Security & Risk Conference
November 3th - 6th 2010
Lucerne, Switzerland

The Truth about ABAP® Security

Ertunga Arsal

Agenda

- Introduction to SAP* Applications
- Information about ABAP programs
- Common Programming Mistakes [Demo]
- The Threat Agent
- How to stay secure

*SAP refers to SAP R/3 and Netweaver applications throughout this presentation, not the company.

About Me

- Ertunga Arsal
  - Security Researcher with focus on Enterprise Systems
  - Founder of ESNC GmbH, a company specialized in SAP Security

SAP Systems

- Business specific
  - HR, Finances, Logistics...
- Hold the Crown Jewels
  - Hence “Business”
- Are usually extensively customized
  - SAP consultants on-site
  - Long running implementation projects
- Less exposure to typical hackers
  - Who would learn ABAP for hacking?
  - How would someone try it at home?

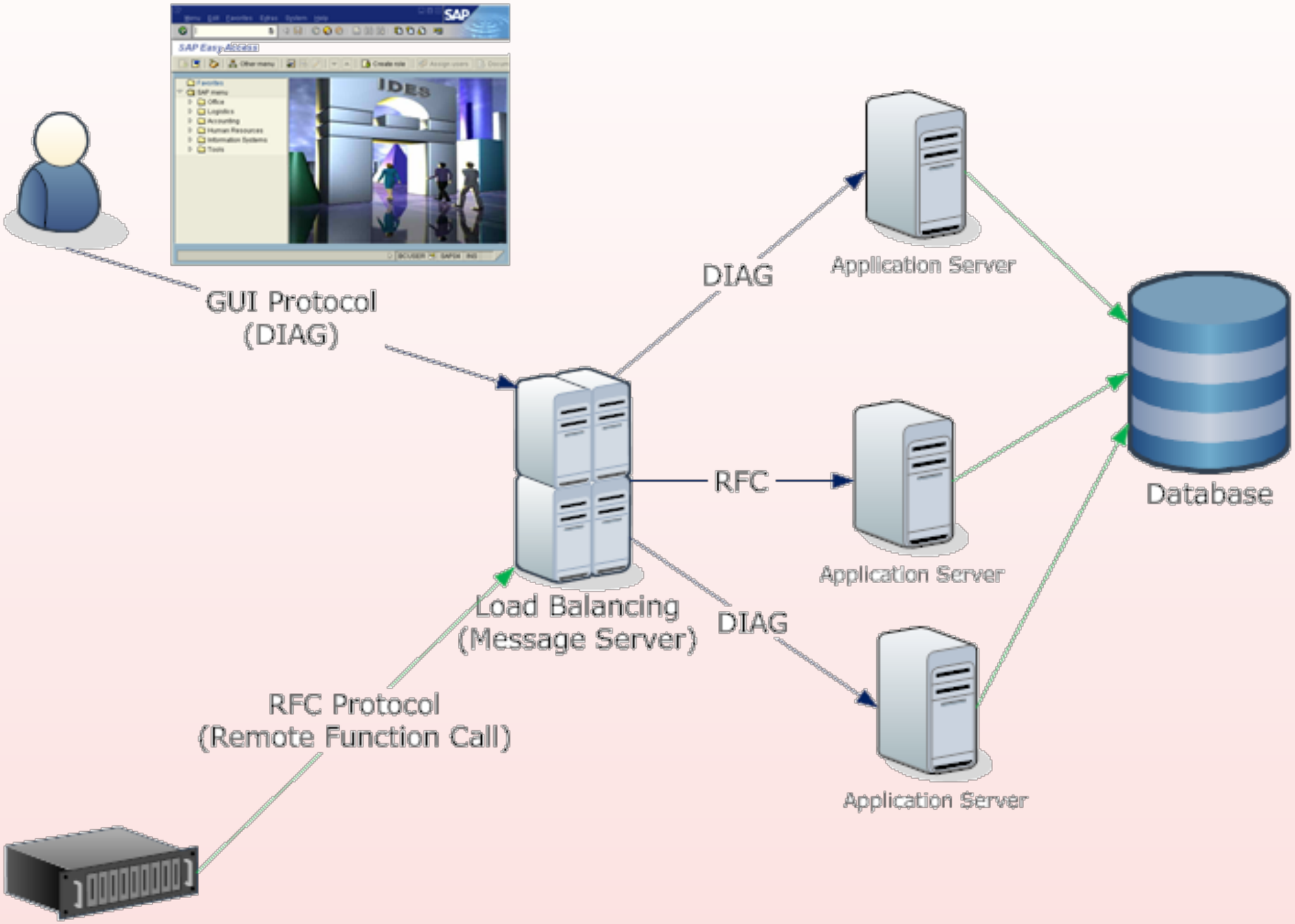
SAP Systems

- Less publicly available vulnerability related information
  - Too few independent security companies in this area
  - Security focuses on authorizations and segregation of duties
- Less detection tools specific to those applications
- Risks are underestimated/general IT Security efforts are typically unbalanced at companies
  - How many Global 500s are running SAP for the core business?
  - How many people from their IT Security teams have SAP security skills?

