Understanding MAPIProxy

Julien Kerihuel, <j.kerihuel@openchange.org>

Contents

1 Introduction

Introduction

What is MAPIProxy?

- Proxy server for ExchangeRPC traffic
- Can act as:
  - **Transparent/Intercepting proxy**:
    - Does not modify request/responses beyond what is required for authentication and identification
  - **Non-Transparent proxy:**
    - modifies the request or response in order to provide some added service to the user agent
  - **Forwarding proxy:**
    - Forward inbound/out-bound traffic
    - Cache results
- MAPI clients (Outlook, openchangeclient, etc.) consider MAPIProxy as the real Exchange server

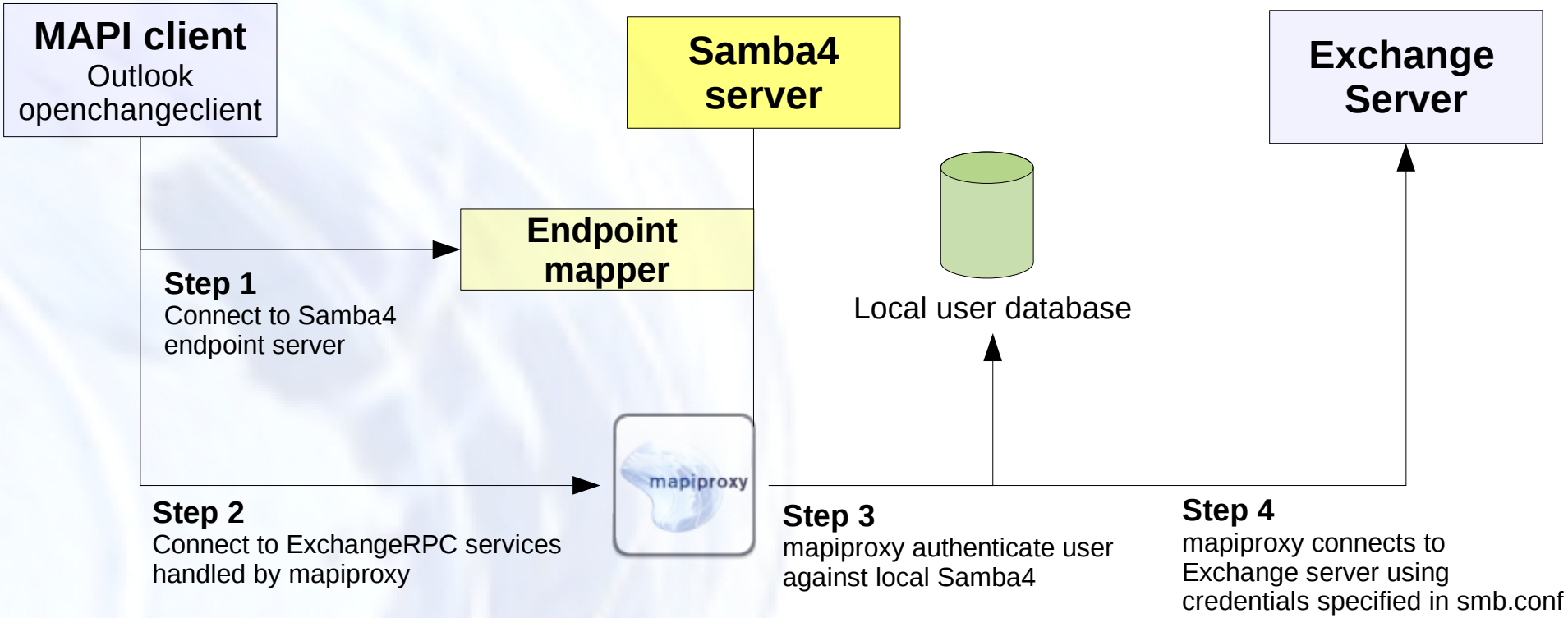
Introduction

What is MAPIProxy?

