CYBSEC®
Security Systems

EXPLOITING SAP INTERNALS

A SECURITY ANALYSIS OF THE RFC INTERFACE IMPLEMENTATION

by Mariano Nuñez Di Croce
[mnunez@cybsec.com]

March 8, 2007

Abstract

SAP security is still a dark world. Very little information can be found on the Net and almost every question related to security assessment of these applications remains unanswered. This paper has the intention of bringing some light into that world, providing the results of a security analysis performed over the SAP RFC interface implementation.

SAP RFC interface is the heart of communications between SAP systems, and between SAP and external software. Almost every system that wants to interact with SAP systems does so using the RFC interface. As stated by SAP: "The RFC library is the most commonly used and installed component of existing SAP software".

This paper describes vulnerabilities discovered in the RFC Library and their security impact. Furthermore, advanced attacks, exploiting default mis-configurations and design flaws in the interface implementation, are presented and explained. Finally, it provides solutions and suggested configurations to protect from described attacks and vulnerabilities..

Table of Contents

1. Introduction

As many legacy and internally developed systems may already be installed (and successfully working) before SAP R/3 is implemented in an organization, SAP must provide a way to communicate with these partners.
With that in mind, SAP has developed quite a lot of *interfaces* to communicate with other systems, such as HTTP, FTP, RFC, XML, ALE and EDI.
Among all these *interfaces*, there is one that demands special attention: *SAP's Remote Function Call (RFC) Interface*.

This *interface* is a key component of the SAP Application Server, providing access to the most largely used protocol for communication between SAP systems and between SAP systems and external (non-SAP) systems.

As the name implies, the purpose of this interface is to allow systems to call function modules on remote SAP servers, and vice versa.

The following section describes the basics of the RFC Interface. Experienced SAP administrators/security professionals can skip this section.

2. Basics of SAP RFC Interface

Originally, SAP implemented IBM's CPI-C interface to communicate with other systems. This protocol enabled the direct transfer of data between systems and fitted well the basic requirements of data communication. However, complex applications demanded more than plain data transfer and the ability to call functions on remote systems was really tempting. Temptation became a need and RFC was born.

Extending CPI-C, SAP's RFC describes a client-server architecture where the server provides services as remotely accessible functions, in a very similar way to Sun's RPC.

We shall now differentiate between two types of RFC connections: Connections between SAP systems and connections between SAP systems and External systems.

2.1. Connections between SAP systems

In SAP Application Servers (SAP AS from now on), RFC services are implemented as *ABAP function modules*. For a *function module* to be accessible by RFC, it must be defined as "Remote-enabled" in its attributes configuration.
These *function modules* receive parameters by means of PARAMETER structures (*importing parameters*) and can also receive (larger) data through TABLE structures. After processing input data, results are sent back as PARAMETER structures (*exporting parameters)* and/or TABLEs. Besides, *exceptions* can be raised, which will propagate to the client.

If an ABAP program in system PR1 wants to call a *function module* in system PR2, a *RFC destination* must be created in PR1. This *destination* is maintained through transaction SM59 and will be included in the function call specification. Specifically, the destination string is used as a index key to the RFCDES table. The associated table record has all the information needed to perform a connection with the target system. Below, an example function call, in an ABAP program, is presented:

```
CALL FUNCTION 'ZCUST_GETMONEY' DESTINATION 'PROD2'
      EXPORTING
            ZCUST_ID = 100
      IMPORTING
            MONEY = cust_money
      CHANGING
            PARCH = parch
      TABLES
            DATA  = cust_data
      EXCEPTIONS
            CUST_NOT_FOUND = 0
            TABLE_EMPTY = 1
```

In case a SAP AS wants to call a RFC function module implemented in an external server, very little changes need to be made to the above ABAP function call: a new *RFC destination* has to be created, specifying the external server connection attributes and modifying the *DESTINATION* field of the *CALL FUNCTION* statement.

2.2. Connections between SAP systems and External systems

To explain this type of connection, it is first necessary to describe what the RFC Library is. The RFC Library is an API released by SAP to allow the creation of external software that interacts with SAP systems by means of the RFC protocol. This library is available for every SAP supported platform and is developed in C.

As stated by SAP, "The RFC Library is the most commonly used and installed component of existing SAP software". The vast majority of systems that connect with SAP R/3 components have this library installed, providing the API to their external client and/or server programs.

