Rootkits and Trojans on Your SAP Landscape

SAP Security and the Enterprise

Ertunga Arsal

SAP systems are the heart of many enterprises. Most critical business functions run on SAP Applications and the complexity of these systems makes it very difficult to protect against attackers. Default setups, forgotten/unimplemented security configurations, weak password management and change processes that apply to one 'unimportant' system can result in complete compromise of the SAP landscape. The legal consequences, lost/damaged business and reputation can be disastrous depending on the type of the attack. While companies invest a lot to secure SAP systems at business process level for example by designing authorization concepts, implementing separation of duties or by using GRC (Governance Risk and Compliance) tools, the security at technical level mostly lacks attention. In this paper, I present several attack paths exploiting configuration weaknesses at technical level, leading to attack potential to single systems, to whole SAP landscapes, and finally the whole enterprise network. By demonstrating creative exploit variants of configuration weaknesses, I motivate the necessity to safeguard a SAP system at technical level.

1 Introduction

With more than 102,000 customers in over 120 countries and including more than half of the world's 500 top companies, SAP systems can be found in any medium to big size enterprise either supporting the internal organizational processes such as human resource management, business intelligence or finance, or directly supporting core business processes, such as material management, supply chain management, or customer relationship management.

From an attacker's point of view, a SAP system can be a primary target for several reasons, the most important one being SAP systems' value as crown jewels. It can be assumed that SAP systems are usually where the money is (more than 250 fortune 500 companies use SAP), therefore a successful attack may result in bigger financial benefit to the attacker compared to other systems. Typical SAP systems have 1000 to 30.000 users. These systems are normally heavily interconnected to others. Many companies need to open these systems to external companies, even competitors, for business requirements.

Consequently, the attack surface is increases but often enough no appropriate countermeasures are taken as a response. Although companies are aware that these systems are part of the critical IT-infrastructure, the main focus with respect to security is at business functional level mostly focusing to authentication and authorization. As a result, the security of the underlying technical infrastructure is weakly developed and

often, directly allows easily attacking an SAP system, the SAP landscape, and whole company network. Competitive advantage can be lost, sensitive personal data can be disclosed to the public, modifications of business figures, financial data, or sales orders can occur without being noticed for a long time. In the following sections, I will present how easy it is to attack a SAP system without proper protection, by showing some attack building blocks that can be combined to leap between the objects of the SAP landscape.

2 The Basics: Technical Overview

In this section the technical components building a SAP system are shortly described with their functions and interactions. The description here is limited to the traditional ABAP application server scenario, serving as basis for old R/3 systems and the new NetWeaver use scenarios.
The following figure shows the main technical components:

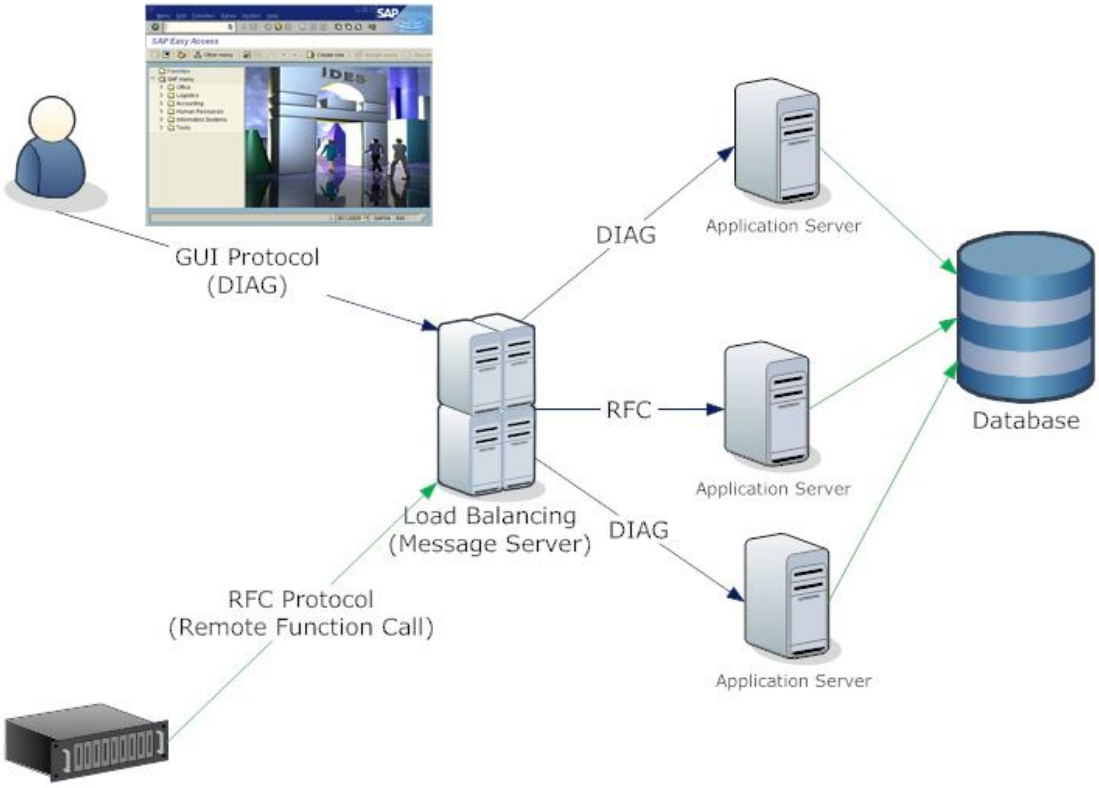

Figure 1: SAP system technical overview (simplified)

- The SAPGUI Client: this component implemented by the SAPGUI program is installed at client machines of SAP users.
- The Message Server [not to be confused with a mail system]: This component is the load balancer, distributing SAPGUI clients to one of the application server of a SAP system. There is only one message server for a single SAP system.
- The SAP Application Server: this component provides the processing capabilities needed to execute the requests of SAP users. There can be many application servers

for an SAP system to provide load distribution. The application server executes ABAP code providing the business and system functions.

- The SAP Gateway [not to be confused with a network router]: This component handles RFC (Remote Function Call) communication requests and manages so-called external RFC server programs.
- The external RFC servers: these components are for example stand alone programs, which provide special or dedicated functions that are not available from the application servers.
- The RFC clients: these components are for example stand alone programs, which access application server functions or external RFC server programs.
- The SAP Database: this component is a database holding all data and code of a SAP system. Oracle and DB2 are the most common database systems running SAP.

The main communication protocol used between the components to exchange data is the RFC (Remote Function Call) protocol, which is used between application servers, between RFC clients and the application server or external RFC servers. There are higher level protocols based on RFC such as DIAG used between SAPGUI Clients and application servers to transport screens and data shown at the SAPGUI application. SAPGUI can only be used by the users that have the "Dialog (GUI)" permission. This permission is determined by selected user type automatically. E.g. a user, which is set as 'communication user' doesn't have this permission.
Users can be assigned to roles and authorizations. Permission profile "SAP_ALL" contains all authorizations within an SAP system (with minor restrictions). This permission is the equivalent of 'root' in UNIX systems.

3 Segregation of Duties

Security at business level is implemented using the segregation of duties principle. For example a person, who requisitions the purchase of goods or services, should not be the person who approves the purchase. SAP has a very granular authorization system that can be adapted to the business needs of the implementing company. Using this authorization system companies are required to enforce proper segregation of duties because of legal requirements, fraud prevention, and various other business related integrity functions. Breaking the segregation of duty property thus has serious business and legal impact. This is because segregation of duty is in the heart of many legal requirements such as SOX, PCI, or GRC in general to achieve necessary security at business level. Looking at the technical configuration of SAP systems it is evident that most of the SAP systems run in companies are vulnerable to attacks immediately breaking the segregation of duty property.

3.1 Attacking the segregation of duties

Attacks to the authentication mechanisms are one of the many ways of attacking the segregation of duties property of a company. This can be accomplished by password cracking, attacks on Single Sign-on mechanisms, by privilege escalation, or by attacks to the message server.

3.1.1 Password Cracking

SAP systems provide means to prevent online password brute forcing by preventing number of attempts with wrong passwords for the same user (password lockout). However, usernames and their salted password hashes are stored in the table USR02, if the regular authentication mechanism is used. It must be noted, that there are several ways of accessing table USR02 providing a variety of potential attack paths to an attacker. To name a few:

- Using transactions SE16, SE16N (and many more)
- Using RFC/RFC-SOAP functions such as RFC_READ_TABLE to retrieve them remotely (many other RFC functions are also available)
- Backups
- Direct database access to database table SAPR3.USR02

One might think that accessing the table would not be possible because of permissions, but on the one hand there are so many access paths and only one weakly protected access path will be sufficient for the attacker. On the other hand there are ways around the assumed protection mechanisms as I will motivate in later sections.
Once getting hold of the table content, attackers can easily use offline password cracking for retrieving the user credentials and access the system with such credentials later on.