- Endpoint server for Samba4
- Initial server skeleton based on dcerpc_remote endpoint from Stefan Metzemacher
- Provides 2 kind of authentication mechanisms
  - **Specified credentials**
    - Client authentication is done by Samba4 local server:
      - **Dedicated credentials** are used to access the remote server:
        - Configure unique credentials (username, password, IP address and domain) in smb.conf to connect to remote Exchange server
        - A single account will be used to relay ExchangeRPC traffic from all clients
  - **Delegated credentials**
    - Credentials are forwarded to the real server
    - Remote server authenticate the user
    - Possible implementation:
      - NTLMSSP MITM
      - MS Kerberos

Introduction

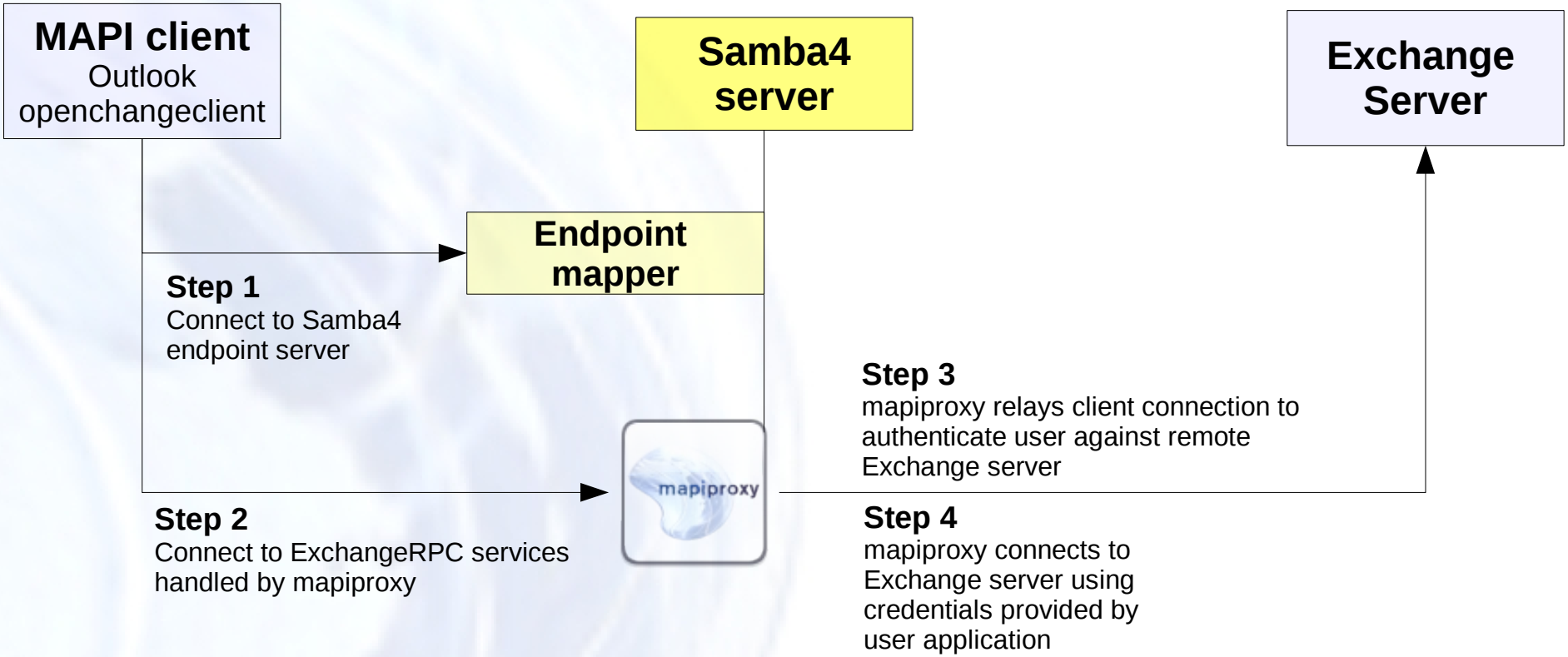
- **Specified credentials and single MAPIProxy instance** use case



- 2 connections:
  - **Mapi client to mapiproxy (using MAPI client credentials)**
  - **mapiproxy to Exchange Server (using credentials specified in smb.conf)**

