INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME

A Scaleable Monitoring Platform for the Internet
Contract No. IST-2001-32404

D3.3 "Risk and Security Analysis"

Abstract: All monitoring systems may possibly violate the privacy of the monitored entities and may theoritically reveal important infromation to unauthorised users. Therefore, it becomes necessary to make every effort to minimise the possibility of unauthorised access to information and misuse of the monitoring platform in general. This document provides the results of the continuous risk analysis that is performed within SCAMPI to identify the security risks of the system together with appropriate recommendations for risk management in order to reduce them to the minimum poisible.

We used the following mechanisms to analyse the possible privacy risks of the system:

- Using attack tools and independent programmers, we attempted to penetrate the security of the system.

- A careful architecture and code review process took place to ensure that the platform conforms to standard secure programming practices.

Contractual Date of Delivery	March 31th 2004
Actual Date of Delivery	April 16th 2004
Deliverable Security Class	Public

The SCAMPI Consortium consists of:

TERENA	Coordinator	The Netherlands
IMEC	Principal Contractor	Belgium
FORTH	Principal Contractor	Greece
LIACS	Principal Contractor	The Netherlands
NETikos	Principal Contractor	Italy
Uninett	Principal Contractor	Norway
CESNET	Principal Contractor	Czech Republic
FORTHnet	Principal Contractor	Greece
Masaryk University	Principal Contractor	Czech Republic
Siemens	Principal Contractor	Germany

Contents

List of Figures

Chapter 1

Introduction

Security within the SCAMPI project is critical for its success. With traffic monitoring being already a very sensitive issue on its own, decision makers would be sceptical about deploying a monitoring platform such as SCAMPI given a high or unknown security risk. A security incident with a production system would also hurt the credibility of the platform. Decision makers would be even more reluctant to deploy a system with a bad security record.

Computer privacy is becoming a major concern in society. Computer users are becoming more sensitive about privacy issues and policy makers are starting to create relevant regulations, such as for example making organizations liable for the security of the private data they maintain in their databases.

When deployed, SCAMPI will have access to very sensitive data and private information of many users. SCAMPI, being a network traffic monitoring system, has access to the raw network traffic before it can be processed, summarized, and anonymized. Furthermore, SCAMPI is targeted for use with high bandwidth networks and therefore is going to be positioned centrally, giving it access to network traffic from very large numbers of users.

In this document we perform a risk and security analysis to identify and have a clear understanding of the risks involved in deploying SCAMPI in a real environment and to scrutinize the system and find security problems before it is deployed.

Security is a continuous concern within SCAMPI. As development continues, new features that have been designed are implemented and their security has to be evaluated. Also, when the SCAMPI framework is used for a specific application, a risk and security analysis of the whole system has to be based on this work augmented with additional information relevant to the particular task. Therefore in the security evaluation of SCAMPI we anticipate future developments and lay out a framework for reevaluating a SCAMPI system.

The risk and security analysis is presented in the following order. First an architectural security analysis identifies security issues based on the design of the system. The implementation security analysis follows, and guided by the first phase verifies whether the implementation correctly follows the design and does not introduce flaws of its own. Finally, an evaluation of the security of a SCAMPI installation including the operating system is presented. We conclude with recommendations regarding security practices and actions to take.

Chapter 2

Architectural Security Analysis

We begin the architectural security analysis of SCAMPI by presenting the security relevant architectural features of the system, namely a system overview (flow of information and deployment diagram), the classes of users, the security assets, and the security requirements. We proceed with identifying ways that different classes of attackers can compromise the protected assets.

2.1 System Overview

The system follows a client-server architecture. It consists of a central server controlling access to monitoring devices, the MAPI daemon. Applications use the MAPI library to connect to the MAPI daemon as clients. The MAPI library provides stubs for the functions of the monitoring application programming interface (MAPI) which communicate with the MAPI daemon that performs the actual operations on the server side.

Client operations can be controlled in detail by the server using the optional admission control module, which can enforce rules expressed in a policy file. The admission control module is also responsible for authentication of connecting applications.

Currently clients communicate with the server using the UNIX domain sockets inter-process communication mechanism. As a result, all applications are limited to run on the same machine, together with the server. In the future the communication mechanism may be replaced with network sockets to enable applications to run on separate machines.

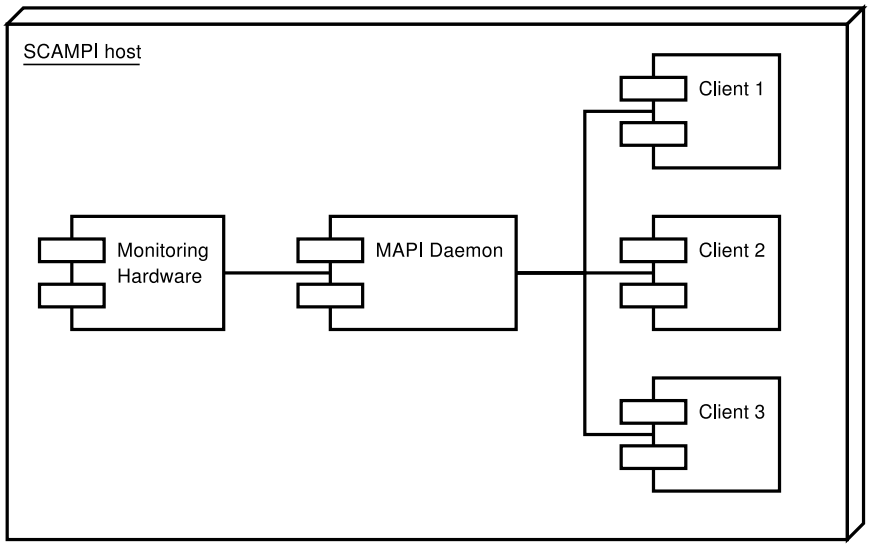

Figure 2.1: SCAMPI deployment diagram.

2.2 Classes of Users

There are two classes of users, users who run applications and communicate directly with the MAPI daemon, and end users, who communicate with MAPI applications. See Figure 2.2 for an illustration of the two classes of users.

Users who run applications are more privileged than end-users, they can do anything an end-user can, while the opposite is not true. For example they can make arbitrary MAPI calls (allowed by the admission control module) while end-users may only feed data to a prewritten MAPI application. They can also send arbitrary (possibly malformed) requests because they have direct access to the MAPI daemon.

Sometimes MAPI applications can be considered trusted and the admission control module disabled. This is often the case when the applications running belong to a single user who also owns the monitor.

2.3 Security Assets

We identified the security assets of a SCAMPI system that require protection. They are, first and most important:

- privacy, then

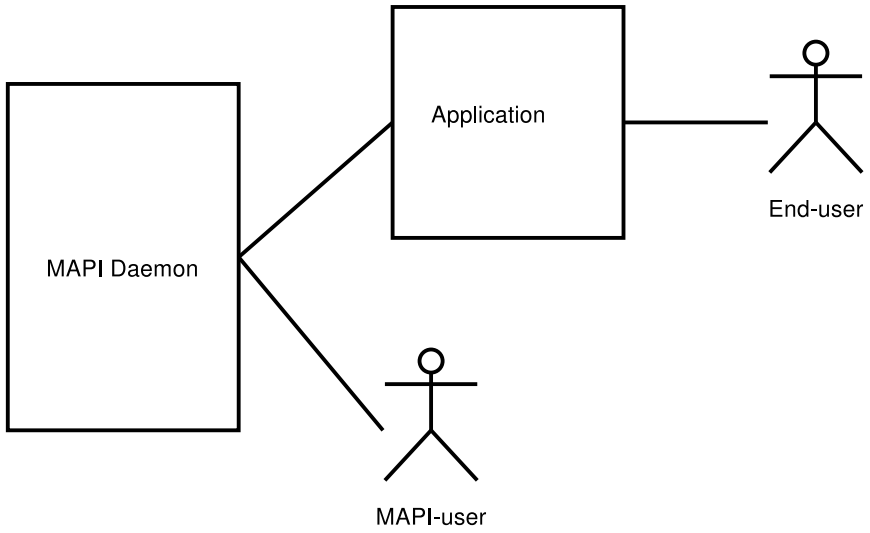

Figure 2.2: Different classes of SCAMPI users.