Therefore, this type of connection can be composed of an external system (client) calling a function module in a SAP AS (server) or a SAP AS (client) calling a "function module" in an external system (server). In the last case, the function module is in fact a standard programming-language function accessible though the RFC Library.

2.2.1. External RFC Clients

Below, we provide an example of a simple RFC client program. For more complex and detailed examples, refer to [1].

```
int main(int argc, rfc_char_t **argv) {

      RFC_HANDLE handle;
      RFC_ERROR_INFO_EX  rfc_error_info;
      RFC_PARAMETER importing[2], exporting[2], changing[2];
      RFC_TABLE    tables[2];
      RFC_RC rc;
      RFC_INT number=1;
      rfc_char_t   *exception = NULL;

      /* Connect with SAP AS */
      handle  =  RfcOpenEx("ASHOST=127.0.0.1  TYPE=3  SYSNR=00  CLIENT=000  USER=SAP*  PASSWD=pass",
&rfc_error_info);
      if (handle == RFC_HANDLE_NULL) {
            /* Connection error */
            return 1;
      }
```

```
        /* Definition of EXPORT parameters */
        exporting[0].name = "EXP_PARAMETER";
        exporting[0].nlen = strlen(exporting[0].name);
        exporting[0].type = RFCTYPE_INT;
        exporting[0].addr = &number;
        exporting[0].leng = sizeof(number);

        /* Next parameter must be NULL */
        exporting[1].name = NULL;
        /* also unused structures  */
        tables[0].name = NULL;
        importing[0].name = NULL;
        changing[0].name = NULL;


        /* Perform the RFC Call and receive results */
        rc  =  RfcCallReceiveEx(handle,  "HOME_FUNCTION",  exporting,  importing,  changing,  tables,
&exception);

        if (rc == RFC_EXCEPTION || rc == RFC_SYS_EXCEPTION || rc != RFC_OK) {
               /* Process exceptions and errors. Grouped for the sake of space here.*/
        }


        /* Process received parameters and tables */
        …
        /* Close the connection */
        RfcClose(handle);
        return 0;

}
```

2.2.2. External RFC Servers

External RFC servers can work in two different modes: *started* and *registered*.

A *started* external server is a program that is initiated by a *Gateway Server* when a RFC call matching its *destination* is received. If the server resides in a remote host, the Gateway connects with the remote host and starts the program. After the call is processed, the server program is closed. It is worth mentioning that the most common way of connecting to remote hosts are through Remote Shell (rsh) or Remote Exec (rexec), basing authentication on *trusting relationships*.

A *registered* external server takes a different approach: it registers itself at a specific SAP *Gateway*. In this registration process, the external server sends an ID string (*Program ID*) to the *Gateway*, under which the external server will be identified and will become reachable. Therefore, the *destination* in SM59 is configured to point to that *Program ID*: upon a RFC call for that *destination*, the call is forwarded to the external system program registered with that *Program ID*.

Below, we present a simple registered external server program. For more complex and detailed examples, refer to [1].

```
static RFC_RC SAP_API home_function(RFC_HANDLE handle);

int main(int argc, rfc_char_t **argv) {
        RFC_HANDLE handle;
        RFC_RC rc;

        /* Register at SAP Gateway
           argv = "-aProgID -ggw_host -xgw_service" */
        handle = RfcAccept(argv);
```



```
        if (handle == RFC_HANDLE_NULL) {
                /* Couldn't register at SAP Gateway */
                return 1;
        }

        /* Install available functions */
        rc = RfcInstallFunction("HOME_FUNCTION", home_function, "documentation");

        do{
                /* Wait for RFC Call and dispatch to one of installed functions */
                rc = RfcDispatch(handle);
        } while (rc == RFC_OK);

        /* Close the connection */
        RfcClose(handle);
        return 0;

}

static RFC_RC SAP_API home_function(RFC_HANDLE handle) {
        RFC_RC rfc_rc;
        RFC_PARAMETER parameter[2];
        RFC_TABLE tables[1];
        RFC_INT number;

        /* Define reception parameter */
        parameter[0].name = "EXP_PARAMETER";
        parameter[0].nlen = strlen(parameter[0].name);
        parameter[0].type = RFCTYPE_INT;
        parameter[0].addr = &number;
        parameter[0].leng = sizeof(number);

        parameter[1].name = NULL;
        tables[0].name = NULL;

        /* Get data from remote RFC client */
        rfc_rc = RfcGetData(handle, parameter, tables);

        /* Process results */

        return rfc_rc;
}
```

2.3. The Gateway Server

You are probably wondering what the Gateway Server is. Also known as CPIC-C Gateway, the Gateway Server is one of SAP R/3 core services. It allows a SAP AS to interact with remote SAP systems and also with External systems.
Logically, it provides three different services: the Gateway Reader (for TCP/IP communications), the Gateway Work Process (for LU 6.2 communications with IBM mainframes) and the Gateway Monitor (for administration).