1 Introduction

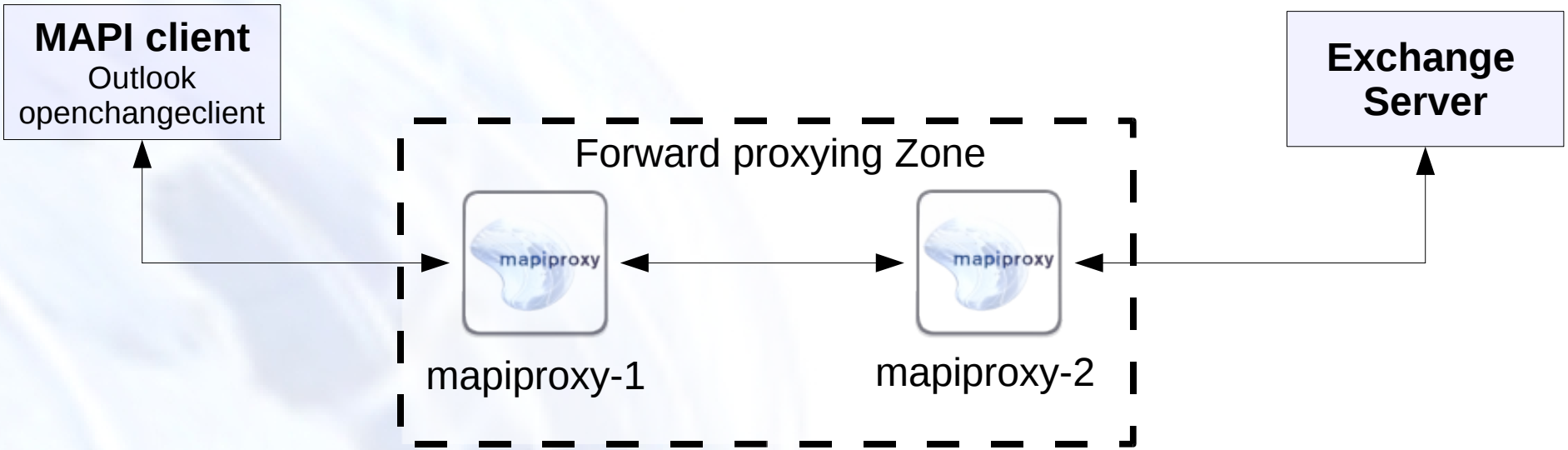
- **Delegated credentials and single MAPIProxy instance** use case



- 2 connections:
  - **Mapi client to mapiproxy**
  - **mapiproxy to Exchange Server**

Introduction

- **multiple MAPIProxy instances** use case



Forward proxying zone can be used to:
- **Compress** ExchangeRPC traffic (acceleration)
- **Encrypt** ExchangeRPC traffic

- 3 connections:
  - **Mapi client to mapiproxy-1**
  - **mapiproxy-1 to mapiproxy-2**
  - **mapiproxy-2 to Exchange Server**

Introduction

- **Why was MAPIProxy developed ?**
  - Writing a server is not trivial
  - While openchange MAPI library can test Exchange Server behavior, we had no similar tool to test Outlook behavior
  - Helps figuring out what is required/mandatory and what is optional (for preliminary OpenChange Server implementation)
  - Furthermore initially designed for MAPI acceleration purposes

2 Technical Concepts

2 Technical Concepts

- mapiproxy is an endpoint server for samba4

- DSO loaded by Samba4 if set in smb.conf:
  - **dcerpc endpoint servers = epmapper, mapiproxy**

- **epmapper** is the only Samba4 endpoint mapiproxy needs to run

- mapiproxy Handles 3 different protocols for complete ExchangeRPC support:
  - **exchange_rfr**: RFR protocol (find NSPI Server)
  - **exchange_nsp**: NSPI protocol (address book and name resolution)
  - **exchange_emsmdb**: EMSMDB protocol (exchange transport)

Technical Concepts

- **NSPI Referral replacement**
  - RFR is used by Outlook to “locate the NSPI server”
  - Outlook relies on the server address returned by **RfrGetNewDSA (0x0) RPC operation**
  - mapiproxy needs Outlook to believe it is the NSPI server
  - It means mapiproxy needs to replace the server name with its own

Technical Concepts

NSPI Bindings replacement

- When Outlook sets a new account, it uses the NSPI protocol:
  - Username resolution
  - Fetch Exchange information

- Information returned by Exchange is:
  - stored within Windows registry
  - Used to connect directly to the EMSMDB pipe

- **If passing through, Outlook will try to connect to the real Exchange server rather than MAPIProxy**

