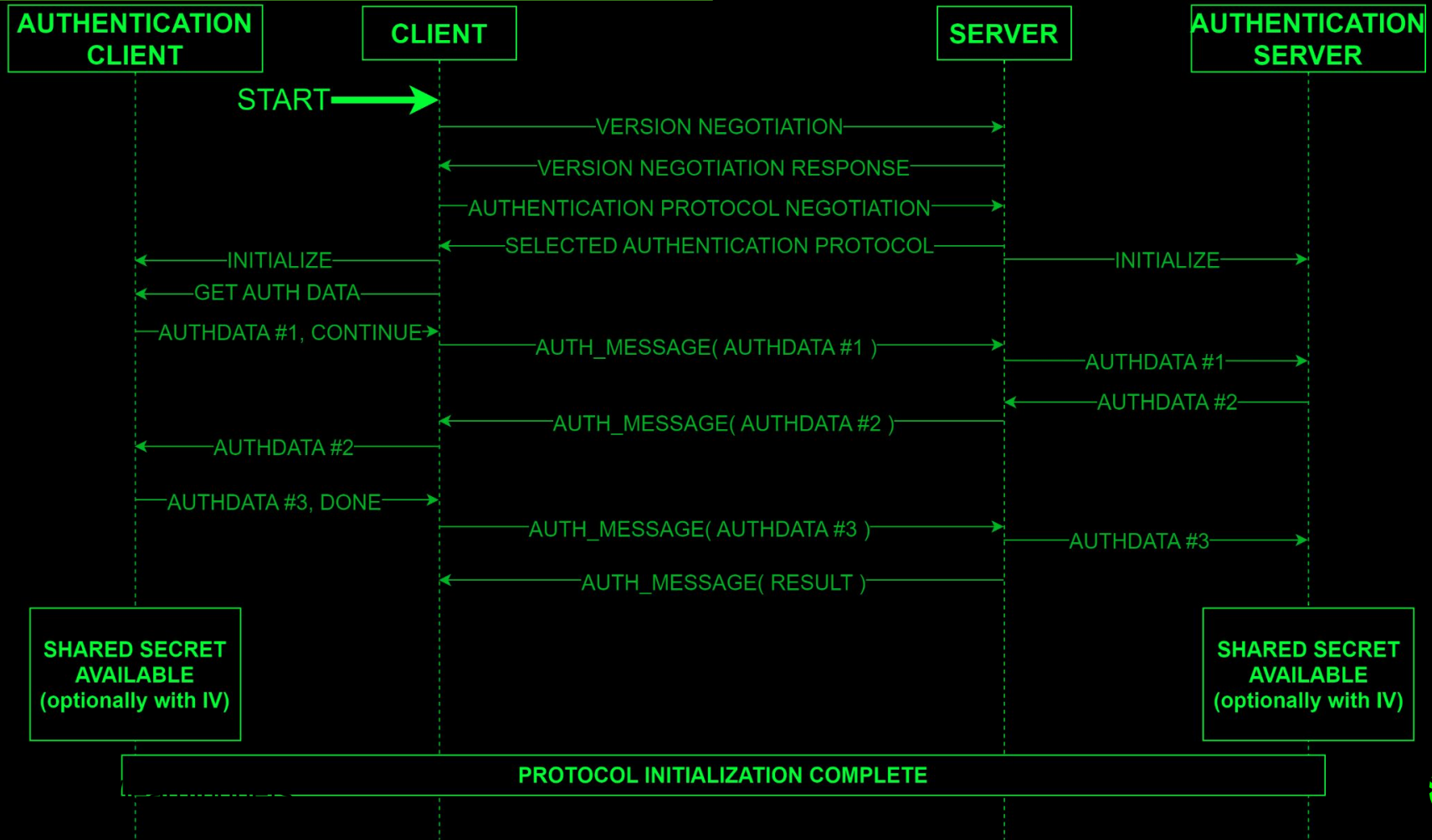
Summary

- Windows Authentication Protocols
  - General description
  - Authentication methods in LDAP (and a bit of SMB)
  - LADDERS SO MANY LADDERS
- What is SSPI
  - General description
  - How to use it
- How to authenticate locally, but remotely, but also locally…