Security of the Gateway is provided by different means:

The Gateway Monitor access is regulated through the profile parameter *gw/monitor*. This parameter allows administrators to specify if Monitor access is forbidden (value = 0), enabled only locally (value = 1) or enabled both for local and remote access (value = 2). Note that up to SAP AS with kernel version 6.20, the default value is 2, allowing local and remote administration. Further on, we will see that remote access to this facility enables attackers obtaining highly valuable information for performing advanced attacks.

Security parameters used to regulate the interaction with started and registered external servers are managed through two files: `secinfo` [4] and `reginfo` [5]. The first one allows administrators to apply restrictions on started external programs. The last one specifies parameters for restricting the registration of external registered servers. By default, these files do not exist, which results in no restrictions being applied.

2.4. Authentication and Authorization Mechanisms

In External systems, authentication and authorization tasks are responsibility of external software developers. Access control procedures can be implemented by means of the *RfcInstallExternalLogonHandler()* API function. If not explicitly implemented, there are no authentication/authorization mechanisms and any received RFC call will be processed.

In SAP Application Servers, it is a different story. Authorization is based in checking whether the user calling the function module has a S_RFC authorization object, providing authorization for the function group of the called function module. It is also possible to explicitly perform this authorization check, calling the AUTHORITY_CHECK_RFC function module.

These authentication and authorization procedures depend on the value of the *auth/rfc_authority_check* profile parameter. By default, *auth/rfc_authority_check* is set to 1, which means that authentication and authorization procedures are performed automatically.

2.5. Further Information

SAP RFC can be described along many more pages, detailing its different types, many possible deployment scenarios, etc. If interested, you may find much more information in [6] .

3. Security Analysis of the SAP RFC Interface Implementation

Enough of that boring (but necessary) introduction. In this section we are going to present the results of the research carried out over the RFC Interface Implementation. Tests were done over SAP systems (kernel version 7.00) deployed in Microsoft Windows Server 2003, communicating with external clients and servers developed with the RFCSDK for Windows and Linux, versions 6.40 and 7.00.

3.1. Traffic Analysis

The first (and obvious) thing you realize when starring at a network dump is that RFC communications are clear-text. You can quickly identify logon information, parameter and table names and values, etc:

```
01a0   00 00 00 00 00 00 06 05 14 00 10 5f 22 ea 45 5e        ...........".E^
01b0   22 c5 10 e1 00 00 00 c0 a8 02 8b 05 14 01 30 00        ".............0.
01c0   0a 72 66 63 5f 73 65 72 76 65 72 01 30 01 11 00        .rfc_server.0...
01d0   06 42 43 55 53 45 52 01 11 01 17 00 0b 81 bb 89        .BCUSER.........
01e0   62 fc b5 3e 70 07 6e 79 01 17 01 14 00 03 30 30        b..?w.oy......00
01f0   30 01 14 01 15 00 01 45 01 15 05 01 00 01 01 05        0......E........
0200   01 05 02 00 00 05 02 00 0b 00 03 36 34 30 00 0b        ...........640..
0210   01 02 00 0e 5a 43 55 53 54 5f 47 45 54 4d 4f 4e        ....ZCUST_GETMON
0220   45 59 01 02 05 14 00 10 5f 22 ea 45 5e 22 c5 10        EY......".E^"..
0230   e1 00 00 00 c0 a8 02 8b 05 14 02 01 00 09 43 4c        ..............CL
0240   49 45 4e 54 5f 49 44 02 01 02 03 00 08 43 55 53        IENT_ID......CUS
0250   54 30 30 31 00 02 03 ff ff 00 00 ff ff 00 00 01        T001............
```

To prevent credential and information sniffing, SAP has developed SNC (Secure Network Communications). By default, SNC is not enabled. As you are probably imagining, most organizations run with this configuration.

To log into a SAP system, the classic credentials required are client, username and password. You can clearly

identify the first two pieces in the above dump, while there seems to be no clue for the password. The reason is that the password is obfuscated.