- authorized use of network resources,
- security of applications,
- uninterrupted operation,
- audit trail.

It should be obvious by the introduction that privacy of the monitored entities is the main security asset of a SCAMPI system.

Another concern is the abuse of the SCAMPI infrastructure for active measurements to launch attacks against third parties or host illegal activities. Such attacks can be categorized as unauthorized use of the network resources of a SCAMPI installation. However, SCAMPI does not provide a significantly larger advantage to an attacker over other possible targets that could be used for that purpose. Therefore, privacy remains the main security concern.

When SCAMPI is used as part of greater system (SCAMPI handling the monitoring), security problems in SCAMPI can be exploited to compromise security assets of the primary application. For example, if SCAMPI is part of a system that has a web interface to network traffic data, there is the asset of the reputation of the organization that owns the web pages. That asset can be compromised by defamation of the web site made possible because of a vulnerability in SCAMPI. This type of exploits may not be necessarily a subset of those that lead to compromise

of privacy, there exist parts of SCAMPI that may involve less risk for privacy, and could have been overlooked because of that, but which involve risks such as those described here.

Another security asset is the uninterrupted operation of the monitor. A possible attacker goal could be the disruption of monitoring or the denial of service to a specific user.

Finally, the audit trail is very important for a SCAMPI system, as is the case with any security conscious system. Logs can help detect and analyze an intrusion as well as provide evidence for legal prosecution. An attacker might want to compromise the privacy without being detected, or might want to erase the traces left by the intrusion so that they cannot lead to the source of the attack or reveal the vulnerability that was exploited.

2.4 Risk Analysis

There are three classes of attackers,

- those that can legitimately run applications,
- those that are allowed to use an application,
- and those that are not legitimate users at all.

From the discussion about the classes of users it is evident that attackers who can write applications have more opportunities to manipulate a SCAMPI system than attackers with end-user privileges. They have access to more interfaces that they may attempt to exploit by providing malicious inputs.

In the following subsections we study attacks against the various security assets by the various classes of attackers. For each attack we try to estimate the probability and effort required on the part of the attacker. We also discuss countermeasures and their effectiveness.

2.4.1 Attacks

2.4.1.1 Operating System Penetration

An attacker may try to gain network access to the SCAMPI host and compromise the operating system so that he can bypass SCAMPI all-together.

This way he could compromise privacy, authorized use of network resources, security of applications, uninterrupted operation, and the audit trail. He could compromise privacy by directly accessing the monitoring devices or by manipulating

SCAMPI through the operating system. Also, with access to the operating system he could use the SCAMPI host for launching attacks against others or host illegal activities. Finally, he could shutdown the system or even destroy the audit trail. The impact of such an attack is enormous.

The countermeasures that can be taken include limiting access to the SCAMPI host using firewalling with appropriate rules that allow access only to legitimate users, hardening the operating system security, and deploying intrusion detection systems that attempt to detect ongoing attacks. With a well protected system this attack should be relatively difficult.

2.4.1.2 Attack against Legitimate User

An attacker may attempt to elevate his privileges by attacking legitimate users. For example, an attacker who has no right to connect to an application running on SCAMPI might steal the identity of a user who has access to the application and connect successfully by pretending to be the legitimate user or even using that user's computer. This way the attacker may launch attacks that were previously impossible to launch.

Similarly to the previous attack the users may be protected using firewalls and hardened operating systems. However, in general it is more difficult to protect a large number of possibly distributed users than it is to protect a centralized server, so the probability of this attack is considerable if there are many users.

2.4.1.3 Attack against MAPI Application

Attackers who have access to applications either as legitimate users or by elevating their privileges as described in the previous attack, may launch attacks against an application.

The code that makes up the application has two parts and each may contain vulnerabilities that compromise the security of the entire application:

- MAPI library code
- application code and other libraries

The attacker may feed data directly only to the application, but the application may in turn feed them to the MAPI library, which in turn

A countermeasure that can be applied is the use of the admission control module to limit the capabilities of applications to the minimum required for their operation. This way, even if an application is subverted, the damage can be limited to the extent of the capabilities granted to the application. By using MAPI and

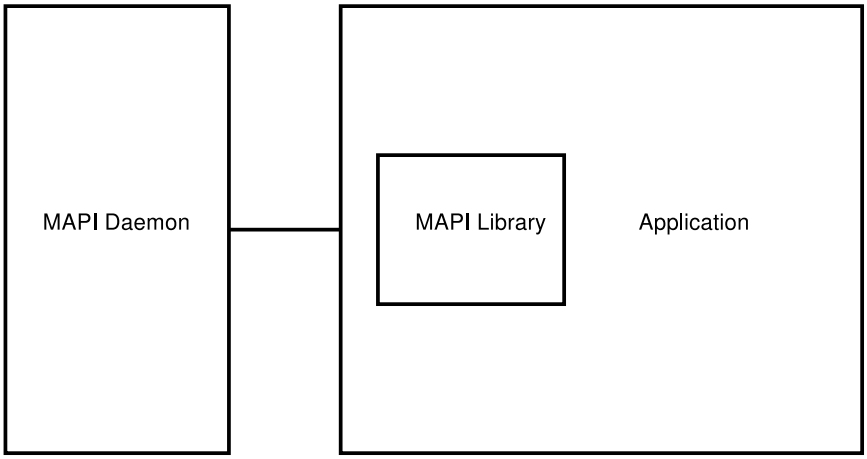

Figure 2.3: Applications contain SCAMPI code.

performing critical operations using MAPI functions the impact of a vulnerability in the application can be minimized.

Additionally, by being aware of the most restrictive admission control policy that can be used with an application, we can quantify the risk that we run by exposing the application to attackers. If no sensitive data is actually transfered to the application, the impact of a compromised application with respect to privacy is limited. If on the other hand sensitive data is transfered to the application, its code has to be checked for security additionally to the SCAMPI code.

2.4.1.4 Attacks against the MAPI Daemon

Attackers with application writing privileges can connect to the MAPI daemon directly. Depending on the use of the admission control module, they may already be able to compromise the assets of privacy and authorized use by invoking the appropriate MAPI operations. If on the other hand they are limited by the admission control, they may try to compromise the MAPI daemon including the admission control module so that they can bypass the restrictions.

This kind of attack may be possible even for attackers who have only indirect access to the MAPI daemon, through an application. The impact is the same but the probability is somewhat smaller.

A compromised MAPI daemon is additionally a threat for security of applications, uninterrupted operation, and even the audit trail. Bad results may be fed to applications, or the daemon may be forced to shut down. As for the audit trail, if the daemon is running with high privileges for accessing the monitoring devices,

Table 2.1: Access required for various attacks.