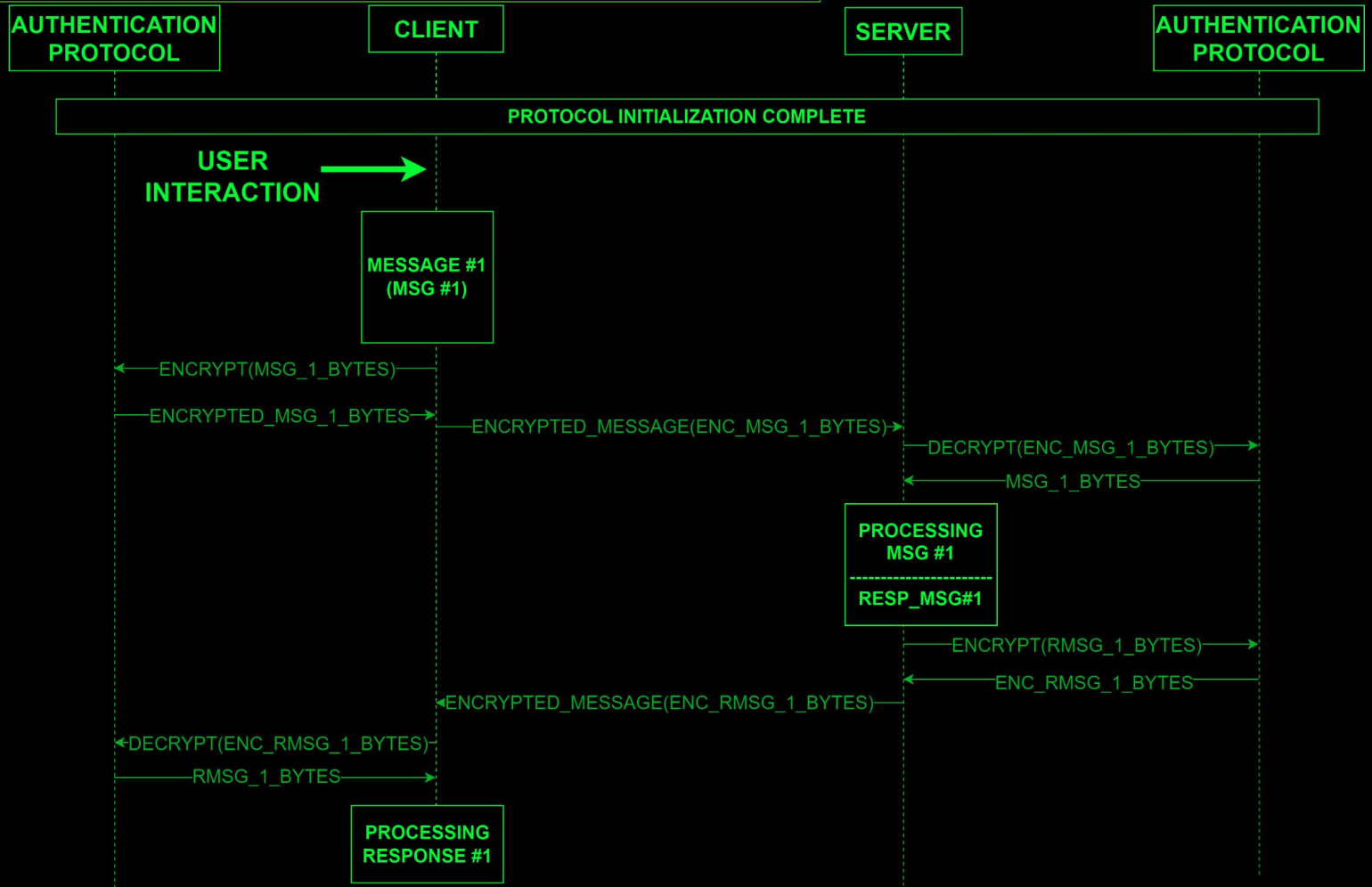
Windows Authentication Protocols

- In a nutshell -

Authentication protocols - Generalization



Authentication protocols - Encryption / Decryption



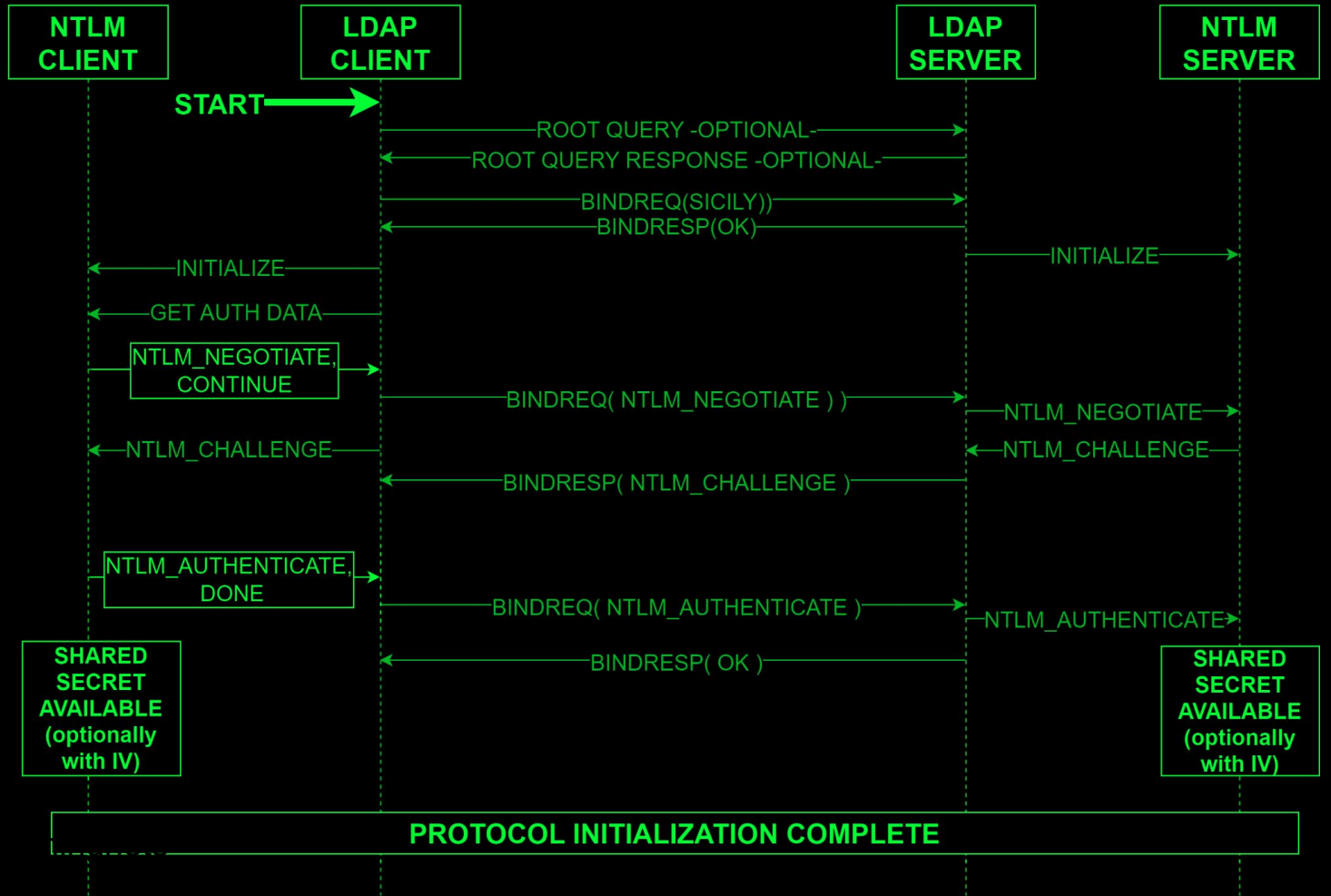
A word from the author

In the following part we are going to use LDAP as baseline to discuss authentication mechanisms relevant for this presentation

LDAP is widely used in Windows domains, and supports all major authentication types we need to touch

It also perfectly represents all evolutionary dead ends

LDAP - SICILY Authentication (NTLM)