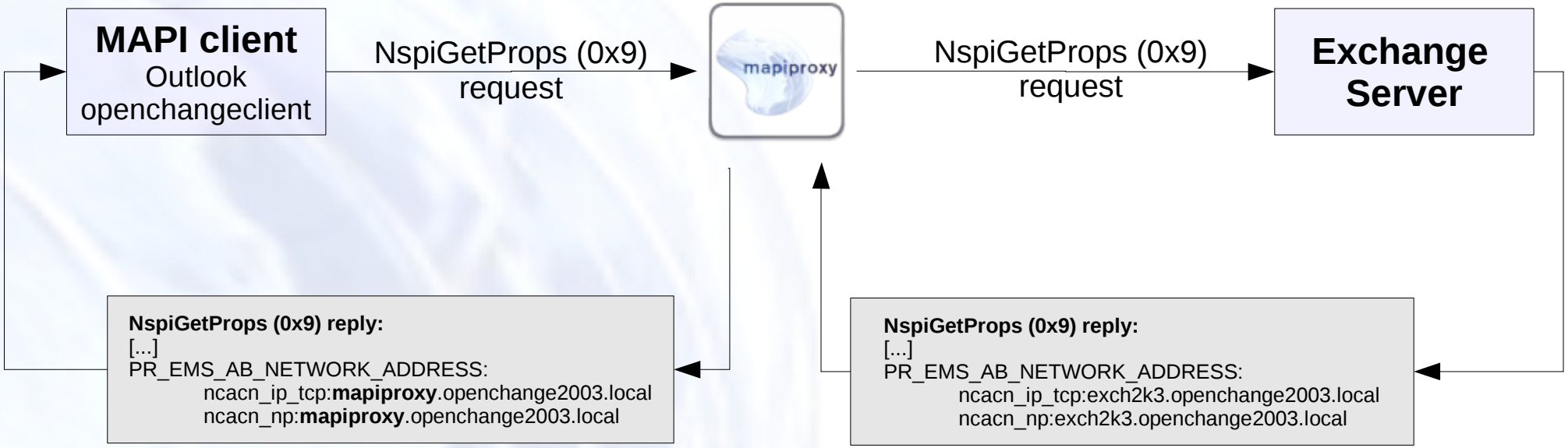
- mapiproxy needs to alter NSPI requests and replies and replace references to the Exchange server NetBios name, IP address with equivalent information for the proxy

Technical Concepts

- **NSPI Bindings replacement**
  - During profile setup, Outlook uses the **NspiGetProps (0x9)** operation:
    - Ask for a set of properties
    - One of the property required is PR_EMS_AB_NETWORK_ADDRESS
      - Returns a list of binding strings:
        - ncacn_ip_tcp:exch2k3.openchange.local
        - ncacn_np:exchange2k3.openchange.local
  - **mapiproxy replaces these binding strings with its own binding strings**

Technical Concepts

- **NSPI Bindings replacement**



Technical Concepts

- **NSPI Bindings replacement**
  - We still have references to the original Exchange server we need to remove for full-transparent proxying:
    - **NspiQueryRows (0x3)**: request for PR_EMS_AB_HOME_MDB
      - /o=Org/ou=OrgUnit/cn=Configuration/cn=Servers/cn=**EXCHANGE**/cn=Microsoft Private MDB
    - Needs to be replaced with mapiproxy netbios name:
      - /o=Org/ou=OrgUnit/cn=Configuration/cn=Servers/cn=**MAPIPROXY**/cn=Microsoft Private MDB
  - Difference with NspiGetProps, the data we return to Outlook is used in further part of the communication
  - **We need to maintain the replacement filter until NSPI connection ends**