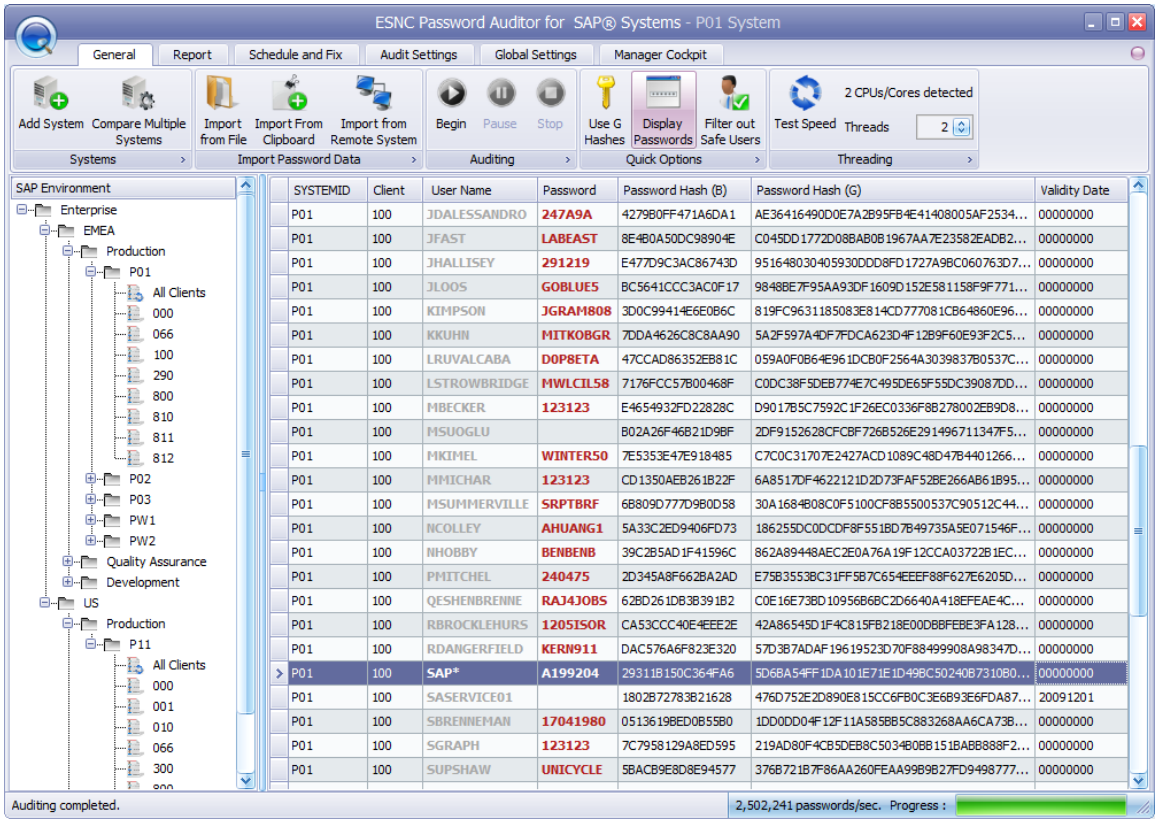

Figure 2: Recovering SAP Passwords

Besides safeguarding potential access paths, the best protection is actually enforcing that all accounts are equipped with strong passwords to drastically increase the time needed

for brute forcing and assuring the strength of these passwords on a regular basis. Note that versions prior to 6.40 only support up to 8 character uppercase passwords.

3.1.2 Attacks to Single Sign-on (SSO)

SAP provides mechanisms for single sign-on between different systems. This mechanism is based on a ticket generated for a user from the sapsys.pse file (i.e. the store of the server's private certificate) of the applications server. With this ticket, it is possible to login as this user, at all SAP systems, where single sign-on trust is established.
The ticket can be used as the authentication provider for RFC, RFC-SOAP and HTTP(s) protocols. Thus getting hold of such tickets or even being able to generate such tickets as needed allows an attacker to logon as a specific or arbitrary user. Often SAP systems are assumed not to be directly accessible from certain network locations thus attacks seem not be possible even with stolen credentials.
SSO tickets can be used with protocols that can be proxied through regular http proxies. Often overlooked is the fact that most internal company http proxies allow RFC-SOAP and HTTP(s) interfaces of SAP systems behind them to be exposed, although direct connection to the SAP system is restricted. Thus if the 'connect' method is allowed for ports other than SSL port (tcp/443), attackers can also connect to the RFC interface of a SAP Application Server for attacks.