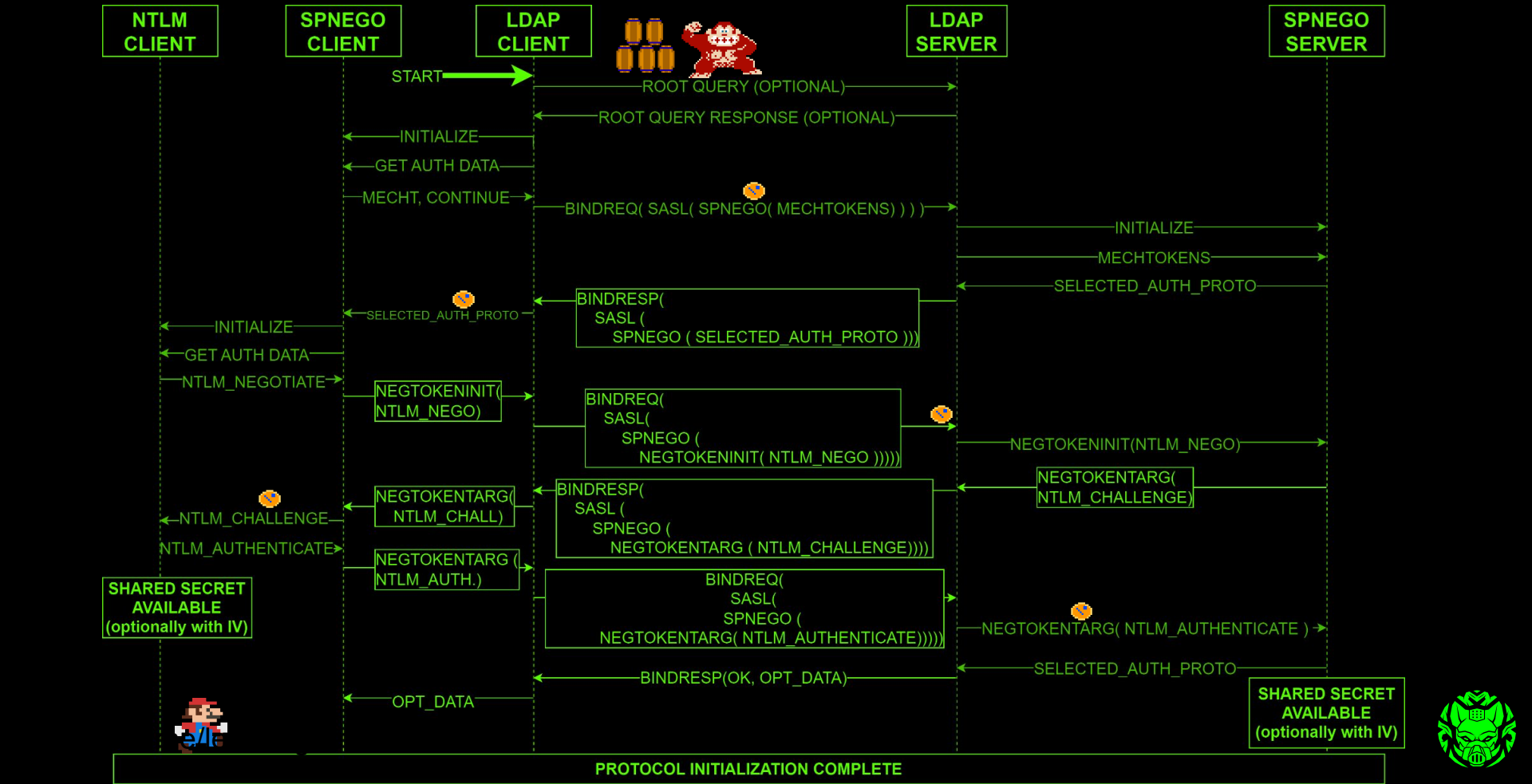
SPNEGO - The authentication negotiator

- SPNEGO stands for Simple and Protected GSSAPI Negotiation Mechanism
- Defined in RFC 2478
- Provides a way to negotiate the type of authentication used between a client and server
- This is usually (but not limited to) Kerberos or NTLM
- SPNEGO itself doesn't provide authentication; it simply selects the mechanism for the authentication
- SPNEGO is beneficial in environments where multiple authentication mechanisms are in use, as it allows for the selection of a common mechanism between client and server

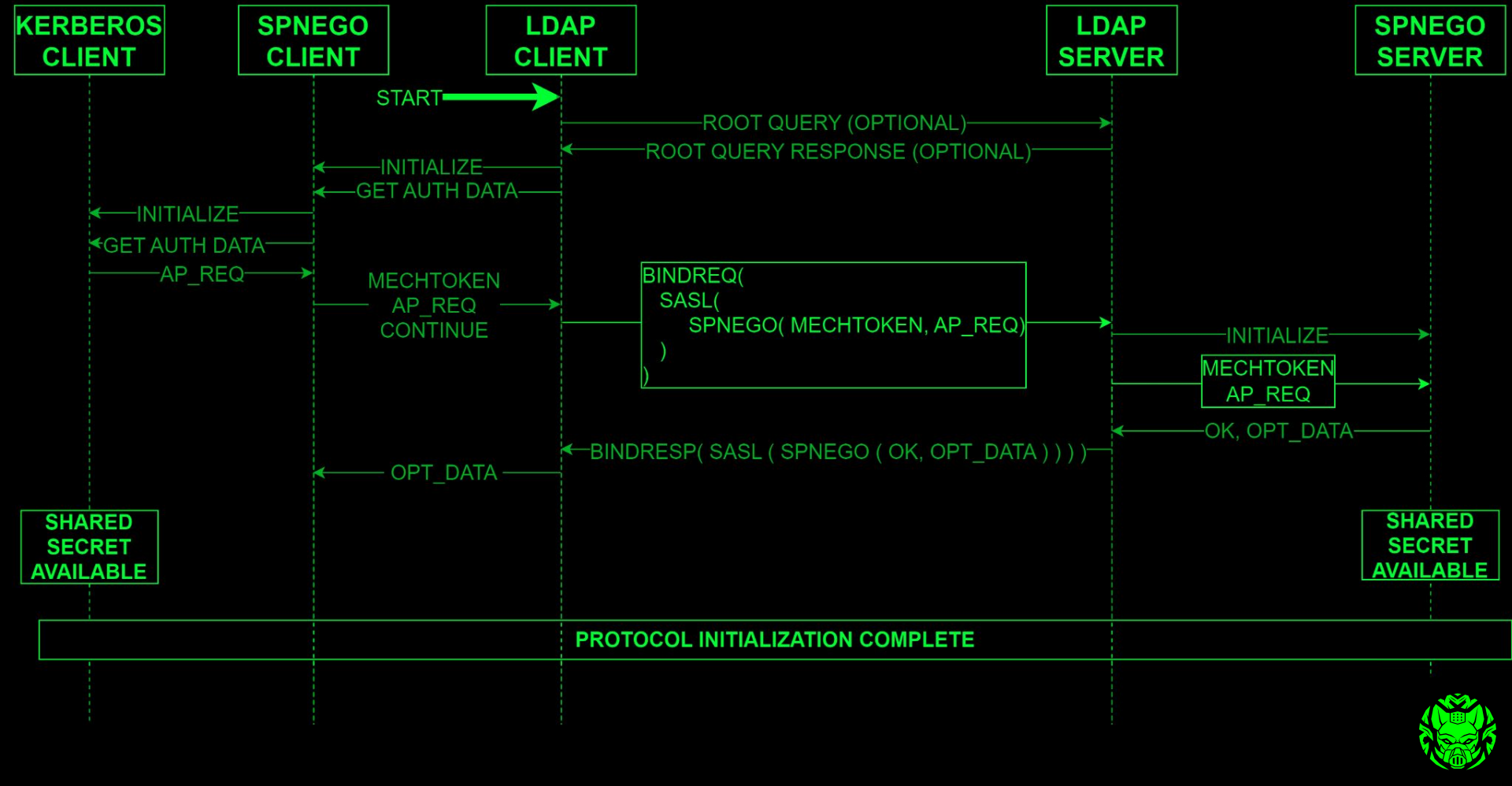
SPNEGO - In case of LDAP

- LDAP supports the following authentication solutions
  - SIMPLE:
    - NONE: No authentication (anonymous BIND)
    - PLAIN: User + password sent over the wire
  - SICILY: I don’t even… This is raw NTLM
  - SASL:
    - EXTERNAL: Uses client TLS/SSL certificate as authentication
    - SPNEGO: What we’re discussing now
    - GSSAPI: This again allows different authentication mechanisms to be used
    - DIGEST-MD5: Because it’s 1999
    - … (many others)