MAPI Daemon Access	Application Access	Attack
no	no	Operating System Penetration
no	no	Attach Against Legitimate User
no	yes	Attack Against MAPI Application
yes	no	Attack Against MAPI Daemon

by compromising the daemon it may be possible to execute arbitrary code with high privileges.

So there are actually two types of attacks against the MAPI daemon:

- attempts to circumvent the internal checks performed by the MAPI daemon
- attempts to exploit the MAPI daemon in order to execute arbitrary code

Such attacks require a skilled-attacker because SCAMPI is a custom system and an attacker cannot reuse exploitation methods circulating on the Internet for widely used systems. However the source code of SCAMPI is available which makes discovery of such attacks easier.

A countermeasure is the correctness of the code that enforces protection, however this may be difficult to ensure. A source code audit will follow for this purpose in the implementation security section.

Another countermeasure for at least protecting the audit trail is to run the MAPI daemon only with the privileges required to access the monitoring devices, however, the way to achieve this is operating system dependent. A more portable way is to start with privileges but drop them right after opening the monitoring devices and before starting to serve clients.

Chapter 3

Implementation Security Analysis

Even a correct design admits a flawed implementation, and that is why the so-called implementation security is important and implementation security analysis if performed. Implementation security analysis is a difficult concern, unlike the security of a design, the security of an implementation requires more effort to evaluate because the implementation is always more elaborate than a design. Also, with low-level programming languages such as C that lack automatic array bounds checking and allow pointer arithmetic the opportunities for implementation flaws are increased.

Implementation flaws include TOCTOU (time-of-check-time-of-use) [1], format string vulnerabilities [2], and buffer overflows which we will discuss next.

3.1 Buffer Overflows

The most frequent implementation security flaws in C code and the ones with the greatest impact are the so called buffer overflows. The majority of vulnerabilities exploited are buffer overflows [2].

Buffer overflows are possible because of the lack of bounds checking of arrays in C. They can be triggered by malicious input to programs which causes writing past the end of an array. For a thorough explanation of their mechanics see [1], note however that there exist even more sophisticated techniques that can exploit static buffers or even non-executable stacks.

To find buffer overflows a source code audit is required. The person performing the audit tracks the use of input strings originating from an untrusted source and identifies suspicious constructs along that path. A construct can be suspicious when

[1] http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/1996-compsys.pdf
[2] http://teso.scene.at/articles/formatstring/

it involves a standard function which does not perform adequate bounds checking and there are plenty of such functions. All suspicious constructs must be verified against buffer overflow attacks by checking whether it is possible to write past the end of an array by feeding in more data than can be held in it.

3.2 Tools

The search for implementation security bugs can be helped by appropriate tools. The first tool released for this purpose was ITS4 [3] which is proprietary. There exist opensource alternatives such as FlawFinder [4] and RATS [5]. These tools search for dangerous code constructs using pattern matching against a database of patterns and produce reports listing all occurrences of such constructs together with an explanation of why they were flagged.

However, human analysis and verification is still required. These tools are limited first of all because they operate on a lexical level by matching regular expressions and not on a syntactic level by parsing the code. Because they cannot reliably eliminate false positives they tend to err on the safe side. So every reported flaw must be verified by a human for correctness. Furthermore these tools may miss more complicated flaws.

The output of two such scanning tools for the MAPI code-base is provided for reference in the appendices. Appendix A contains the output of FlawFinder and appendix B contains the output of RATS.

3.3 Focus

For the implementation analysis phase we will focus on the MAPI client library and on the MAPI daemon.

Flaws in the MAPI client library could leak sensitive data. The code of the client library runs in the address space of the MAPI applications. A buffer overflow in the MAPI library caused by malicious user input to an application could allow an attacker to manipulate the application within the limits allowed by the MAPI daemon's admission control.

Flaws in the MAPI daemon may allow unconditional access to sensitive data. The MAPI daemon has direct access to network devices. It is responsible for enforcing any rules through the admission control module. In case of compromise of the MAPI daemon, any security checks could be bypassed.

[3] http://www.cigital.com/its4/

[4] http://www.dwheeler.com/flawfinder/

[5] http://www.securesw.com/download_rats.htm

Important code areas include the `mapidcom` function in file `mapid.c`, which is invoked to handle client requests and dispatches control to the various functions that do the actual processing and the implementation of the MAPI interface on the client side (file `mapi.c`).

3.4 Verification

We are going to track all user provided strings within the MAPI library and within the MAPI daemon. Any suspicious code constructs identified by the tools along this path are going to be examined for correctness.

3.4.1 MAPI Client Library

The entry point for suspicious strings to the MAPI library code is through the API functions.

- `int mapi_create_flow(char *dev);`
- `int mapi_create_offline_flow(char *path,int speed);`
- `int mapi_apply_function(int fd, char* funct, ...);`
- `int mapi_connect(int fd);`
- `struct mapipkt *mapi_get_next_pkt(int fd,int fid);`
- `int mapi_read_results(int fd, int fid,void *res);`
- `int mapi_close_flow(int fd);`
- `int mapi_read_error(int* err_no, char* errorstr);`
- `int mapi_set_authdata(int fd,const char *pub, const char *priv,const char *creds);`

The following API functions are safe with respect to malicious input strings because they do not have any string arguments that could be used to provide malicious input.

- `mapi_connect`,
- `mapi_get_next_pkt`,
- `mapi_read_results`,

- `mapi_close_flow`, and
- `mapi_read_error`

The remaining ones are relatively safe, because their arguments are not expected to be provided by users of the applications, however, we are going to analyze them in some detail. We are going to check for correctness suspicious constructs as reported by FlawFinder that are relevant to the code that we are examining.

We first check the `mapi_create_flow` function. It accepts a string argument denoting the device to be opened. The code of the function is located in the `mapi.c` file and has the following logic:

`mapi/mapi.c:153`

```
#define DATA_SIZE 7168
...
unsigned char data[DATA_SIZE];
...

int mapi_create_flow(char *dev)
{
        struct mapiipcbuf qbuf;
        strcpy(qbuf.data, dev);
...
```

The `strcpy` function is one of the suspicious standard functions because it does not check whether there exists enough space in the receiving buffer. It is clear from the code above that if the device name comes from an untrusted source and has not been checked by the application to be less than `DATA_SIZE` then it could overflow the buffer used to hold it. This is a typical buffer overflow vulnerability. The application writer is not aware of the existence of this fixed size internal buffer and the `DATA_SIZE` limit is not documented.

We proceed with the `mapi_create_offline_flow` function which accepts a string argument with the path of the offline trace file to open. The logic is similar to the one presented above and the exact same flaw exists.

`mapi/mapi.c:189`

```
#define DATA_SIZE 7168
...
```

```
unsigned char data[DATA_SIZE];
...

int mapi_create_offline_flow(char *dev)
{
        struct mapiipcbuf qbuf;
        strcpy(qbuf.data, dev);
...
```

The `mapi_apply_function` function accepts a function name. This could have been exploited in the same way as the device name above, however this time instead of `strcpy`, the `strncpy` function is used which accepts an additional argument specifying the length of the destination buffer. This is enough to prevent the attack.

`mapi/mapi.c:272`

```
strncpy(qbuf.function,funct,FUNCT_NAME_LENGTH);
```

The `mapi_set_authdata` function accepts three string arguments which denote various filenames. They are internally passed to the `open` syscall which is safe from such attacks and are not stored in any internal buffers. This function appears to be safe too.

3.4.2 MAPI Daemon

For this analysis we assume that the attacker can provide any input to the MAPI daemon, we cannot rely on any checks performed by the client library.