SAP Applications (ABAP)

- ABAP code holds almost all of the business logic
- More than 2.000.000 programs are present at an SAP ECC 6.0 system after installation.
  - Some programs have more than 50.000 lines of source code
- ABAP Language is very powerful and easy to learn
  - High level and easy to read applications
  - Low level functionality is proxied to the kernel executables when required. e.g for encryption.
    + ABAP stack can “call” the kernel.
    + We’ll only focus on the native ABAP code for this presentation.

Simplified Connection Overview



ABAP CODE

- Common types are:
  - Reports (Programs)
    + With or without GUI
  - Function Modules (Programs that can be executed as a function)
    + Normally you can’t execute them without writing code but there are a few screens (transactions) such as SE37 which let you run them
      # Never ever give execution rights of e.g SE37 to a person in a productive system without a big fight. Be extremely cautious. (The punch may even come from a harmless “display” user)
  - Remote Function Modules (Programs that can be called from a remote system)
    + You don’t need a GUI or browser to run them.
- Others:
  - Classes, BSPs, Webdynpros etc. etc.

“GUI users are the most powerful users” Myth

- Vulnerabilities in self developed remote function modules can expose the SAP system to complete internal (and probably external) network.
- Running remote function modules:
  - Use Java, C, other languages with RFC-SDK or simply execute the test program startrfc.
  - Example: Following creates a new user with god rights:

```
startrfc -3 -h 10.1.5.4 -s 05 -c 010 -u ERTUNGA -p HASHDAYS   -F SUSR_RFC_USER_INTERFACE -E USER=SATRIANI -E ACTIVITY=01 -E
PASSWORD=RUBINA -E USER_TYPE=A -T USER_PROFILES,12,r=-<press ENTER>
SAP_ALL<press enter>
<press ctrl-z and enter>
```

  - There is no exploit involved. Everything is intended functionality.
    + Solid example to beat the argument “RFC users are not a threat because they cannot login via SAPGUI”
    + Time to recheck your company’s shared folders whether there are any scripts with hardcoded passwords and time to eliminate them.

- RFC (a.k.a communication) users are thus very very important!
  - secure their passwords
  - make them part of password change process
  - Don’t forget: GUI (a.k.a dialog) users which have S_RFC rights can also execute remote function modules
  - SAP_ALL FOR COMMUNICATION USERS IS A NO GO!

A few RFC's to note down and protect: (Proper user authorizations is the key)

- RFC_READ_TABLE
  - Reads the contents of any table
    + Especially problematic where sensitive data e.g salary data is processed
  - Has bugs in converting e.g binary fields
    + 1 Byte = 2 Hex, so 20 byte hash -> 40 hex chars
    + Only returns first 20 chars -> only first half of the password hashes
- SUSR_RFC_USER_INTERFACE
  - can be used for creating/modifying users.
- RFC_ABAP_INSTALL_AND_RUN
  - Takes abap source lines and executes them
    + does not execute on production systems
      # non-production does not mean that system is unimportant!
  - Widely known!!! tighten user authorizations to prevent abuse
  - More restricted in latest NetWeaver Systems
    + SAP_ALL RFC users don't have those restrictions!!!

Self Developed ABAP CODE

- Like any other code:
  - Might have defects
  - Those defects can result in security vulnerabilities
- “Code injection” is possible via similar flow:
  - A nice to have feature that saves the developer in tough situations
  - Becomes a weapon at the hands of an attacker
- We’ll focus on a few deadly ones

Dynamic ABAP

- Statement: “GENERATE SUBROUTINE POOL”
  - Dynamically generates ABAP code.
  - If the code is generated via user specified input, mistakes mean:
    + ABAP Injection
    + Game over
  - An example is the TMS_CI_START_SERVICE vulnerability

TMS_CI_START_SERVICE

- Transport Management System required this
- It is a remotely executable function module
- Takes an input table as source code and if the parameters are specified properly, executes the contents of it.