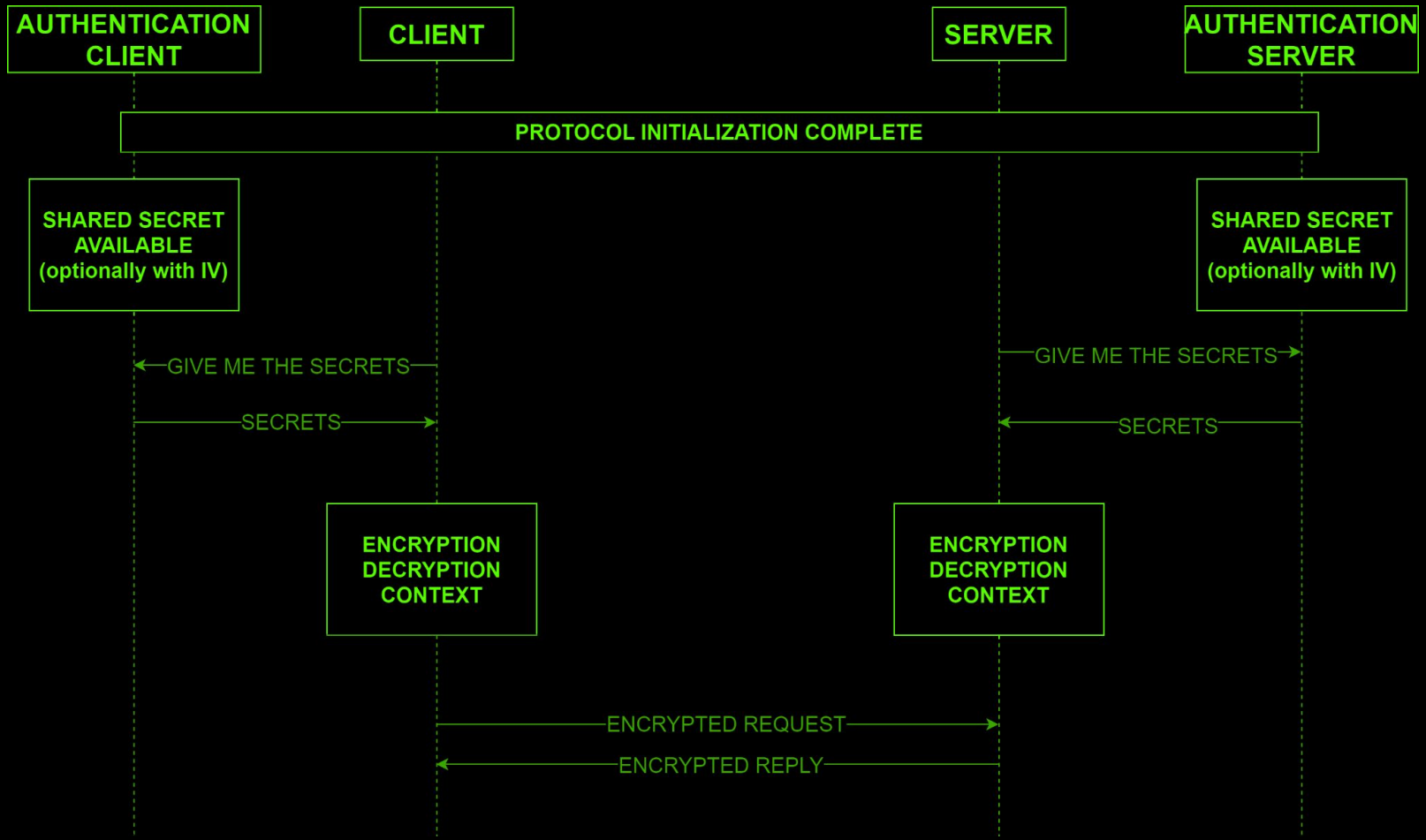
LDAP - SASL - SPNEGO - NTLM Authentication



LDAP - SASL - SPNEGO - Kerberos Authentication



SMB ENCRYPTION/DECRYPTION



Security Support Provider Interface (SSPI)

SSPI - Overview

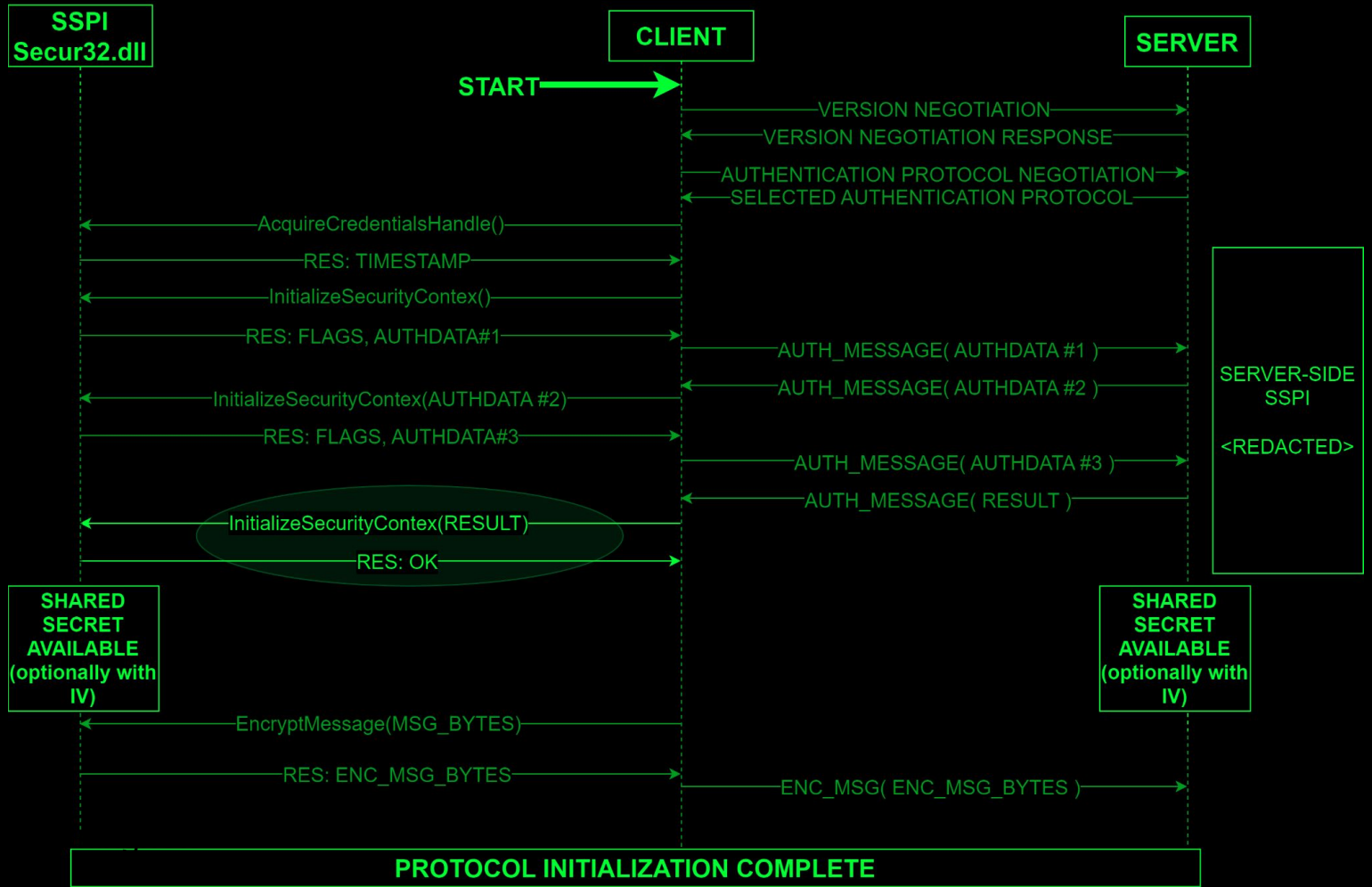
- Security Support Provider Interface
- API by Microsoft used in Windows systems
- SSPI allows applications to use various security models available on a computer or network without changing the interface to the security system
- The key function of SSPI is to provide a framework that abstracts the specifics of individual security models
- It supports various security models like Kerberos, NTLM, Schannel (for SSL/TLS) and others
- SSPI provides a mechanism to connect, sign on or off, encrypt or decrypt messages, and validate authority
- It works with security support providers (SSPs), which are dynamic-link libraries (DLLs), that make one or more security packages available to applications