3.1.2.1 Stealing the ticket

Traditional XSS attacks, network sniffing, man–in-the-middle attacks can result in theft of the SSO cookies of SAP applications. Please note that although many protocols used with SAP systems allow transmission to be encrypted, this protection is usually not used within companies. As the formerly well-defined company network perimeter is vanishing more and more by using new technologies, equipment (e.g. WIFI, mobile communicators) and roaming users, such attacks assumed already defeated are perfectly valid again. Since the origin of the ticket is not checked during authentication, an attacker can use these tickets for accessing the SAP system from any computer. Tickets are not bound to the protocol; hence stealing the ticket from one protocol gives the attacker the flexibility of choosing the remote protocol for further attacks.

3.1.2.2 Generating a ticket

The private key of a SAP system for generating the tickets for users is stored in the sapsys.pse file. If an attacker retrieves the pse-file, he can issue tickets for any user account. The Application server needs to load the pse-file on startup. Thus, from an operational point of view it is not possible to password protect it. There is an option to specify a password file named 'credits_v2' during pse generation. However, this password mechanism only uses obscurity and can be bypassed with ease.
Please note that besides the file system, the pse-file is also stored in the database. Thus any attacker having access to the database, database backups, or database management functions within SAP can retrieve the contents of the pse file and start generating tickets allowing him to access any SAP system in the SSO environment as any user.
In consequence companies need to ensure that pse-files are protected in all locations and they should have proper key revocation/replacement strategies.

3.1.3 Privilege Escalation

Another way to subvert the segregation of duty property of a SAP system is the escalation of privileges by either assigning more privileges to the current user account or by illegally switching to a high privileged user account. This can be easily achieved by using ABAP reports or functions vulnerable to injection. By this way, an attacker can for example escalate the privileges of the current user account. The easiest way to perform the attack is using ABAP injection vulnerabilities (see below) or if database access is possible, database manipulation of tables storing the user privileges can also result in same effect.

4 Gateway Service and Secinfo Protection

The SAP gateway is the component that provides the RFC interface. Running operating system commands at the gateway machine is a built-in functionality. SAP provides an access configuration file 'secinfo' (and reginfo, on kernel 6.40 and above) to restrict the use of gateway functions. Due to a 'kernel bug' (according to SAP), it is possible to bypass the secinfo file restrictions and still execute operating system commands. SAP fixed this with the kernel patch 1298433.