Technical Concepts

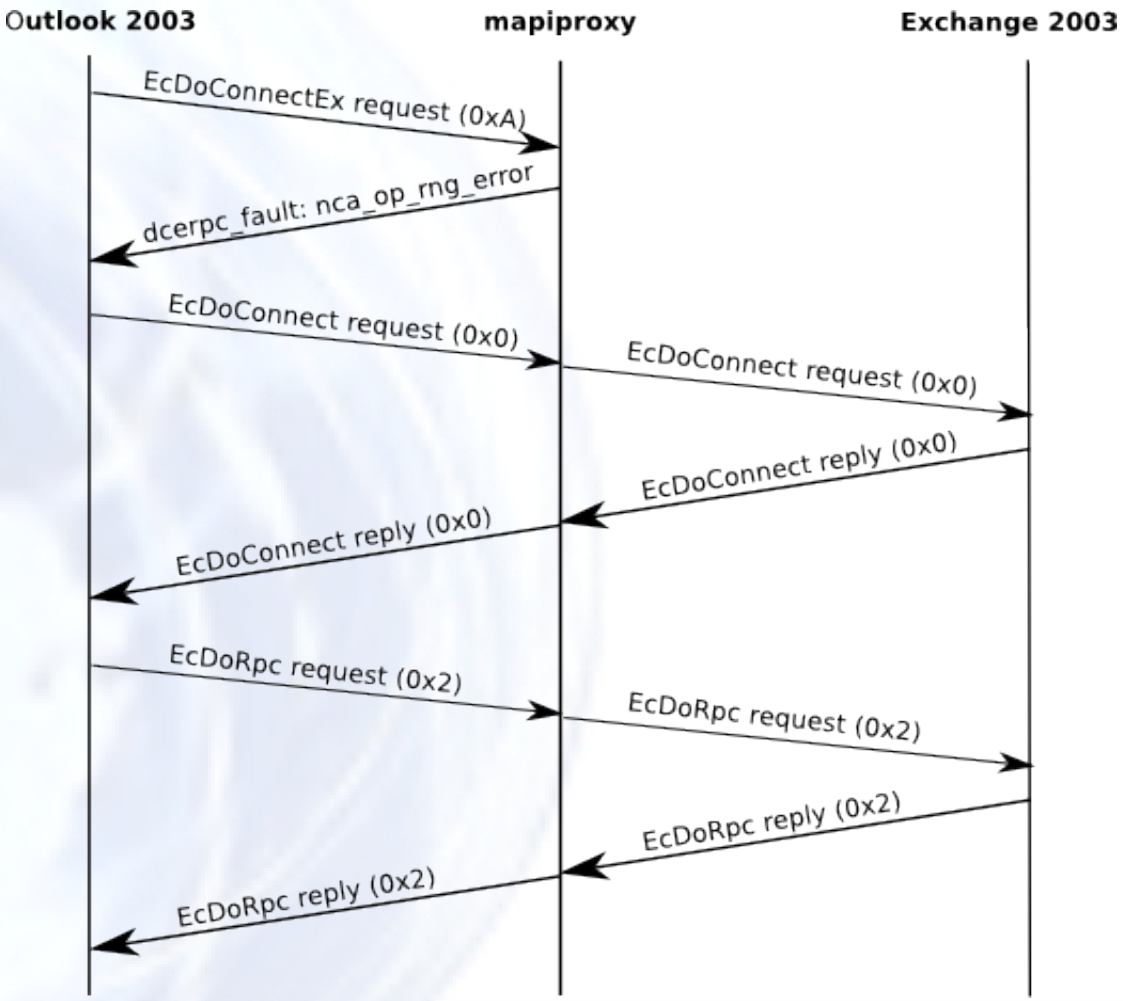
- **Force EMSMDB protocol version**
  - When Outlook 2003 and above opens, it connects to the EMSMDB pipe using EcDoConnectEx (0xA).
  - If successful, Outlook will send MAPI data within EcDoRpcExt2 (0xB):
    - LZ77 + Direct2 encoding
  - If not available, Outlook will use the old EcDoConnect:
    - It means the server may be Exchange 2000
    - Only support xor'ed content (xor 0xa5)
    - Use EcDoRpc (0x2) for Exchange transport

Technical Concepts

- **Force EMSMDB protocol version**
  - To force Outlook not to use EcDoRpcExt2, mapiproxy returns:
    - **dcerpc_fault: nca_op_rng_error** to EcDoConnectEx calls
  - When Outlook connects to the EMSMDB pipe using EcDoConnect (0x0), **mapiproxy needs to change the store version** (3 unsigned short integers):
    - Outlook relies on the store version returned by Exchange to adapt its behavior
    - Store version should match an Exchange 2000 version
    - If set to an Exchange 2003 version, Outlook will keep trying to connect using EcDoConnectEx causing an infinite loop.