SSPI - Methods

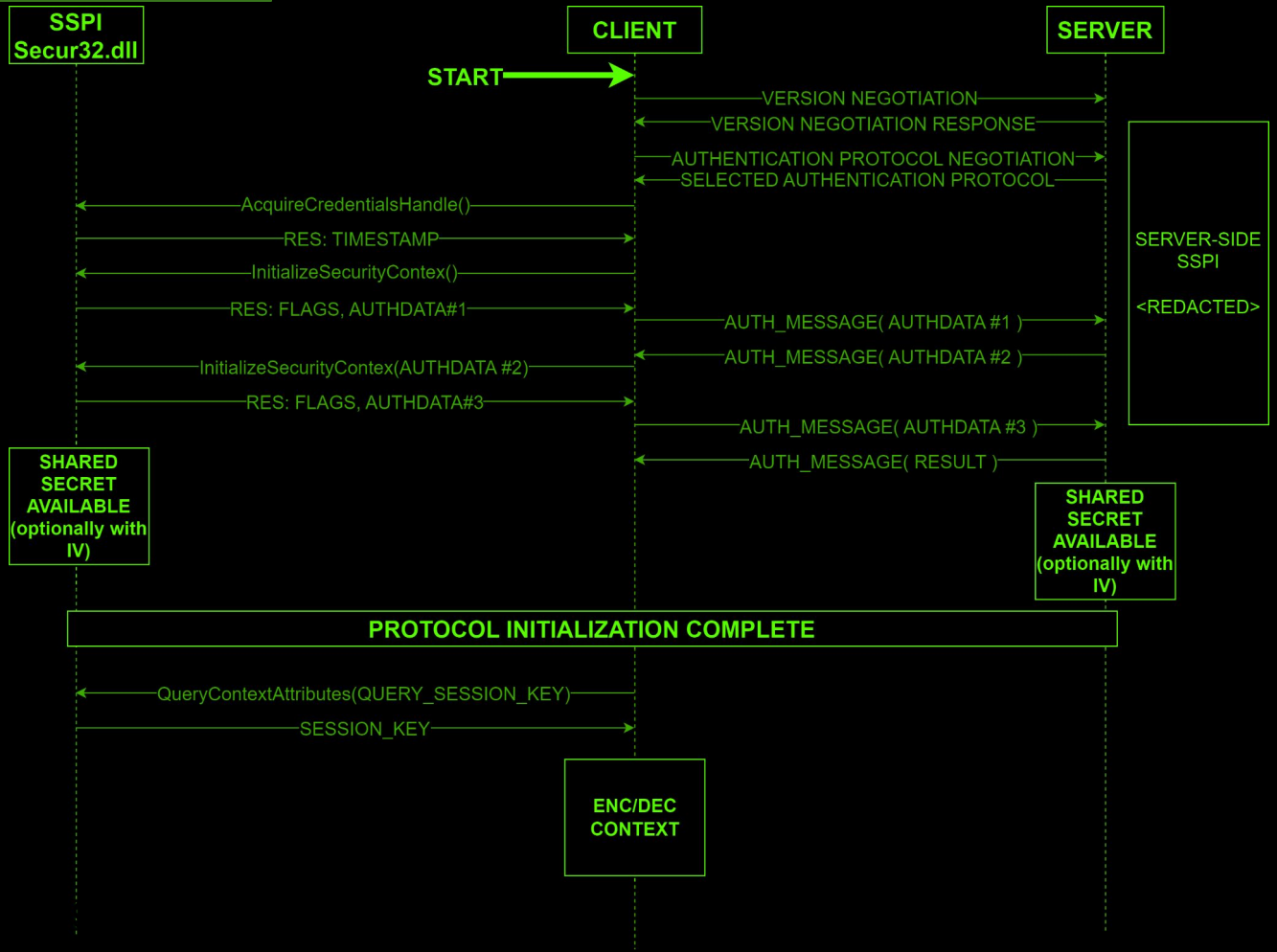
For a basic client authentication via SSPI, the following methods can be used:

- AcquireCredentialsHandle
- InitializeSecurityContext
- QueryContextAttributes
- EncryptMessage
- DecryptMessage