TMS_CI_START_SERVICE

- Here is a simple representation of the vulnerable part of it:

```
Generate subroutine pool pp_table name ix_context.

perform (ix_command) in program (ix_context) tables pp_table.
```

- SAP Patched it via 2009 March updates
  - SAP Note 1298160: Forbidden program execution possible
- TMSADM default password is at least for the last 5 years public
  - Password is “PASSWORD”
  - No need to tell you to change it!

SQL Injection

- ABAP typically uses parametrized queries.
  - Developers can still specify parts of sql statements dynamically by parentheses

- Not dynamic: `SELECT ColumnA FROM TableA INTO[...]`

- Dynamic: `SELECT(var_ColumName)FROM (var_TableName)INTO[...]WHERE(var_WhereClause)`

- Avoid dynamic statements where possible!

SQL Injection

- It’s not a bug, its a feature in concept “Run Time Type Creation”
  - (e.g Z_RTTC report in NSP Test system)
  - https://wiki.sdn.sap.com/wiki/display/Snippets/Concept+of+Run+Time+Type+Creation
- Means generic table access - if not done properly
- !!! Also check the “EXEC SQL”

Cross Site Scripting

- Hard to believe we are still talking about it in 2010
- Proper sanitization/encoding of the input data is the key for self developed web code such as BSPs.
- If not done, an attacker can do everything related to XSS, plus steal e.g the SSO2 (Authentication) cookies from the clients
  - SSO2 cookies are stateless so client impersonation is a breeze.
    + Avoid using this mechanism without proper controls
  - If you have F5's or similar devices, encrypt cookies based on origin ip
    + can kill business if you encrypt based on full ip (32 bits)
    + can be too open if you just encrypt /24 of that ip
      # What happens to NAT clients?

INSERT REPORT

- Statement: `INSERT REPORT`
- Writes custom code to any ABAP program
- It’s even possible to call an editor to make it more user friendly
  - Called editor is similar to the ABAP development environment
- Very suspicious if found in self developed code

Executable Program RS_REPAIR_SOURCE

- Unpatched version does not have authorization checking.
- People with e.g SE38 rights can execute this and manipulate the system and data of it.
- Same as ABAP injection, only more convenient.
- SAP Released Patch on March 2009
  - SAP Note 1167258: Program RS_REPAIR_SOURCE
- There are other critical ABAP statements but they are beyond our scope for today. [one hour time limit hit]

The Threat Agent: ABAP Developer

- Writes code that runs at the heart of the system
- The user rights and permissions don’t apply to him where he develops
  - He can assign god rights to itself
  - Audit logs are typically disabled on development systems
    + If enabled, most probably developers will be able to disable/tamper them
      # remember to log always to an external system.
- You need to trust the developers more than your security team
  - Would you hire an ABAP developer who recently worked at a competitor?
    + IF answer EQUALS ”HELL, YEAH”, think again now.

How to stay secure? (Development specific basics)

- Proper systems architecture is a prerequisite.
  - Read and Apply the “SECURE CONFIGURATION SAP NETWEAVER - APPLICATION SERVER ABAP” document from SAP
    + Make sure relevant people in your company also read it!
    + Check: https://service.sap.com/~sapidb/011000358700000968282010E.pdf
- Audit the code against security vulnerabilities before transporting to production systems
  - Currently only 2 products known to me. From ESNC GmbH and from VirtualForge GmbH
- Ensure self developed programs are always checking for proper authorizations

How to stay secure? (Development specific basics)

- Don’t let development systems directly upload code to production systems
- Don’t make development related changes on production systems. (some companies directly develop on production systems even with a proper landscape)
  - hard to believe but true
- Syncing passwords to development systems means, possibility of developers to capture valid passwords for production systems. Avoid it!

How to stay secure?

- Get rid of insecure and/or default passwords
- Follow vendor’s security notes and guidelines
- Convince the upper management that staying 2 years behind the security patches is a bad idea!
- Install the latest security patches
- Install the latest security patches
- Install the latest security patches
- Install the latest security patches
- Install the latest security patches

Recommended Reading

- “Secure Coding - ABAP”
  - http://help.sap.com/saphelp_nw2004s/helpdata/en/58/4d767ed850443c891ad27208789f56/frameset.htm
- “Secure ABAP Programming” by Wiegenstein, Schumacher, Schinzel, Weidemann
- Every month, the recent SAP Security Notes
  - (Every day would also do fine)
  - https://service.sap.com/securitynotes

Questions?

Ertunga Arsal

ertunga@sabanciuniv.edu