Technical Concepts

- **Force EMSMDB protocol version**



3 Stackable Modules System

Stackable Modules System

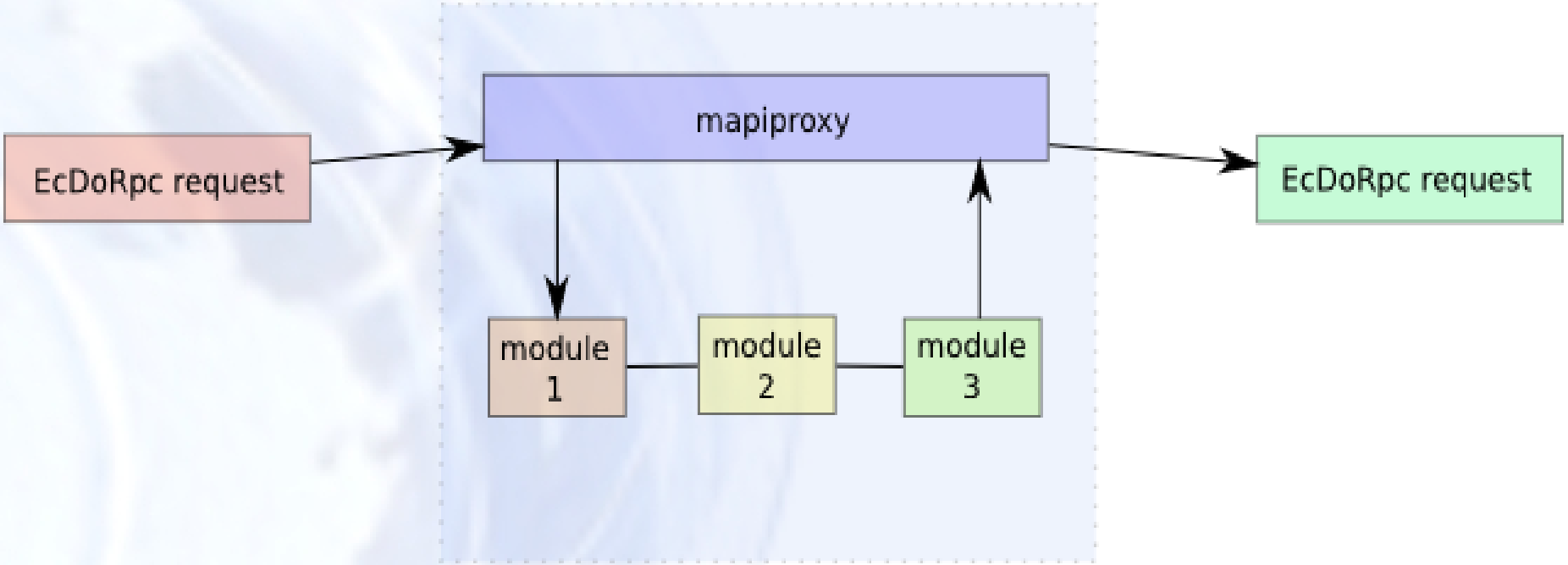
General Overview

- Development framework to add new features
- Let developers focus on ExchangeRPC traffic rather than transport
- **Stackable:**
  - Modules are added to a list
  - Each of these modules can have a specific scope
  - Modifications from one module **transparently relayed** to the next one
  - **They have a limited set of hooks**

- **Mapiproxy modules are DSO** (dynamic shared object):
  - Install in a specific location (dcerpc_mapiproxy folder)
  - Enabled or not in smb.conf:
    - dcerpc_mapiproxy:modules = downgrade,dummy
    - Modules are sequentially processed
    - **Module ordering matters**

Stackable Modules System

- **General Overview**



Stackable Modules System

Module Entry Point

- Modules MUST have an entry point called **samba_init_module**
- Within this function:
  - we set **general information** about the module
  - Specify module's hooks
  - Register our module using **mapiproxy_module_register** function
- Required general information are:
  - **module.name**:
    - the name we will be using in the dcerpc_mapiproxy:modules line in smb.conf
  - **module.description**:
    - a brief description line for information purpose only
  - **module.endpoint**:
    - the interface this module operates on
    - This can be a specific interface or "**any**"