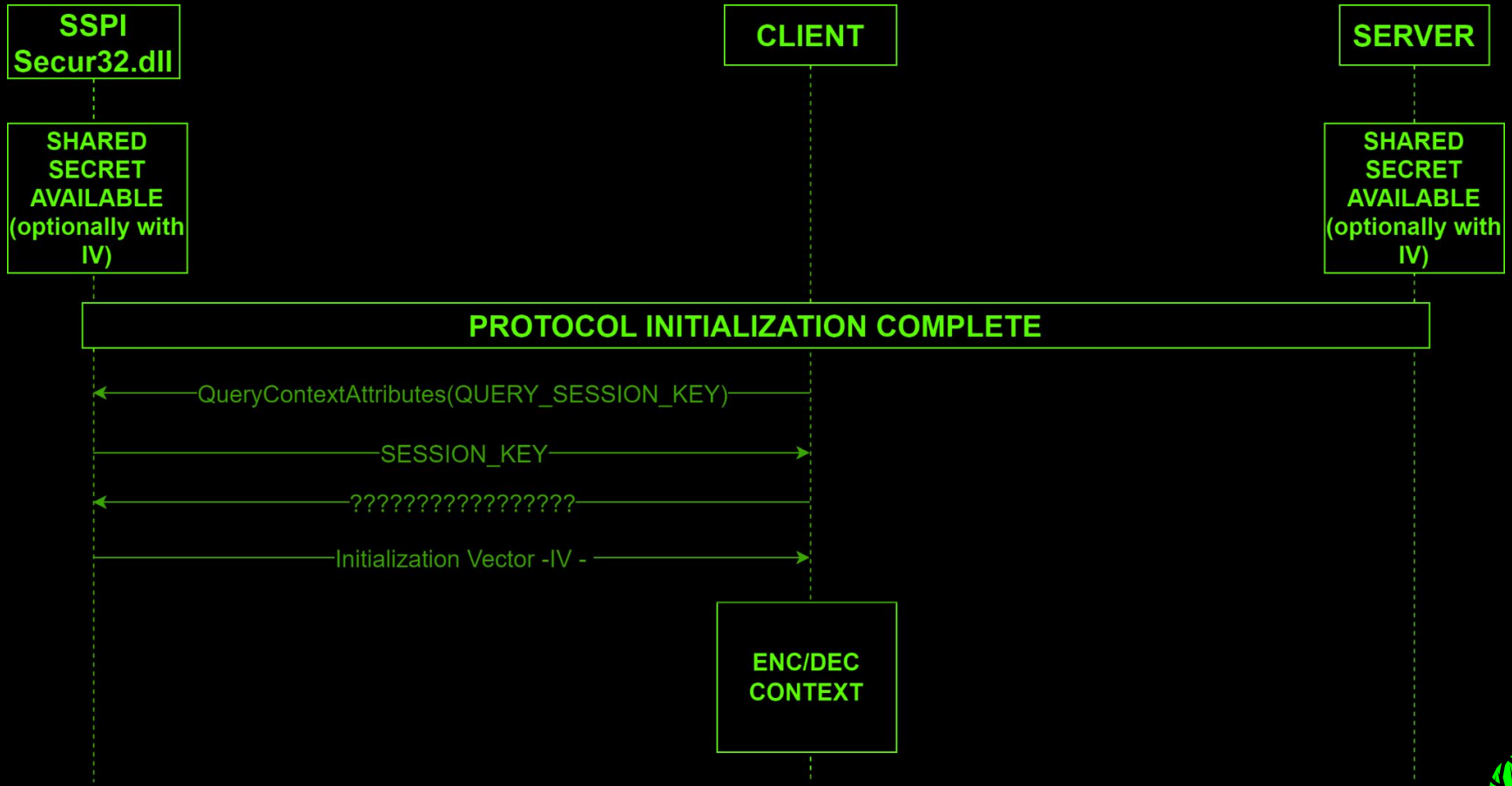
SSPI - Generalization



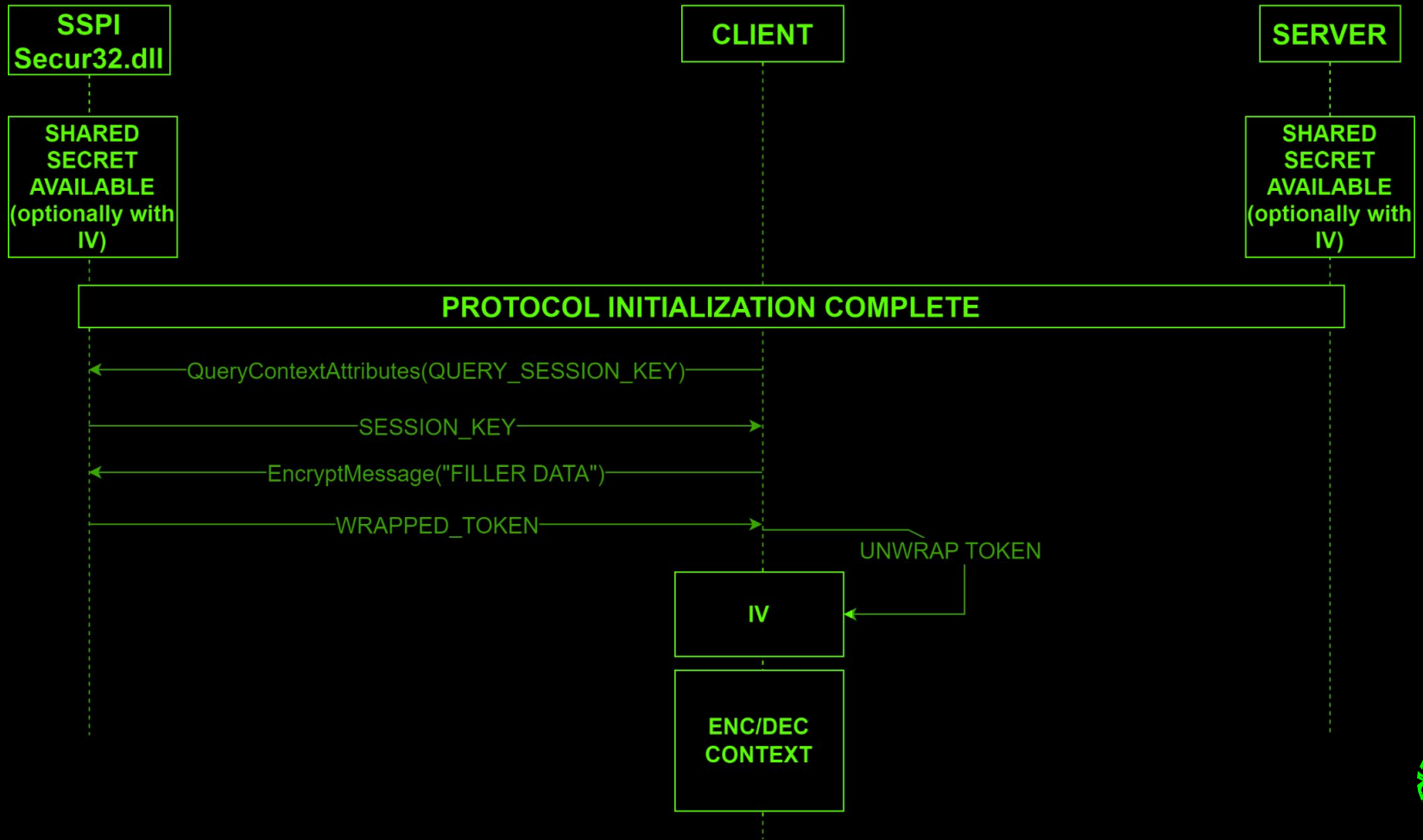
SSPI - How to use it SMB



SSPI - But there is a problem with mutual authentication



SSPI - Solution for mutual authentication



Your LSASS, but on **MY** machine

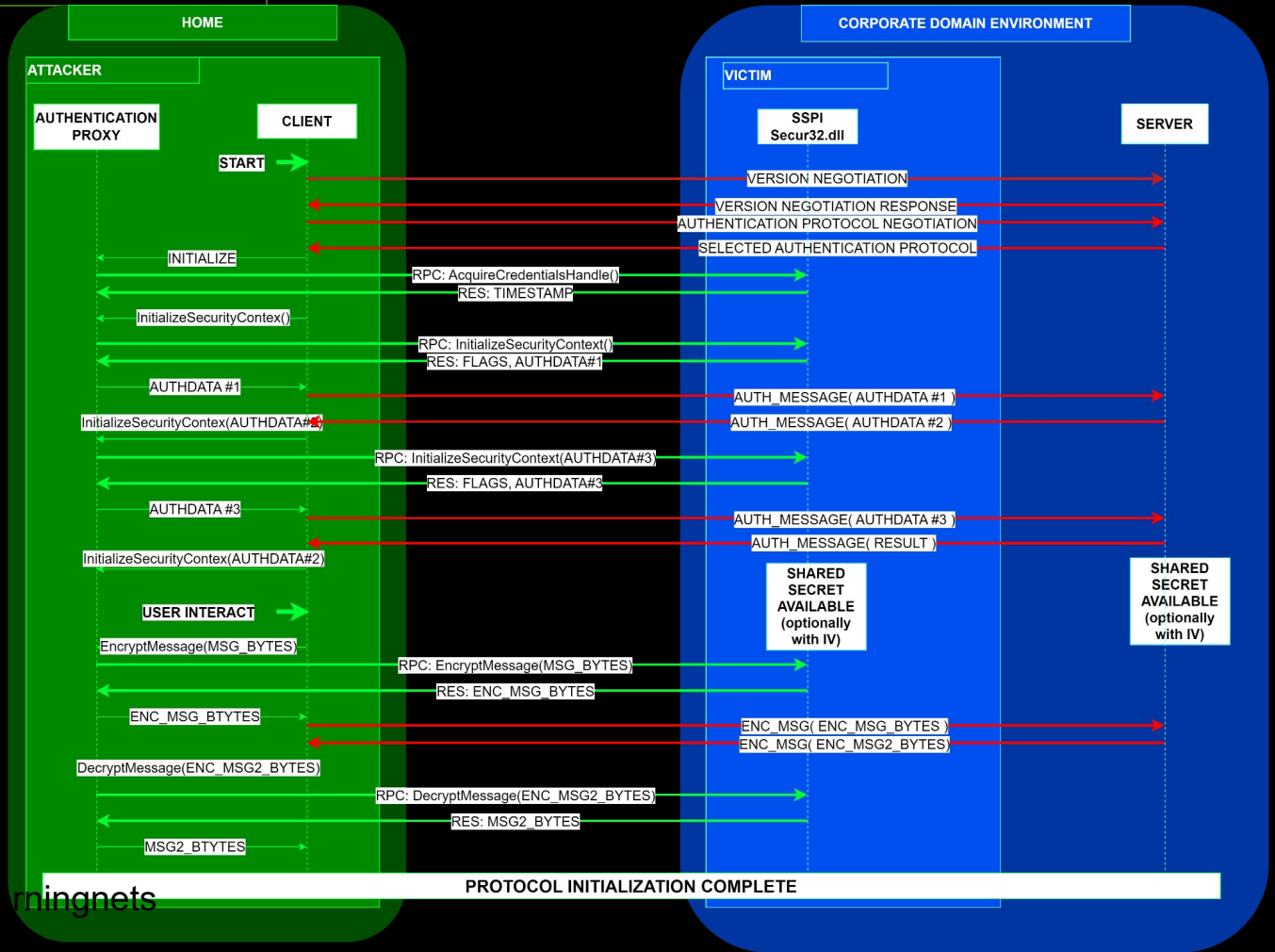
SSPIProxy - The goal

Imagine a system where one would be able to use SSPI function calls remotely:

- Authentication on behalf of current user context
- SSPI does not require admin privilege to perform authentication