From inspecting the code we find that the entry point for suspicious strings to the MAPI daemon is the `mapidcom()` function in the `mapid.c` file. Data is received in a `mapiipcbuf` structure which is defined in the `mapiipc.h` header file. The structure has two character array fields that could be used to a perform a buffer overflow attack.

```
char function[FUNCT_NAME_LENGTH];
unsigned char data[DATA_SIZE];
```

The data are fed to a number of functions which stand for commands that the client sends to the MAPI daemon. We will examine these functions.

We first check `cmd_create_flow`, which is the server side implementation of `mapi_create_flow`. It makes use of the `data` field. It uses the field to look up the device and it uses the `strcmp` function to compare it with a series of device names. This use is safe from buffer overflow attacks.

The `cmd_close_flow`, `cmd_read_results`, `cmd_connect`, `cmd_set_authdata`, and `cmd_read_error` functions do not use any of the string fields so are safe from this type of attacks.

We then check `cmd_apply_function`, which uses the function field passed by the application. The function name is passed to the drivers layer. We will check the generic `mapid_apply_function` which is located in the MAPID library, a library of helper functions for drivers. The function name ends up to `get_function` in `mapilibhandler.c` where it is compared against a series of function names using `strcmp`.

We finally examine `cmd_create_offline_flow` which is given the `data` field. The field which stands for a device name is given to `mapidrv_probe` and `mapidrv_create_offline_flow`. We check the ones from the NIC driver in the `mapinicdrv.c` file. `mapidrv_probe` is safe. `mapidrv_create_offline_flow` passes the string to `mapidrv_create_flow` which does not perform unsafe operations with it.

3.5 Summary

We checked the MAPI library and the MAPI daemon for buffer overflow vulnerabilities. We did not find any vulnerabilities in the MAPI daemon, but we discovered two buffer overflows in the following functions of the MAPI library:

- `mapi_create_flow()`
- `mapi_create_offline_flow()`

Chapter 4

Operating System Security

SCAMPI is no more secure than the underlying operating system is. If the operating system is compromised the MAPI daemon can be manipulated at will and the monitoring devices can be accessed directly. We must consider the hosting operating system for a complete security evaluation of SCAMPI.

Furthermore, correct function of the operating system is important for detecting intruders. Intruders with unlimited operating system access can tamper with log files and attempt to hide their presence. Therefore the operating system should protected from a compromised MAPI daemon to protect the audit trail.

The MAPI daemon runs with privileges that allow it to access the monitoring devices, typically super user privileges. If the daemon is compromised unlimited access to the operating system is gained. According to the principle of least privilege, the daemon should run with minimal privileges, only those required to access the monitoring devices. The way to achieve this is operating system dependant. A more portable method that can be applied if the daemon requires privileges only at startup is dropping of privileges.

4.1 Security Scanners

A security scanner is a software which will audit remotely a system for known vulnerabilities. It operates by maintaining a vulnerability database with scripts that allow probing of each vulnerability. They determine the open ports of the scanned system and run all applicable scripts.

4.1.1 Nessus

Nessus [1] is an open-source project that aims to provide to the Internet community a free, powerful, up-to-date and easy to use remote security scanner.

We used Nessus to assess the security of the existing SCAMPI installations at Forth, Cesnet and Uninett. The reports of the Nessus scans for FORTH and Cesnet are provided in Appendix B and C.

The systems at Cesnet and Uninett are protected from unauthorized external connections by firewalls. For the purposes of these scans permission was given to the machine running Nessus to connect to the systems.

A small excerpt from the complete report which can be found in Appendix B follows:

> In this test, Nessus has tested 1 host and found **2 severe security holes**, as w ell as 10 security warnings and 21 notes.These problems can easily be used to break int o your network. You should have a close look at them and correct them as soon as possib le.
> Note that there is a big number of problems for a single network of this size.
> We strongly suggest that you correct them as soon as you can, although we know it is no t always possible.
> On the overall, Nessus has given to the security of this network the mark E because of the number of vulnerabilities found. A script kid should be able to break into your net work rather easily.
> There is room for improvement, and **we strongly suggest that you take the approp riate measures to solve these problems *as soon as possible*** If you were considering hiring some security consultant to determine the security of yo ur network, we strongly suggest you do so, because this should save your network.

The most important alert issued by Nessus was a possibly vulnerable version of OpenSSH. However, it stated that this may be a false positive, since the software may have been patched. Nessus suggested checking the OpenSSH version manually, which revealed a vulnerable version.

[1] http://www.nessus.org

Chapter 5

Recommendations

1. First of all, we recommend fixing the problems identified in the source code audit. Specifically, the buffer overflow vulnerabilities in the following locations:

   - function `mapi_create_flow()` in file `mapi.c`
   - function `mapi_create_offline_flow` in file `mapi.c`

2. The source code checking tools emitted numerous security warnings but we only analyzed the ones related to high risk code areas. The complete output of one such tool, FlawFinder, is present in Appendix A and we recommend that programmers check out the warnings referring to the code they have contributed.

3. When code is contributed to the project, we recommend running source code checking tools. Use the output both for correcting flaws as well as evaluating the quality of the new code based on the number of warnings issued.

4. We further recommend running the MAPI daemon only with the privileges required for accessing the monitoring devices instead of full superuser privileges, or modify the MAPI daemon so that it drops superuser privileges after having established access to the monitoring devices, if such behavior is possible.

5. When using the system with specific applications, we recommend finding out the exact security requirements of applications by finding the most restrictive admission control settings that allow them to function. This can be used to determine and evaluate the risk resulting from compromise of the application. For example consider an application that only counts packets,

then with appropriate admission control settings that do not allow the application to do anything more than just count packets, an attacker could not gain access to packet payload data by compromising the application. This in turn would mean that the application code is not so critical for security and does not represent a high risk.

6. Ensure that the operating system running SCAMPI is properly secured. Harden the SCAMPI servers before deployment. Further ensure that the operating system stays secured by regularly installing security updates to the operating system.

7. Enable logging facilities at the operating system level. Log messages should be preferably forwarded to a different host so that they are preserved in the event of a compromise. Additionally an intrusion detection system could be installed producing logs of its own. Logging is important both after a compromise but also useful for detecting one if they are inspected regularly.

8. Always check with Nessus any deployed SCAMPI system and also check periodically with updated vulnerability databases.

9. Limit network access to the SCAMPI facilities to the absolute minimum. This can be performed by using firewalls and allowing network connections only as needed for correct operation.