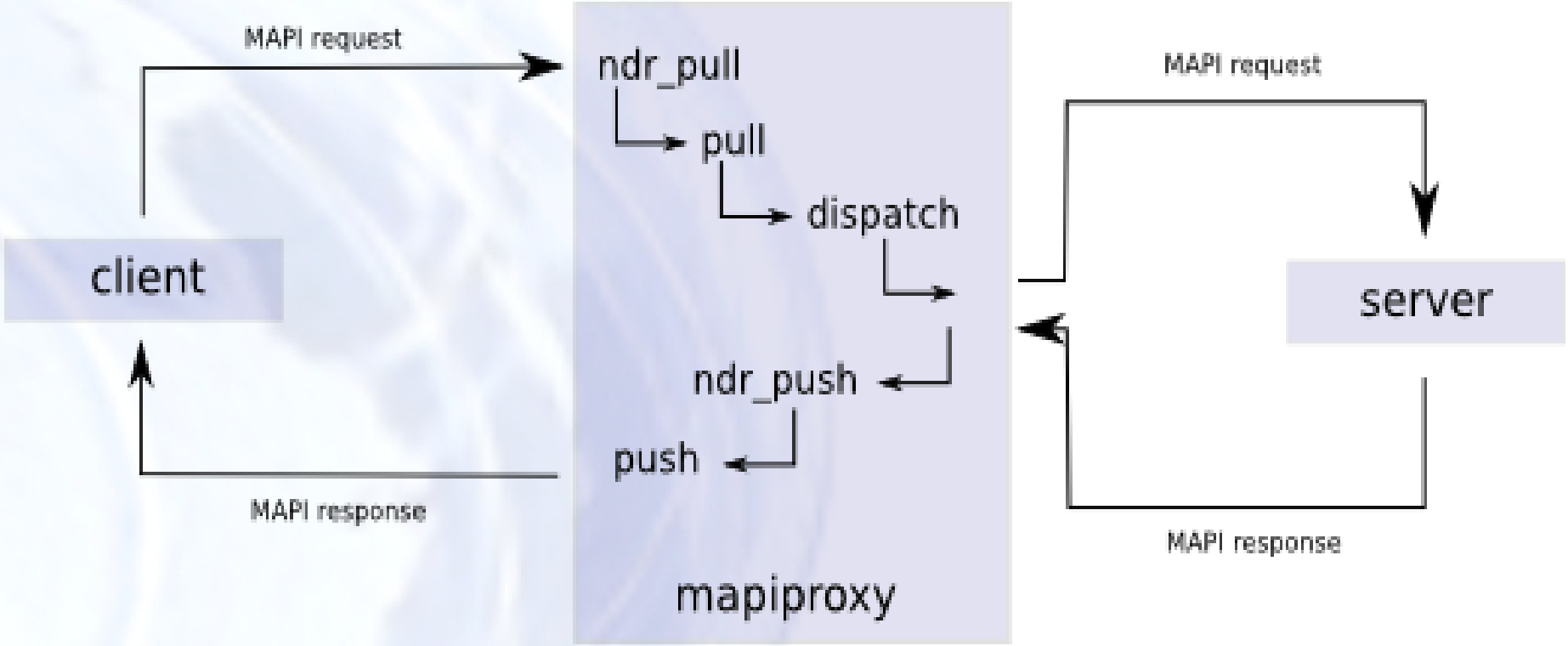
Stackable Modules System

▪ Module Entry Point (Sample example)

```
NTSTATUS samba_init_module(void)
{
	struct mapiproxy_module module;
	NTSTATUS		ret;

	/* Fill in our name */
	module.name        = "sample";
	module.description = "A sample module";
	module.endpoint    = "any";

	/* Fill in all the operations */
	module.init     = sample_init;
	module.push     = sample_push;
	module.ndr_pull = sample_ndr_pull;
	module.pull     = sample_pull;
	module.dispatch = NULL;
	module.unbind   = NULL;

	/* Register ourselves with the MAPIPROXY subsytem */
	ret = mapiproxy_module_register(&module);
	if (!NT_STATUS_IS_OK(ret)) {
		DEBUG(0, ("Failed to register 'sample' mapiproxy module!\n"));
		return ret;
	}

	return ret;
}
```

Stackable Modules System

- **Module Hooks**



Stackable Modules System

- **Module Hooks**
  - **init:**
    - Initialization routine **only called once**
    - Generally used to get smb.conf parametric options for the module
    - Initialize global module structures
  - **ndr_pull:**
    - called before data from a **request** is extracted from the NDR blob.
  - **pull:**
    - function called when mapiproxy receives a **MAPI request**.
    - The request has already been extracted and its information filled into MAPI structures
  - **dispatch**
    - Similar to the mapiproxy top-level dispatch function, it is used to dispatch the information.
    - This function is called after the pull but before the push.
    - Moreover it is **called before the request is forwarded to the remote endpoint**.

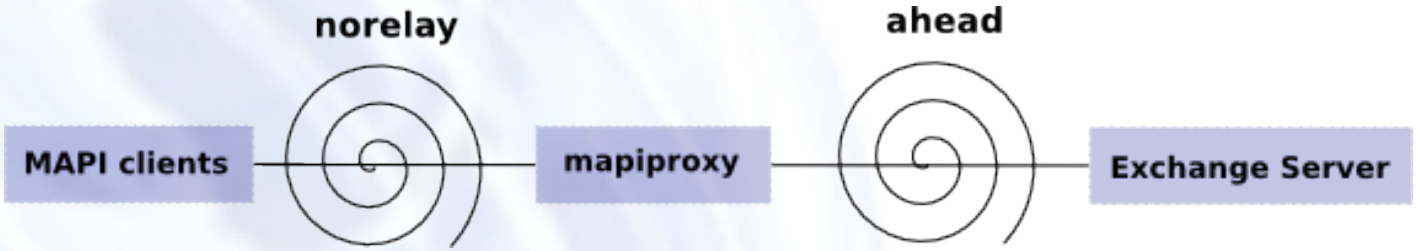
Stackable Modules System

- **Module Hooks**
  - **ndr_push:**
    - This is the function called before data from a **response** is extracted from the NDR blob.
  - **push:**
    - called when mapiproxy receives a **MAPI response**.
    - The response has already been extracted and its information filled into MAPI structures
  - **unbind:**
    - function called when the connection closes.
    - It can be used to free data associated to a given session and stored within a module global list.