- No need to touch LSASS directly (e.g. no dumping)
- The proxy process doesn’t do anything suspicious
- After authentication finishes, the authentication context on the target machine is not needed to continue the communication on higher-layers

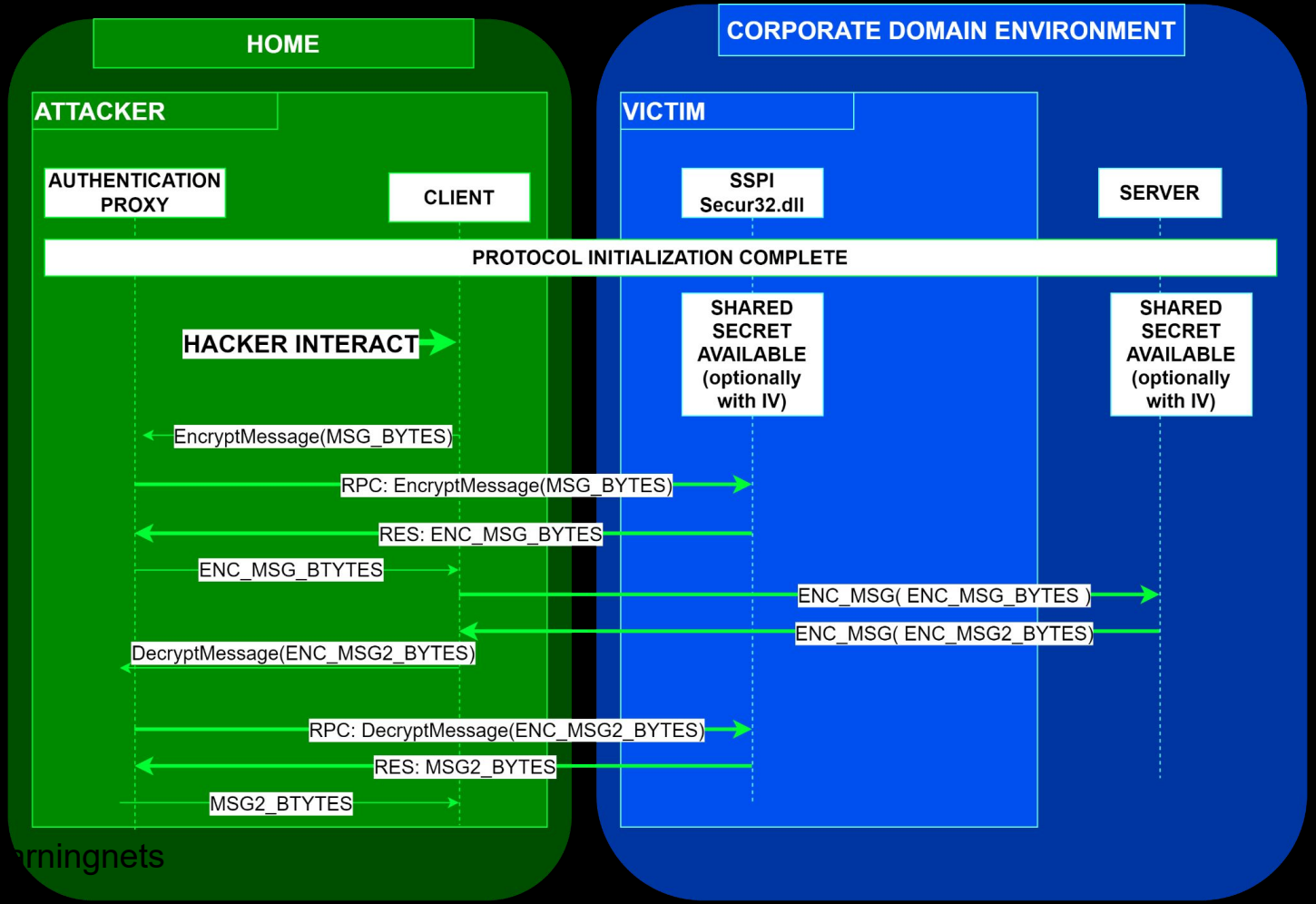
If implemented correctly, the authentication proxy would allow authentication from a 3rd party machine (e.g. teamserver) to a server on the internal network of the target environment, **WITHOUT** additional code to be pushed on the victim machine besides the initial agent