Appendix A

FlowFinder Output

```
mapi/adm_ctrl/adm_ctrl.c:705  [4] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf.
mapi/adm_ctrl/authd.c:69  [4] (format) syslog:
  If syslog's format strings can be influenced by an attacker, they can
  be exploited. Use a constant format string for syslog.
mapi/adm_ctrl/db_manage.c:86  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:98  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:142  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:154  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:192  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:196  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:216  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/adm_ctrl/db_manage.c:274  [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
```

```
mapi/adm_ctrl/lib/string_buf.c:37  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/adm_ctrl/lib/string_buf.c:88  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/admctrl_cl.c:110  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/admctrl_cl.c:111  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/admctrl_cl.c:275  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/admctrl_cl.c:276  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/dag.h:116  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely.
  try using a library call that implements the same functionality if
  available.
mapi/dag.h:127  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely.
  try using a library call that implements the same functionality if
  available.
mapi/dag.h:380  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely.
  try using a library call that implements the same functionality if
  available.
mapi/demo/lib_IPC/IPC_IPC.c:11  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/demo/lib_IPC/IPC_IPC.c:32  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/demo/lib_IPC/IPC_IPC.c:79  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/demo/lib_IPC/IPC_IPC.c:97  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/demo/lib_IPC/IPC_IPC.c:120  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/demo/lib_IPC/IPC_IPC.c:134  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapi.c:102  [4] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf.
mapi/mapi.c:106  [4] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf.
mapi/mapi.c:153  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapi.c:164  [4] (buffer) strcpy:
```

```
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapi.c:188  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapi.c:435  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapid.c:213  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapidlib.c:67  [4] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf.
mapi/mapidlib.c:71  [4] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf.
mapi/mapidlib.c:203  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapierror.c:99  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapierror.c:119  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapierror.c:131  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapierror.c:150  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapierror.c:192  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapiipc.c:71  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapiipc.c:80  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapiipc.c:100  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/mapilibhandler.c:81  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/parseconf.c:24  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/parseconf.c:60  [4] (buffer) sscanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different input
  function.
mapi/parseconf.c:63  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/parseconf.c:64  [4] (buffer) strcpy:
```

```
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/parseconf.c:87  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/sids_src/parser.c:182  [4] (format) vfprintf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/sids_src/sids.c:217  [4] (format) vfprintf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/sids_src/sids.c:225  [4] (format) vfprintf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/sids_src/sids.c:245  [4] (format) fprintf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/sids_src/sids.h:35  [4] (format) printf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/sids_src/sids.h:36  [4] (format) printf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/stdlib/bpffilter.c:89  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily misused).
mapi/stdlib/ethereal.c:80  [4] (format) printf:
  If format strings can be influenced by an attacker, they can be
  exploited. Use a constant for the format specification.
mapi/adm_ctrl/authd.c:141  [3] (buffer) getopt_long:
  Some older implementations do not protect against internal buffer
  overflows . Check implementation on installation, or limit the size of all
  string inputs.
mapi/mapid.c:701  [3] (buffer) getopt_long:
  Some older implementations do not protect against internal buffer
  overflows . Check implementation on installation, or limit the size of all
  string inputs.
mapi/sids_src/sids.c:47  [3] (buffer) getopt:
  Some older implementations do not protect against internal buffer
  overflows . Check implementation on installation, or limit the size of all
  string inputs.
mapi/adm_ctrl/adm_ctrl.c:94  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.c:138  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/adm_ctrl/adm_ctrl.c:525  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.c:711  [2] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf. Risk
```

```
  is low because the source has a constant maximum length.
mapi/adm_ctrl/adm_ctrl.h:32  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:57  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:58  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:69  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:71  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:75  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/adm_ctrl.h:93  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/arith_parser.c:115  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/arith_parser.c:185  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/authd.c:305  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/adm_ctrl/db_manage.c:66  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/db_manage.c:122  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/db_manage.c:178  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/lib/darray.c:46  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/adm_ctrl/lib/stack.c:60  [2] (buffer) memcpy:
```

```
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/adm_ctrl/resource_ctrl.c:87  [2] (misc) open:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/adm_ctrl/resource_ctrl.c:91  [2] (misc) open:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/adm_ctrl/resource_ctrl.c:127  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/resource_ctrl.c:267  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/adm_ctrl/resource_ctrl.h:29  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/adm_ctrl/resource_ctrl.h:58  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/admctrl_cl.c:198  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:200  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:202  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:204  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:279  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:321  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:322  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:324  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.c:377  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/admctrl_cl.h:17  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
```

```
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/admctrl_cl.h:18  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/admctrl_cl.h:19  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:177  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:414  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:570  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:571  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:597  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:597  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dag.h:597  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dagapi.h:10  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dagapi.h:17  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dagapi.h:18  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dagapi.h:20  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/dagapi.h:26  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
```

```
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:8  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:19  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:29  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:40  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:55  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:160  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/demo/lib_IPC/IPC_IPC.c:234  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/fhelp.c:60  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/fhelp.c:70  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:86  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mapi.c:373  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:466  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:498  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:500  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:502  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapi.c:513  [2] (misc) open:
```

```
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/mapid.c:431  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapidlib.c:42  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mapidlib.c:51  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mapidlib.c:253  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapierror.c:90  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/mapierror.c:122  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/mapierror.c:142  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/mapiipc.c:19  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mapiipc.c:70  [2] (buffer) sprintf:
  Does not check for buffer overflows. Use snprintf or vsnprintf. Risk
  is low because the source has a constant maximum length.
mapi/mapiipc.c:141  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapiipc.c:146  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapiipc.c:151  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mapiipc.h:46  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mapiipc.h:53  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
```

```
  the maximum possible length.
mapi/mapinicdrv.c:137  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/mstring.c:186  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mstring.c:230  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/mstring.c:284  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/parseconf.c:19  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/parseconf.c:46  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/parseconf.c:49  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/pcapio.h:52  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/scampi_kernel.h:76  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/parser.c:20  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/parser.c:29  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/parser.c:38  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move
  things around to create a race condition, control its ancestors, or change
  its contents?.
mapi/sids_src/parser.c:79  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/sids.c:32  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
```

```
mapi/sids_src/sids.h:23  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/util.c:28  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/sids_src/util.c:30  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/stdlib/cooking.c:214  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:216  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:605  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:613  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:635  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:636  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/cooking.c:775  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/ethereal.c:63  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/stdlib/ethereal.c:75  [2] (buffer) strcat:
  Does not check for buffer overflows when concatenating to destination.
  Consider using strncat or strlcat (warning, strncat is easily misused).
  Risk is low because the source is a constant string.
mapi/stdlib/strsearch.c:83  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.
mapi/stdlib/strsearch.c:179  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/tobuffer.c:144  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/stdlib/tobuffer.c:145  [2] (buffer) memcpy:
  Does not check for buffer overflows when copying to destination. Make
  sure destination can always hold the source data.
mapi/adm_ctrl/adm_ctrl.c:150  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
```

```
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:380  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:384  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:388  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:411  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:543  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:544  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:550  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:551  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:557  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:558  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:568  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:575  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:576  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:582  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:583  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
```

```
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:589  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:590  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:615  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:616  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:622  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:634  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:656  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:657  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:679  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:730  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:735  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:738  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:748  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:753  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
```

```
mapi/adm_ctrl/adm_ctrl.c:756  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:772  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:777  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:780  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:790  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:795  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:798  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:813  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:814  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/adm_ctrl/adm_ctrl.c:895  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:914  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/adm_ctrl.c:929  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/authd.c:258  [1] (access) umask:
  Ensure that umask is given most restrictive possible setting (e.g.,
  066 or 077).
mapi/adm_ctrl/db_manage.c:100  [1] (buffer) getchar:
  Check buffer boundaries if used in a loop.
mapi/adm_ctrl/db_manage.c:156  [1] (buffer) getchar:
  Check buffer boundaries if used in a loop.
mapi/adm_ctrl/db_manage.c:228  [1] (buffer) getchar:
  Check buffer boundaries if used in a loop.
mapi/adm_ctrl/db_manage.c:286  [1] (buffer) getchar:
  Check buffer boundaries if used in a loop.
```

```
mapi/adm_ctrl/keynote.h:54  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:56  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:58  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:60  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:62  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:64  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:66  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:68  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:70  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:72  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/keynote.h:74  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/lib/string_buf.c:20  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/lib/string_buf.c:67  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/resource_ctrl.c:177  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/resource_ctrl.c:185  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/resource_ctrl.c:571  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/resource_ctrl.c:686  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/adm_ctrl/resource_ctrl.c:757  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:133  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
```

```
  crash if unprotected).
mapi/admctrl_cl.c:147  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:148  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:151  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:184  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:185  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/admctrl_cl.c:186  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapi.c:97  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapi.c:161  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapi.c:266  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapi.c:483  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapi.c:484  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapi.c:515  [1] (buffer) read:
  Check buffer boundaries if used in a loop.
mapi/mapid.c:657  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapid.c:657  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapidlib.c:53  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapidlib.c:62  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapidlib.c:202  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapidlib.c:380  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapierror.c:98  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
```

```
  crash if unprotected).
mapi/mapierror.c:118  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:130  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:149  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:173  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:174  [1] (buffer) strncpy:
  Easily used incorrectly; doesn't always \0-terminate or check for
  invalid pointers.
mapi/mapierror.c:174  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:175  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapierror.c:191  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapiipc.c:129  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapiipc.c:151  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapiipc.c:153  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapilibhandler.c:79  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapinicdrv.c:143  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mapinicdrv.c:231  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mstring.c:133  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/mstring.c:134  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/parseconf.c:21  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/parseconf.c:61  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/parseconf.c:62  [1] (buffer) strlen:
```

```
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/parseconf.c:86  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/sids_src/sids.c:170  [1] (port) snprintf:
  On some very old systems, snprintf is incorrectly implemented and
  permits buffer overflows; there are also incompatible standard definitions
  of it. Check it during installation, or use something else.
mapi/stdlib/bpffilter.c:88  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/stdlib/cooking.c:435  [1] (buffer) read:
  Check buffer boundaries if used in a loop.
mapi/stdlib/cooking.c:438  [1] (buffer) read:
  Check buffer boundaries if used in a loop.
mapi/stdlib/cooking.c:441  [1] (buffer) read:
  Check buffer boundaries if used in a loop.
mapi/stdlib/ethereal.c:74  [1] (buffer) strcat:
  Does not check for buffer overflows when concatenating to destination.
  Consider using strncat or strlcat (warning, strncat is easily misused).
  Risk is low because the source is a constant character.
mapi/stdlib/strsearch.c:106  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/stdlib/strsearch.c:194  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/stdlib/strsearch.c:210  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
mapi/stdlib/strsearch.c:215  [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).
Number of hits = 323
Number of Lines Analyzed = 15521 in 1.36 seconds (18130 lines/second)
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
```