Stackable Modules System

mapiproxy structure

- Sometimes a module may want to **bypass the module stack**
- Want to **impact the overall mapiproxy** behavior
  - Modules can alter the default behavior in their dispatch routine



- **norelay:**
  - Boolean variable
  - Do not to relay the incoming request to the remote server
  - directly jump to the push (response) mapiproxy code

- **ahead:**
  - Boolean variable
  - Do not to relay the incoming response to the client through the push and dcerpc_ndr_request routine
  - loop over the dispatch routine

Stackable Modules System

- **mapisession API**
  - Server creates the session context using dcerpc_handle_new()
  - Mapiproxy relays the traffic and **relies on handle returned by Exchange and used by Outlook**
  - If a module is performing complex operations with a **lifetime > 1 call,** you need to **save the server policy_handle for further calls**
  - Mapisession API makes this tracking easier to handle:
    - create a context
    - can store private data
    - can set a destructor
    - compare current session with saved one
    - release the context

MAPIProxy Security Project

MAPIProxy Security project

- **Objectives**
  - **Write one or more Security modules for mapiproxy**
  - 3 different categories:
    - **Protector**
    - **Monitoring**
    - **Freedom**
  - Among other things it is extremely important you **pay attention to potential license incompatibilities** preventing your code from being redistributed
  - The level of difficulty of your module/proposal is a **marking factor**
  - **<mapiproxy-secu@openchange.org>**
    - **Mail in English only**
  - **http://www.openchange.org (Developers > Mapiproxy Project)**

MAPIProxy Security project

- **Protector Category**
  - Any module related to e-mail security
    - Antivirus
    - Anti-spam
    - Anti-phishing
  - You can either choose to focus on a particular module or **embrace a more generic solution**:
    - Extract MAPI data (stream or properties)
    - Use LMTP/Unix sockets to pass data to relevant services
  - Looking for an antivirus to use directly from your module?
    - Have a look at StormAV and contact the LSE

MAPIProxy Security project

- **Monitor Category**
  - Any module related to Outlook-Exchange data monitoring
    - Nagios
    - Intelligent data agent
  - The above list is just an overview of possible modules and proposals are welcome
  - Some of these modules will require an important development effort
  - **This factor will be taken into account for grading**

MAPIProxy Security project

- **Freedom Category**

- Any module which doesn't fit in previous categories

- If you have an idea of module related to security and which development can cope with agenda detailed later, have a chat with us

MAPIProxy Security project

- **Agenda**
  - **Step 1: Registration and Validation**
    - Form a group
    - Upload a proposal on Epitech Intranet
    - if non-available send your proposal to mapiproxy-secu@openchange.org and prefix your subject with **[mapiproxy-proposal]**
    - **Proposals will be published on OpenChange website**
  - **Step 2: Development and Mid-Term Survey**
    - Evaluate current state of development
    - Diagnose problems
    - Dedicated support
  - **Step 3: Final Defense: January 19th 2009 Week**
  - **Step 4: Award Ceremony**
    - committee will evaluate each projects
    - Nominate the best

MAPIProxy Security project

- **Daily support / Questions:**
  - OpenChange team is available on IRC:
    - Server: **irc.freenode.net**
    - Channel: **#openchange**
  - If you have any question related to MAPI understanding, mapiproxy module development, **just come and ask** (in English)
  - If nobody answers, **idle and wait** – you'll always have an answer to your question
  - We will create a technical FAQ on openchange.org with common questions
  - Mid-survey: physical meeting
  - IRC: chat about the project
  - Email: non-urgent questions

Bibliography

- MAPIProxy documentation, OpenChange Project, 2008
http://mapiproxy.openchange.org

- MAPIProxy Security Project, OpenChange Project, 2008
http://www.openchange.org/index.php?option=com_content&task=view&id=122&Itemid=90

- Proxy Server article, Wikipedia
http://en.wikipedia.org/wiki/Proxy_server

Thanks

- Thanks to Brad Hards for reviewing the slides