SSPIProxy - Generic logic

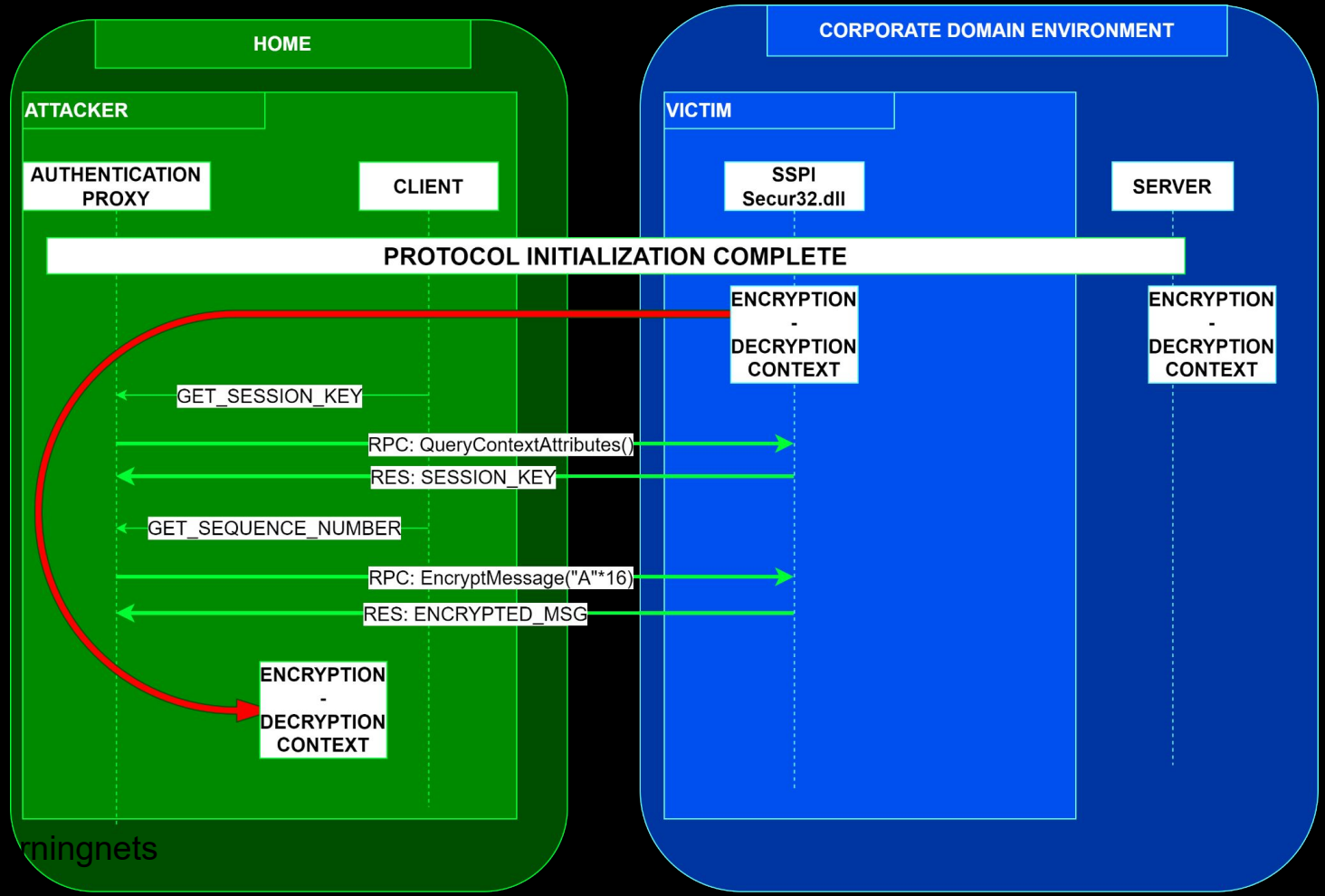


rningnets

SSPIProxy - Encryption/Decryption the wrong way



SSPIProxy - Encryption/Decryption the right way



SSPIProxy - Basic use case

SSPIProxy’s agent side can be implemented easy in any programming language

For this use case I’ve created a project called WSNET, which on the agent side starts a basic listener socket

Attacker can connect to this server via AioSMB/MSLDAP/other tools and utilize the victim’s authentication context remotely on the attacker’s machine


ME
SSPI
PROXY
JR. REDTEAMER
DOMAIN USER
CROWDSTRIKE
LSASS
https://t.me/learningnets
imgflip.com

DEMO

SSPIProxy - A different use case I

SSPIProxy technique plays very well with small embedded systems.

This use case describes how to use a reverse connecting SSPIProxy implementation on a Raspberry Pi Zero 2W device to attack an unlocked Windows workstation.

The following method can be used on any USB-OTG embedded system

Ingredients

- 1x RpiZ2W device
- 1x ZeroStem Pi Zero USB Stem or similar
- 1x SSPIProxy reverse connecting agent
- 1x USB-Ethernet emulation (Rpi)
- 1x USB HID (keyboard) emulation (Rpi)
- 1x DHCP server (Rpi)
- 1x C2 server running (Rpi)
- 1x SMB or HTTP server (Rpi)

SSPIProxy - A different use case II

Follow these steps

Upgrade the Rpi, install Samba, Nginx, C2 server and the Ducky autotyper service

You might want to set up an AccessPoint (optional)

Set up the USB-HID and USB-Ethernet emulation on the Rpi

Set up all networking services to use the appropriate ethernet device

Solder the USB stem to the Raspberry Pi board, and put the result in a plastic case.

Find an unlocked Windows workstation (domain enrolled) and plug in the device

SSPIProxy - a different use case III

What is going to happen

- Rpi operating system starts
- USB host initializes
- Target Windows system detects USB plugged in, and find the Keyboard and Network device automatically. (Default drivers used, no need to fetch new ones)
- New network device gets an IP address automatically from the DHCP server
- Ducky autotyper takes the emulated keyboard and loads the SSPIProxy agent
- Agent starts, and connects back to the C2 which is on the RPi device
- At this point the C2 server can utilize the authentication provided by the SSPIproxy agent and automatically start attacking services inside of the domain without additional code being executed on the target machine

DEMO

SSPIProxy - Current status

Requires an agent + libraries to use custom auth protocol implementation on server side.

SSPIProxy agent demo is implemented in a Python library called “WSNET[1]”. It is published under MIT license on Github.

Authentication protocol library called “asyauth[2]” is published under MIT license on Github.

All my major protocol implementation libraries use “asyauth” under the hood, thus they already have a way to support and use this technique

1 - https://github.com/skelsec/wsnet

2 - https://github.com/skelsec/asyauth

SSPIProxy - Comparison

Benefits of this solution:

- Can authenticate to network services without any exploits on the initial foothold
- No need to push additional code on the initial foothold
- Work from home
- AV/EDR detection is minimal
- The main complexity is in the code of the attacker machine -> Agent can be (re)implemented quite easily in other ways to avoid detection
- Drive the SoC insane - with multiple compromised hosts, you can trigger authentication from different workstations, but use a network connection from a workstation where the user can't even log in

Let's discuss some drawbacks:

- This technique (as of now) requires custom protocol implementations that allow interfacing with the authentication proxy. (Unless you are @_EthicalChaos_)
- Compared to traditional C2 solutions, this solution will generate much more network traffic, as it needs to invoke the RPC calls for the authentication proxy for each connection created to a target server

SSPIProxy - FUTURE

The SSPIProxy technique will keep on living in all my projects, I have big plans for it. See you at DEF CON 31 for more :)

- ALSO -

There is this person on Twitter by the handle of @_EthicalChaos_

- A few years ago, in unrelated research, he published a project called “lsarelayx[3]”
- It hooks LSASS itself to override/redirect NTLM authentication mechanism for ALL local windows applications
- This could be extended to support Kerberos/SPNEGO.
- It would allow any windows application (on the attackers machine) to leverage SSPIProxy functionality

3 - https://github.com/CCob/lsarelayxet

SSPIProxy - FUTURE

There is at least one more authentication method which could be implemented using SSPIProxy logic:

- Kerberos - PKINIT

By implementing 3 more RPC calls, we can proxy certificate based kerberos authentication using the target user’s certificate store

This is on my ROADmap to make @_dirkjan happy

Thank you

Shoutout to the following people who helped me - in no specific order -

@_EthicalChaos_ - LSARELAYX

@BoreanJordan - smbprotocol / python-gssapi

@awakecoding - sspi-rs

@agsolino - Impacket

Links to the projects discussed in this presentation:

https://github.com/skelsec/aiosmb

https://github.com/skelsec/msldap

https://github.com/skelsec/asyauth

https://github.com/skelsec/wsnet

https://github.com/skelsec/wsnet-dotnet

https://community.octopwn.com

Q & A

THANK YOU

Please sponsor us on Porchetta Industries!

https://porchetta.industries