Analyzing different traffic dumps we discovered that the obfuscation algorithm turned out to be a simple XOR operation with a fixed key:

```
for each CHAR in CLEAR_TEXT_PASS
     OBFUSCATED_PASS[i] = CHAR XOR KEY[i]
```

where KEY = [0x96, 0xde, 0x51, 0x1e, 0x74, 0xe, 0x9, 0x9, 0x4, 0x1b, 0xd9, 0x46, 0x3c, 0x35, 0x4d, 0x8e, 0x55, 0xc5, 0xe5, 0xd4, 0xb, 0xa0, 0xdd, 0xd6, 0xf5, 0x21, 0x32, 0xf, 0xe2, 0xcd, 0x68, 0x4f, 0x1a, 0x50, 0x8f, 0x75, 0x54, 0x86, 0x3a, 0xbb]

With this information, the possibility of obtaining valid credentials is just limited to the chance of intercepting an RFC communication (without SNC).

An additional point that is worth to be mentioned, is that developer traces (files automatically created for debugging of connections) access should be secured, as they can have full traffic dumps from which valid credentials can be obtained.

3.2. Authentication and Authorization Mechanisms and the SRFC Function Group

As previously described, authentication and authorization of RFC calls are based on the value of the *auth/rfc_authority_check* profile parameter.

There is a special function group, named SRFC, which contains system function modules. Unless *auth/rfc_authority_check* is set to 9 (remember default value is 1), authentication and authorization procedures are not performed for calls to function modules in this function group. Therefore, it is possible to call these function modules anonymously, which will provide the following information:

- System Information.
- System availability.
- System local servers.
- System local destinations.

If the RFC call is performed by a program using SAP's RFC library, the program will first call the RFCPING function module to check logon information. As this function module resides in the SYST function group (and automatic authentication and authorization take place for any function group different from SRFC), authentication will fail unless valid logon data is provided.

It is possible to bypass this initial call (and therefore the login validation), by opening the connection to the SAP AS specifying the *LCHECK=0* parameter in the connection string.

3.3. Vulnerabilities in the RFC Library

As described in the previous section, if you want to develop an external RFC server, you would use the RFC Library to enable your server to communicate with RFC client partners.

As commented in [6], there are some RFC functions which are installed by default in every external RFC server.

We have detected that many of these default functions can be abused to perform security sensitive operations over external RFC servers, with impact ranging from information disclosure to remote code execution. Following, we describe the analyzed functions and the security caveats detected:

3.3.1. RFC_PING

This function can be used to analyze availability of RFC interfaces, both in SAP Application Servers and external systems.

3.3.2. RFC_GET_DOCU

Calling this function, an attacker can obtain information about installed (accessible) RFC functions in an external RFC server.

3.3.3. RFC_SYSTEM_INFO

This function, present in both SAP AS and external servers, returns quite a lot of information about the server system.

3.3.4. RFC_TRUSTED_SYSTEM_SECURITY *

Developed for internal use by SAP only, this function can be abused to verify the existence of Windows user/ group accounts in an external server system, its domains and trusted domains.

3.3.5. RFC_SET_REG_SERVER_PROPERTY *

This function enables the definition of properties of external registered servers. Calling this function with the appropriate parameters, allows an external client to obtain exclusive use of the server. This clearly represents a denial of service vulnerability.

3.3.6. RFC_START_GUI *

To allow starting SAPGUI on Front-end systems, this function is also present in every external RFC server by default. Analysis of this function spotted a buffer overflow vulnerability which, if properly exploited, would result in the ability to execute remote arbitrary commands over the external server system.

3.3.7. SYSTEM_CREATE_INSTANCE *

This function enables the creation of remote objects, where an object adapter is available.
A buffer overflow vulnerability in the processing of received parameters was also detected in this function, with the same consequences as the above case.

3.3.8. RFC_START_PROGRAM *

This function enables the “controlled” remote execution of programs on external servers.
Analysis of this function revealed information disclosure and buffer overflow vulnerabilities, which would allow to execute remote arbitrary commands on the server

To protect from abuse, SAP delivered the *RfcAllowStartProgram()* function within the RFC Library. This function works as an ACL to regulate RFC_START_PROGRAM access:

No *RfcAllowStartProgram()* = Remote execution disabled
RfcAllowStartProgram("cmd1.exe") = Execution of "cmd1.exe" is authorized.
RfcAllowStartProgram(NULL) = All commands are authorized.