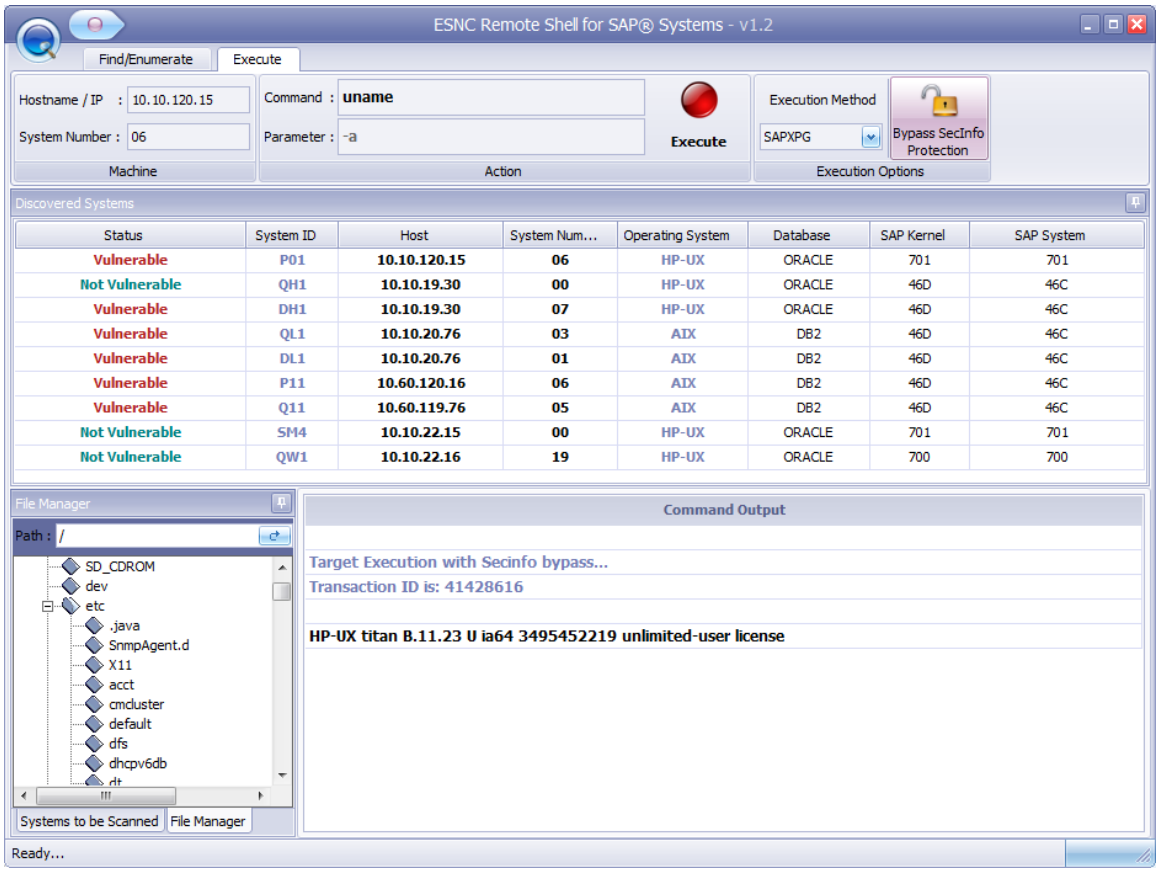

Figure 3: PoC Gateway remote shell with Secinfo bypass

Any company which has not configured the gateway protection or has not installed the kernel patch, might be the victim of this attack. Thus, implementing the latest patches are crucial.

5 Sap ABAP Security

Companies put a lot of effort in customizing their systems to their business requirements. An often overlooked part of code development is the security aspect. Although many companies pay special attention to proper authorization of execution of custom developed code, mostly the code that goes into production only passes a functionality test and nothing more.
In SAP, all custom developed code should pass proper coding standards. Although SAP does not mention anything about ABAP injection, the attack vector is not fictious and any proper injection can be disastrous, as it can bypass almost all security restrictions that are supplied by the ABAP stack afterwards. The following section shortly lists the most dangerous ABAP statements that may result in attack possibilities at ABAP level.

"GENERATE SUBROUTINE POOL" injection

The `generate subroutine pool` construct is used for creating ABAP statements in memory and then executing them. Once created on memory the execution is triggered using the `perform` command.

An example to this vulnerability would be a specific function of the transport management system, which was patched on March 2009 (see sapnote 1298160)