Appendix B

Nessus Results for FORTH

Introduction

In this test, Nessus has tested 1 host and found **2 severe security holes**, as well as 10 security warnings and 21 notes.These problems can easily be used to break into your network. You should have a close look at them and correct them as soon as possible.
Note that there is a big number of problems for a single network of this size.
We strongly suggest that you correct them as soon as you can, although we know it is not always possible.
On the overall, Nessus has given to the security of this network the mark E because of the number of vulnerabilities found. A script kid should be able to break into your network rather easily.
There is room for improvement, and **we strongly suggest that you take the appropriate measures to solve these problems *as soon as possible*** If you were considering hiring some security consultant to determine the security of your network, we strongly suggest you do so, because this should save your network.

B.1 HIDDEN.ics.forth.gr

B.1.1 Open ports (TCP and UDP)

`HIDDEN.ics.forth.gr` has the following ports that are open :

- `smtp (25/tcp)`
- `telnet (23/tcp)`
- `ssh (22/tcp)`
- `ftp (21/tcp)`
- `daytime (13/tcp)`
- `discard (9/tcp)`
- `time (37/tcp)`
- `finger (79/tcp)`
- `sunrpc (111/tcp)`
- `auth (113/tcp)`
- `printer (515/tcp)`
- `general/tcp`
- `unknown (32768/tcp)`
- `sunrpc (111/udp)`
- `omad (32768/udp)`
- `ntalk (518/udp)`
- `general/icmp`
- `general/udp`

You should disable the services that you do not use, as they are potential security flaws.

B.1.2 Details of the vulnerabilities

B.1.2.1 Problems regarding : smtp (25/tcp)

Security note :

- `An unknown service is running on this port.`
  `It is usually reserved for SMTP`

- An unknown server is running on this port.
  If you know what it is, please send this banner to the Nessus team:
  00: 32 30 30 34 2d 30 32 2d 32 35 20 31 39 3a 34 30 2004-02-25 19:40
  10: 3a 30 31 20 46 61 69 6c 65 64 20 74 6f 20 6f 70 :01 Failed to op
  20: 65 6e 20 63 6f 6e 66 69 67 75 72 61 74 69 6f 6e en configuration
  30: 20 66 69 6c 65 20 2f 65 74 63 2f 65 78 69 6d 2f   file /etc/exim/
  40: 65 78 69 6d 2e 63 6f 6e 66 0a   exim.conf.

B.1.2.2 Problems regarding : telnet (23/tcp)

Security warnings :

- The Telnet service is running.
  This service is dangerous in the sense that
  it is not ciphered - that is, everyone can sniff
  the data that passes between the telnet client
  and the telnet server. This includes logins
  and passwords.

  Solution:
  If you are running a Unix-type system, OpenSSH can be
  used instead of telnet. For Unix systems, you can comment
  out the 'telnet' line in /etc/inetd.conf. For Unix systems
  which use xinetd, you will need to modify the telnet services
  file in the /etc/xinetd.d folder. After making any changes
  to xinetd or inetd configuration files, you must restart
  the service in order for the changes to take affect.

  In addition, many different router and switch manufacturers
  support SSH as a telnet replacement. You should contact your
  vendor for a solution which uses an encrypted session.

  Risk factor : Low
  CVE : CAN-1999-0619

Security note :

- A telnet server seems to be running on this port
- Remote telnet banner :
  Debian GNU/Linux testing/unstable dag.ics.forth.gr
  dag.ics.forth.gr login:

B.1.2.3 Problems regarding : ssh (22/tcp)

Security holes :

- You are running a version of OpenSSH which is older than 3.7.1

```
Versions older than 3.7.1 are vulnerable to a flaw in the buffer
 management
functions which might allow an attacker to execute arbitrary commands
 on this
host.

An exploit for this issue is rumored to exist.


Note that several distribution patched this hole without changing
the version number of OpenSSH. Since Nessus solely relied on the
banner of the remote SSH server to perform this check, this might
be a false positive.

If you are running a RedHat host, make sure that the command :
          rpm -q openssh-server

Returns :
 openssh-server-3.1p1-13 (RedHat 7.x)
 openssh-server-3.4p1-7  (RedHat 8.0)
 openssh-server-3.5p1-11 (RedHat 9)

Solution : Upgrade to OpenSSH 3.7.1
See also :
 http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375452423794&w=2
    http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375456923804&w=2
Risk factor : High
CVE : CAN-2003-0693, CAN-2003-0695
BID : 8628
```

Security warnings :

- ```
  You are running OpenSSH-portable 3.6.1 or older.

  There is a flaw in this version which may allow an attacker to
  bypass the access controls set by the administrator of this server.

  OpenSSH features a mechanism which can restrict the list of
  hosts a given user can log from by specifying a pattern
  in the user key file (ie: *.mynetwork.com would let a user
  connect only from the local network).

  However there is a flaw in the way OpenSSH does reverse DNS lookups.
  If an attacker configures his DNS server to send a numeric IP address
  when a reverse lookup is performed, he may be able to circumvent
  this mechanism.

  Solution : Upgrade to OpenSSH 3.6.2 when it comes out
  Risk Factor : Low
  CVE : CAN-2003-0386
  BID : 7831
  ```

- ```
  You are running OpenSSH-portable 3.6.1p1 or older.

  If PAM support is enabled, an attacker may use a flaw in this version
  to determine the existence or a given login name by comparing the times
  ```

```
the remote sshd daemon takes to refuse a bad password for a non-existant
login compared to the time it takes to refuse a bad password for a
valid login.

An attacker may use this flaw to set up  a brute force attack against
the remote host.

*** Nessus did not check whether the remote SSH daemon is actually
*** using PAM or not, so this might be a false positive

Solution : Upgrade to OpenSSH-portable 3.6.1p2 or newer
Risk Factor : Low
CVE : CAN-2003-0190
BID : 7482, 7467, 7342
```