Further analysis of the *RfcAllowStartProgram()* function revealed that the process of validating the received command against allowed ones can be abused to execute other programs in the server system:

The function only verifies that the first N bytes of the requested command matches the first N bytes of the allowed command, where N is the length of the *allowed* command. This allows you to send a request with the following format:

> "allowedCommand.exe\..\..\..\path\to\evil\command.exe"

Of course, to perform this attack successfully, knowledge of an allowed command is needed. Again, this can be obtained through passive sniffing provided that SNC is not present.

** Vulnerabilities discovered by the author and reported to SAP, who, in turn, released appropriate patches. [9]*

3.4. Attacking SAP External Servers

The RFC SDK is shipped with many examples. One of them is the `rfcexec` program, which was originally delivered for testing purposes, but is being commonly used in productive systems. *rfcexec* works as an external registered server and installs the following RFC functions:

- RFC_RAISE_ERROR
- RFC_MAIL
- RFC_REMOTE_PIPE
- RFC_REMOTE_FILE
- RFC_REMOTE_EXEC

Function names speak for themselves. This external server enables the execution of operating system commands, reading and writing files and sending mails. Clearly, it represents a huge security risk.

Due to the fact that this service is deployed in many SAP installations, SAP started to release it with an external logon handler. Therefore, upon a RFC call for any of its functions, *rfcexec* first analyze the contents of the `rfcexec.sec` file. This file allows administrators to define restrictions based on different characteristics of the RFC client and the RFC request. Detailed information can be obtained at [8].

We have discovered that some of these validations are also flawed. To make the situation worse, the default configuration of this file is to allow everything.

4. Advanced Attacks

In this section we are going to describe different attacks we have developed, which abuse default mis-configurations and design flaws. To carry out these attacks, an attacker would need some information about the current deployment of SAP systems. Some of the possible ways of obtaining this information are passive sniffing (if SNC is not used) or having remote access to the Gateway Monitor.

4.1. Evil Twin

As we have previously described, an external RFC server working in *Registration* mode, registers itself at the Gateway specifying a *Program ID*. To allow the deployment of "multithreaded" external servers, it is possible to

register a server program at the Gateway several times, using the same *Program ID*. The problem is that *any* external server can register at the Gateway with an already used *Program ID*.

You are know probably wondering how the Gateway reacts upon a RFC call to a *Program ID* belonging to two (or more) different servers. Our tests indicate that it implements a circular queue algorithm, dispatching every new connection to the next server of the queue. In the case that the selected server is busy processing another client's request, the call is forwarded to the next available server.

Therefore, this attack is quite simple (dishonoring the section title):

1. Attacker opens connection to server with Program ID = EXT1, blocking the connections of other clients.
2. Attacker registers an external server with Program ID = EXT1.
3. Eventually, the client tries to connect with the (original) external server, specifying EXT1 as the destination Program ID.
4. As the server who registered first (original server) is busy with another connection, the call is forwarded to attacker's controlled server.

This attack can be used to obtain valid credentials for login to an external server, analyze function calls contents and to perform denial of service attacks. Note that these attack vectors are possible, even though the original external server is in a different (and possibly restricted) network segment.

4.2. A Wiser (and Stealth) Evil Twin

One problem with the above described attack, is that the normal flow of communication between the original client and server is interrupted. This situation may be undesirable and easily detected. Therefore, a wiser attack has to be made. The following idea came up:

1. Attacker opens connection with Program ID = EXT1, blocking the connections from other clients.
2. Attacker registers an external server with Program ID = EXT1.
3. Eventually, the client tries to connect with the (original) external server, specifying EXT1 as the destination Program ID.
4. As the server who registered first (the original) is busy with another connection, the call is forwarded to attacker's controlled server.
5. At this point the attacker is in control of client parameters and tables, being able to log or modify their contents.
6. Attacker uses established connection with (original) external server, forwarding the (possibly modified) RFC call.
7. Attack receives (original) external server processing results.
8. Send results back to original client.
9. Disconnects from (original) external server.
10. Back to step 1.

The described steps are subject to change depending on the technique used to keep the original server unavailable to legimate clients. If attacking a single-threaded external server, the described method will work just fine. Otherwise, defining the exclusive use of the original server would be necessary, which will allow to avoid connecting/disconnecting from it in every loop of the attack.

Because of its methodology, this attack can be cataloged as a MITM attack over RFC.