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.
exit.
```

The user supplied table `pp_table` is used to create a dynamic function with another user supplied function name `ix_content`. The form `ix_command` of the function is later executed with `perform` command. The result is appended to the table `pp_table`. Without proper checking any user supplied data can be used for injection.

"INSERT REPORT" and "GENERATE REPORT"

No development code should be allowed for generating reports on the fly. These statements are especially dangerous as they can be used to overwrite existing reports or functions modifying the SAP system's behavior. There are other statements with similar functionality for e.g dynpros. These should also be handled with care.

"CONCATENATE „

If a user supplied input is used in a `concatenate` statement, it can have undesired results, depending on where the concatenated variable is used later on. Critical examples are for example strings that are used to define ABAP reports on-the-fly using "Generate Subroutine pool", "insert report", or "generate report".

SQL injection

Although often neglected, SAP systems are as well vulnerable to SQL injection, especially when the "`where (condition)`" is used in SQL statements. If an attacker can manipulate the string "condition" it is possible to inject and execute SQL statements within ABAP.

"EXEC SQL" Injection

ABAP reports often generate additional reports on-the-fly that make use of the native SQL interface of SAP systems indicated by `EXEC SQL ... ENDSQL` statements encapsulating the native SQL commands. Depending on the way the report and statements are created, it is possible to inject additional SQL statements or even inject ABAP code into the report created on-the-fly. As the generated report is normally executed after creation the injected commands are also executed. This can by used to either execute arbitrary ABAP commands or arbitrary SQL statements natively at the database of the SAP system. An attacker with access to such a possibility can use it as building block for further attacks described for example above.

6 SAP Rootkits

In the existence of the possibilities described shortly above, an attacker can combine these building blocks to install a rootkit to the SAP system for example using any of the following attack paths:

- Direct database access (e.g. because of weak Oracle security)
- Transporting code to SAP system (e.g. exploiting improper transport authorizations, using the ability to write to transport directories, exploiting unpatched reports, which give transport access)
- Using ABAP injection
- By gaining SAP_ALL rights (e.g. by user impersonation, privilege escalation, password cracking)

The most easy attack path is available after gaining SAP_ALL permission, which can be achieved using a variety of attacks as described above. As then full system control is available the attacker would change any SAP system code such as specific RFC functions that are by default accessible without authorization. The malicious code would for example extent the normal function of the RFC module by hidden functionalities such as:

- receive ABAP code and execute it
- receive a table name and return the table content

This would allow performing any operation by just uploading and executing appropriate ABAP code and would allow access to any system or business data stored in tables.
The consequences of such an attack for a company would be disastrous: Any financial statement, any data in the systems, any business decision based on this data could not be trusted as unnoticed manipulation or disclosure could have happen. This should raise strong motivation for all involved persons to take SAP security as serious as possible as they will be held responsible at the end.

7 Triple-Penetration Attacks

By taking control of the SAP system, we shortly have a look on how more elaborated attacks can be launched:

- By modifying the code of the SAP logon program used by the SAPGUI application. Doing that it is easily possible to collect usernames and passwords to an internal table and dump this list when a special username is used for logon.
- By modifying the menu painter displaying the SAPGUI menu page after logon, it is possible to impose attacks on each and every SAPGUI client machine in the company. This can be performed by downloading any code to the client and execute it using standard functions of SAP systems. This opens up a vast amount of attack possibilities to client machines ranging from adding new users to the local machine to installing key loggers.

An attacker can initiate an attack starting from the weakest SAP systems to attack the whole SAP landscape. Often one weakly configured or weakly protected SAP system is sufficient to successful impose attacks on the remaining systems in the landscape using the following staged approach.

Stage 1: Attacking the weakest SAP system
The weakest SAP system in the enterprise is compromised using any of the mentioned methods above. Usually test systems, systems with default passwords, IDES, Minisap instances are the target. The attacker infects this system's main logon functions with a "download and run" type of client payload as described above.

Stage 2: Attacking the users of the weakest system
After modification of the SAP system's main logon function, next time a developer, admin, or any other user connects to this system with SAPGUI, his machine gets infected by the downloaded malicious code.

Stage 3: Attacking all other systems the users connect to
The malicious payload can start a key logger, inject code to the SAP GUI, sniff traffic, or initiate man-in-the middle attacks to retrieve credentials of other systems and modify these systems.

Protection from these multi stage attacks is possible by checking and protecting the source code against tampering and ensuring secure configuration of the whole SAP landscape is in place.

8 Attacking the infrastructure: Man-in-the-Middle - registering malicious application servers

SAP application servers perform tasks in the business process by executing appropriate ABAP code. For load balancing reasons the message server component is used to which SAPGUI clients connect first to be routed to any of the SAP system's application servers. Application servers in turn register with the message server during startup. Unfortunately registering a malicious application server at a message server is possible with a default SAP installation as clients and servers use the same port for communication. This attack path implements a man-in-the-middle attack as the malicious application server will receive a share of the client requests (depending on the load balancing configuration). The malicious application server then receives a client call, can extract any client supplied data (e.g. username, password, and business data), can modify the data to any extent, and finally diverts the call to one of the real application servers. Thus the user will not notice anything of the attack.

Protection from this attack is easily possible as the message server allows to limit message server to application sever communication to a dedicated internal port and also provides a filter list of allowed application servers that may register with the message server using the ms/acl_info file. Also, using secure network communications (SNC), the later stages of the attack can be mitigated depending on the system configuration.

9 Summary

This paper shortly discusses the technical basics of SAP systems, several possibilities to attack them and the protection possibilities. SAP server and client rootkits are critical threats to the companies and they can only be prevented by proper security controls and processes.