Security note :

- An ssh server is running on this port
- Remote SSH version : SSH-2.0-OpenSSH_3.4p1 Debian 1:3.4p1-1.woody.3
- The remote SSH daemon supports the following versions of the SSH protocol :

  . 1.99
  . 2.0

B.1.2.4 Problems regarding : ftp (21/tcp)

Security note :

- An FTP server is running on this port.
  Here is its banner :
  220 dag.ics.forth.gr FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17)
   ready.
- Remote FTP server banner :
  220 dag.ics.forth.gr FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17)
   ready.

B.1.2.5 Problems regarding : daytime (13/tcp)

Security warnings :

- The remote host is running a 'daytime' service. This service
  is designed to give the local time of the day of this host
  to whoever connects to this port.

```
The date format issued by this service may sometimes help an attacker
to guess the operating system type of this host, or to set up
```

```
timed authentication attacks against the remote host.

In addition to that, the UDP version of daytime is running, an attacker
may link it to the echo port of a third party host using spoofing, thus
creating a possible denial of service condition between this host and
a third party.

Solution :

- Under Unix systems, comment out the 'daytime' line in
 /etc/inetd.conf
  and restart the inetd process

- Under Windows systems, set the following registry keys to 0 :
  HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpDaytime
  HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpDaytime

Then launch cmd.exe and type :

   net stop simptcp
   net start simptcp

To restart the service.

Risk factor : Low
CVE : CVE-1999-0103
```

B.1.2.6 Problems regarding : discard (9/tcp)

Security warnings :

- ```
  The remote host is running a 'discard' service. This service
  typically sets up a listening socket and will ignore all the
  data which it receives.

  This service is unused these days, so it is advised that you
  disable it.


  Solution :

  - Under Unix systems, comment out the 'discard' line in
   /etc/inetd.conf
    and restart the inetd process

  - Under Windows systems, set the following registry key to 0 :
    HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpDiscard

  Then launch cmd.exe and type :

     net stop simptcp
     net start simptcp

  To restart the service.
  ```
  ```

```
Risk factor : Low
CVE : CAN-1999-0636
```

B.1.2.7 Problems regarding : time (37/tcp)

Security note :

- A time server seems to be running on this port

B.1.2.8 Problems regarding : finger (79/tcp)

Security warnings :

- The 'finger' service provides useful information to attackers, since it allows them to gain usernames, check if a machine is being used, and so on...

```
Here is the output we obtained for 'root' :

Login: root             Name: root
Directory: /root                    Shell: /bin/bash
On since Tue Feb 24 15:05 (EET) on pts/0 from castro.ics.forth.gr
   1 day idle
     (messages off)
On since Tue Feb 24 15:12 (EET) on pts/1 from castro.ics.forth.gr
   6 hours 52 minutes idle
     (messages off)
On since Tue Feb 24 16:30 (EET) on pts/2 from castro.ics.forth.gr
   1 day 3 hours idle
     (messages off)
On since Tue Feb 24 16:30 (EET) on pts/3 from castro.ics.forth.gr
   1 day 1 hour idle
     (messages off)
No mail.
No Plan.


Solution : comment out the 'finger' line in /etc/inetd.conf
Risk factor : Low
CVE : CVE-1999-0612
```

Security note :

- A finger server seems to be running on this port

B.1.2.9 Problems regarding : sunrpc (111/tcp)

Security note :

- The RPC portmapper is running on this port.

  An attacker may use it to enumerate your list
  of RPC services. We recommend you filter traffic
  going to this port.

  Risk factor : Low
  CVE : CAN-1999-0632, CVE-1999-0189
  BID : 205

- RPC program #100000 version 2 'portmapper' (portmap sunrpc rpcbind) is running on this port

B.1.2.10 Problems regarding : auth (113/tcp)

Security warnings :

- The remote host is running an ident (also known as 'auth') daemon.

  The 'ident' service provides sensitive information to potential
  attackers. It mainly says which accounts are running which services.
  This helps attackers to focus on valuable services (those
  owned by root). If you do not use this service, disable it.

  Solution : Under Unix systems, comment out the 'auth' or 'ident'
  line in /etc/inetd.conf and restart inetd

  Risk factor : Low
  CVE : CAN-1999-0629

Security note :

- An identd server is running on this port

B.1.2.11 Problems regarding : printer (515/tcp)

Security note :

- An unknown server is running on this port.
  If you know what it is, please send this banner to the Nessus team:

```
00: 64 61 67 2e 69 63 73 2e 66 6f 72 74 68 2e 67 72   dag.ics.forth.gr
10: 3a 20 6c 70 64 3a 20 59 6f 75 72 20 68 6f 73 74    : lpd: Your host
20: 20 64 6f 65 73 20 6e 6f 74 20 68 61 76 65 20 6c     does not have l
30: 69 6e 65 20 70 72 69 6e 74 65 72 20 61 63 63 65    ine printer acce
40: 73 73 0a                                           ss.
```

B.1.2.12 Problems regarding : general/tcp

Security warnings :

- The remote host does not discard TCP SYN packets which
  have the FIN flag set.

  Depending on the kind of firewall you are using, an
  attacker may use this flaw to bypass its rules.

  See also :
   http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html
             http://www.kb.cert.org/vuls/id/464113

  Solution : Contact your vendor for a patch
  Risk factor : Medium
  BID : 7487

B.1.2.13 Problems regarding : unknown (32768/tcp)

Security note :

- RPC program #100024 version 1 'status' is running on this port

B.1.2.14 Problems regarding : sunrpc (111/udp)

Security note :

- RPC program #100000 version 2 'portmapper' (portmap sunrpc rpcbind) is
   running on this port

B.1.2.15 Problems regarding : omad (32768/udp)

Security holes :

- The remote statd service may be vulnerable to a format string attack.

  This means that an attacker may execute arbitrary code thanks to a bug
   in
  this daemon.

  *** Nessus reports this vulnerability using only information that was
   gathered.
  *** Use caution when testing without safe checks enabled.

  Solution : upgrade to the latest version of rpc.statd
  Risk factor : High
  CVE : CVE-2000-0666, CAN-2000-0800
  BID : 1480

Security warnings :

- The statd RPC service is running. This service has a long history of security holes, so you should really know what you are doing if you decide to let it run.

  *** No security hole regarding this program have been tested, so
  *** this might be a false positive.

  Solution : We suggest that you disable this service.
  Risk factor : High
  CVE : CVE-1999-0018, CVE-1999-0019, CVE-1999-0493
  BID : 127, 450

Security note :

- RPC program #100024 version 1 'status' is running on this port

B.1.2.16 Problems regarding : ntalk (518/udp)

Security note :

- The remote host is running a 'talkd' daemon.

  talkd is the server that notifies a user that someone else wants to
   initiate
  a conversation with him.

  Malicious hackers may use it to abuse legitimate users by conversing with
  them with a false identity (social engineering). In addition to this, an
  attacker may use this service to execute arbitrary code on your system.

  Solution:
   Disable talkd access from the network by adding the approriate rule
   on your
   firewall. If you do not need talkd, comment out the relevant line in
   /etc/inetd.conf and restart the inetd process.

  See also : http://www.cert.org/advisories/CA-1997-04.html
  Risk factor : Medium
  CVE : CVE-1999-0048

- talkd protocol version: 1
  CVE : CVE-1999-0048

B.1.2.17 Problems regarding : general/icmp

Security warnings :

- The remote host answers to an ICMP timestamp request. This allows an attacker to know the date which is set on your machine.

  This may help him to defeat all your time based authentication protocols.

  Solution : filter out the ICMP timestamp requests (13), and the
   outgoing ICMP
  timestamp replies (14).

  Risk factor : Low
  CVE : CAN-1999-0524

B.1.2.18 Problems regarding : general/udp

Security note :

- For your information, here is the traceroute to 139.91.70.XX :
  139.91.70.YY
  139.91.70.XX

Conclusion

A security scanner, such as Nessus, is not a garantee of the security of your network.
A lot of factors can not be tested by a security scanner : the practices of the users of the network, the home-made services and CGIs, and so on... So, you should not have a false sense of security now that the test are done. We recommand that you monitor actively what happens on your firewall, and that you use some tools such as tripwire to restore your servers more easily in the case of an intrusion.
In addition to that, you must know that new security holes are found each week. That is why we recommand that you visit `http://www.nessus.org/scripts.html`, which is a page that contains the test for all the holes that are published on public mailing lists such as BugTraq (see `http://www.securityfocus.com` for details) and test the security of your network on a (at least) weekly basis with the checks that are on this page.
This report was generated with Nessus, the open-sourced security scanner. See http://www.nessus.org for more information

Appendix C

Nessus Results for CESNET

Introduction

In this test, Nessus has tested 1 host and found **1 severe security holes**, as well as 3 security warnings and 5 notes.These problems can easily be used to break into your network. You should have a close look at them and correct them as soon as possible.
Note that there is a big number of problems for a single network of this size.
We strongly suggest that you correct them as soon as you can, although we know it is not always possible.
On the overall, Nessus has given to the security of this network the mark E because of the number of vulnerabilities found. A script kid should be able to break into your network rather easily.
There is room for improvement, and **we strongly suggest that you take the appropriate measures to solve these problems *as soon as possible*** If you were considering hiring some security consultant to determine the security of your network, we strongly suggest you do so, because this should save your network.

C.1 HIDDEN.cesnet.cz

C.1.1 Open ports (TCP and UDP)

HIDDEN.cesnet.cz has the following ports that are open :

- ssh (22/tcp)
- www (80/tcp)
- unknown (12865/tcp)
- general/udp

You should disable the services that you do not use, as they are potential security flaws.

C.1.2 Details of the vulnerabilities

C.1.2.1 Problems regarding : ssh (22/tcp)

Security holes :

- You are running a version of OpenSSH which is older than 3.7.1

  Versions older than 3.7.1 are vulnerable to a flaw in the buffer
   management
  functions which might allow an attacker to execute arbitrary commands
   on this
  host.

  An exploit for this issue is rumored to exist.

  Note that several distribution patched this hole without changing
  the version number of OpenSSH. Since Nessus solely relied on the
  banner of the remote SSH server to perform this check, this might
  be a false positive.

  If you are running a RedHat host, make sure that the command :
            rpm -q openssh-server

  Returns :
   openssh-server-3.1p1-13 (RedHat 7.x)
   openssh-server-3.4p1-7 (RedHat 8.0)
   openssh-server-3.5p1-11 (RedHat 9)

  Solution : Upgrade to OpenSSH 3.7.1
  See also :
   http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375452423794&w=2
      http://marc.theaimsgroup.com/?l=openbsd-misc&m=106375456923804&w=2
  Risk factor : High
  CVE : CAN-2003-0682, CAN-2003-0693, CAN-2003-0695
  BID : 8628
  Other references : RHSA:RHSA-2003:279-02, SuSE:SUSE-SA:2003:039

Security warnings :

- You are running OpenSSH-portable 3.6.1 or older.

  There is a flaw in this version which may allow an attacker to
  bypass the access controls set by the administrator of this server.

  OpenSSH features a mechanism which can restrict the list of
  hosts a given user can log from by specifying a pattern
  in the user key file (ie: *.mynetwork.com would let a user
  connect only from the local network).

  However there is a flaw in the way OpenSSH does reverse DNS lookups.
  If an attacker configures his DNS server to send a numeric IP address
  when a reverse lookup is performed, he may be able to circumvent
  this mechanism.

  Solution : Upgrade to OpenSSH 3.6.2 when it comes out
  Risk Factor : Low
  CVE : CAN-2003-0386
  BID : 7831

- The remote SSH daemon supports connections made
  using the version 1.33 and/or 1.5 of the SSH protocol.

  These protocols are not completely cryptographically
  safe so they should not be used.

  Solution :
   If you use OpenSSH, set the option 'Protocol' to '2'
   If you use SSH.com's set the option 'Ssh1Compatibility' to 'no'

  Risk factor : Low

- You are running OpenSSH-portable 3.6.1p1 or older.

  If PAM support is enabled, an attacker may use a flaw in this version
  to determine the existence or a given login name by comparing the
   times
  the remote sshd daemon takes to refuse a bad password for a
   non-existent
  login compared to the time it takes to refuse a bad password for a
  valid login.

  An attacker may use this flaw to set up a brute force attack against
  the remote host.

  *** Nessus did not check whether the remote SSH daemon is actually
  *** using PAM or not, so this might be a false positive

  Solution : Upgrade to OpenSSH-portable 3.6.1p2 or newer
  Risk Factor : Low
  CVE : CAN-2003-0190
  BID : 7482, 7467, 7342
  Other references : RHSA:RHSA-2003:222-01

Security note :

- An ssh server is running on this port

- Remote SSH version : SSH-1.99-OpenSSH_3.4p1 Debian 1:3.4p1-1
- The remote SSH daemon supports the following versions of the SSH protocol :

  . 1.33
  . 1.5
  . 1.99
  . 2.0

C.1.2.2 Problems regarding : www (80/tcp)

Security note :

- This port was detected as being open by a port scanner but is now closed. This service might have been crashed by a port scanner or by a plugin

C.1.2.3 Problems regarding : general/udp

Security note :

- For your information, here is the traceroute to 195.113.147.HIDDEN :
  139.91.70.HIDDEN
  139.91.70.HIDDEN
  139.91.34.HIDDEN
  ?

Conclusion

A security scanner, such as Nessus, is not a garantee of the security of your network.
A lot of factors can not be tested by a security scanner : the practices of the users of the network, the home-made services and CGIs, and so on... So, you should not have a false sense of security now that the test are done. We recommand that you monitor actively what happens on your firewall, and that you use some tools such as tripwire to restore your servers more easily in the case of an intrusion.
In addition to that, you must know that new security holes are found each week. That is why we recommand that you visit `http://www.nessus.org/scripts.html`, which is a page that contains the test for all the holes that are published on public mailing lists such as BugTraq (see `http://www.securityfocus.com` for details) and test the security of your network on a (at least) weekly basis with the checks that are on this page.
This report was generated with Nessus, the open-sourced security scanner. See http://www.nessus.org for more information

Bibliography

[1] Aleph1. Smashing the stack for fun and profit. *Phrack*, 7(49):14, November 1996.

[2] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. *DARPA Information Survivability Conference*, January 2000.