4.3. Attacking the SAP Application Server with a Registered Server

This last attack changes the aim from external servers to the core of the system: the SAP Application Server. We will now explain how, the simple fact of being able to register an external RFC server may enable an

attacker to take complete control of a SAP R/3 system.

There was one situation that was overlooked in the introduction to the SAP RFC Interface: apart from normal operation of client/server communication, it can happen that, after receiving a function call, a server needs to request more information from the client to complete the process. In this case, the server performs a *callback.*

A *callback* works just as any other function call (using the same RFC Library functions), with a very slight difference: it uses the already established connection with its partner. The roles are temporally interchanged and a function call is sent to the client (now working as server). Both servers and clients are able to perform *callbacks*.

If the client is a SAP Application Server, the *callback* hits the same context in the SAP system. In other words, the *callback* RFC call executes under the privileges of the user who initiated the first call, bypassing any authentication method. If the user has SAP_ALL authorizations (or any other privileged roles), you can take complete control of the SAP Application Server.

The following steps describe the attack in detail:

1. Attacker opens connection with Program ID = EXT1, blocking the connections from other clients.
2. Attacker registers an external server with Program ID = EXT1.
3. Eventually, the client tries to connect with the (original) external server, specifying EXT1 as the destination Program ID.
4. As the server who registered first (the original) is busy with another connection, the call is forwarded to attacker's controlled server.
5. Attacker performs a callback over the established connection. Depending on initial user authorizations, privileged actions can be performed over the SAP AS.

5. Protection

The described vulnerabilities in the RFC Library have been reported to SAP and patches are already available. Protection from the detailed attacks is also possible, mainly by restricting remote access to Gateway Monitor and effectively controlling the interaction with external servers through the *reginfo* and *secinfo* files. Finally, it is important to prevent credential and information sniffing. This can be done with proper network segmentation and activation of SNC.

6. Conclusions

Where installed, SAP R/3 probably represents one of the most critical systems deployments of an organization, integrating and processing all of its business-related information. This paper exposes different vulnerabilities and attacks that can be implemented against the SAP RFC interface implementation, which is the default and most commonly used interface for communication between SAP systems and also with external systems. Protection from these attacks is possible and must be implemented. We are still researching on this subject and have already obtained some new interesting results, which will probably be included in an upcoming publication.

7. References

1. RFC Library Guide.https://websmp206.sap-ag.de/~form/sapnet?_FRAME=CONTAINER&_OBJECT=011000358700001892452002E
2. “SAP Interface Programming”. SAP Press.
3. RFC Security Guide. http://help.sap.com/saphelp_nw2004s/helpdata/en/37/1a9b6a338cca448508f3a48d2d1e2d/frameset.htm
4. Configuring secinfo. http://help.sap.com/saphelp_erp2005vp/helpdata/en/5a/03a069d3811d188a70000e83539c3/content.htm
5. Configuring reginfo. http://help.sap.com/saphelp_erp2005vp/helpdata/en/1a/4723f86bfb40129db2f7ec3cb1f4f1/content.htm
6. RFC Basics. http://help.sap.com/saphelp_nw04/helpdata/en/22/0424fe488911d189490000e829fbbd/content.htm
7. Default functions.http://help.sap.com/saphelp_erp2005vp/helpdata/en/22/04302e488911d189490000e829fbbd/frameset.htm
8. Security of rfcexec: The rfcexec.sec file. SAP Note 618516.
9. RFC Library Vulnerabilities. SAP notes 1003908, 1003910, 1004084 and 1005397.

8. Special Thanks

I want to thank the people of CybsecLabs and CYBSEC Team-A, mainly to Victor Montero, Andres Riancho, Juan Pablo Perez Etchegoyen and Ezequiel Mina, for their support and collaboration during this investigation.

About CYBSEC S.A. Security Systems

Since 1996 CYBSEC S.A. is exclusively devoted to provide professional services specialized in Computer Security. More than 150 clients around the globe validate our quality and professionalism.

CYBSEC S.A. does not represent, neither sell, nor is associated with other software and/or hardware provider companies, maintaining our services absolutely vendor-independent.

Our services are strictly focused on Information Security, protecting our clients from emerging security threats, maintaining their IT deployments available, safe, and reliable.

Beyond professional services, CYBSEC is continuously researching new defense and attack techniques and contributing with the security community with high quality information exchange.

For more information, please visit www.cybsec.com.