Security Laboratory Prototype

Cryptographic Attacks on RSA and SQL Injection

Filipa Silveira e Castro

Instituto Superior Técnico, Faculdade de Direito de Lisboa, Escola Naval,

Lisbon, 2017

Abstract

Today we communicate with everyone through the Internet, but for this we have to be able to trust it. Encryption is an effective way to ensure confidentiality, authenticity, and integrity. However it is necessary that its parameters are correctly chosen to avoid vulnerabilities.

Web applications can be vulnerable to attacks, SQL Injections are an effective way to deceive the machine, forcing it to reveal information.

Training is a factor that really contributes to the correct design and safe use of these systems, either in the theoretical field, in the dissemination of the various concepts, and also in the practical side, through test environments where it is possible to experiment and verify the scope of errors, without real consequences.

In this thesis, a Security laboratory Prototype was developed to demonstrate the cryptographic attacks of Wiener, Common-modulus and Least Significant Bit, and to test SQL Injection attacks. These machines were designed for beginners who take the first steps in the area of Information Security, however, allow the future development of additional modules, for testing other type of vulnerabilities.

1 Introduction

The technological evolution of the last decades has helped the emergence of new forms of communication. This new mode of digital interaction has brought enormous advantages, allowing streamlining the relationship between all types of commerce and services and its clients. However, for all these activities to be developed and adopted it is necessary to trust it, or in other words it is essential that confidentiality, authenticity and integrity are ensured. Encryption is a powerful weapon to achieve these objectives. This ancient science has adapted to modern times and developed strong ciphers with ciphering algorithms that are very difficult to reverse without the knowledge of the key. However, in some particular circumstances, a wrong choice of parameters may weaken the security robustness of cryptographic systems such as RSA.

The security of communication can be compromised in many other ways. Information is a valuable and much desired asset. The search for failures that lead to its disclosure is an area in constant evolution. Today, communication with all kind of entities is often done through web applications that interact with users, receiving and returning information. These sites generally use databases to organize and manage their content. In some cases the dissemination of information is promoted and encouraged, as in the case of marketing. But in other cases, such as for legal or security reasons, its scope should be barred and only accessible to an authorized person. SQL Injections are an effective way to cheat the machine, forcing it to reveal information. This technique uses the dialog boxes (authentication, search ...) of web applications to introduce the attack and induce the machine to reveal, manipulate or even destroy the information.

Throughout this thesis a security lab prototype was developed to demonstrate the cryptographic attacks of Wiener's, Common-modulus and Least Significant Bit, and to test SQL Injection attacks.

1.1 The indispensable allies of information security

Information security should not be reduced to a simple computer problem. The alliance of efforts with organizational security and the Law are the basis for enhancing more effective security and minimizing the incident consequences, even though reaching an infallible safety goal is hard to achieve.

It is in the area of Information Security in Organizations that complementary resources are often found to help the computer engineering mitigating the consequences of harmful interference in the various systems; by signalling known threats and vulnerabilities or by recommending good safety practices for physical, network, machine and software systems, by means of risk management, anticipation of scenarios, but also by the strategy and planning of the most appropriate response to each incident.

No less important in cybersecurity is lawful and regulatory compliance, whether at national, European or global level, which seeks to contextualize the use of computer systems in accordance with national laws. Cyberspace is not limited by geopolitical borders, so law seeks to regulate, harmonize, and improve cooperation and joint response to various incidents. In the European Union, an effort has been made to this goal, the GDPR - General Data Protection Regulation (Regulation (EU) 2016/679 of the European Parliament and of the Council, of 27th April 2016) and NIS, the European Network and Information Security Directive (EU 2016/1148 of the European Parliament and of the Council of 6th July 2016) are two good examples of this effort. [1]

2 Cryptography

2.1 The use of cryptography

In today's world we are dealing with an enormous amount of information, which we do not always want to be visible to everyone or which credibility we need to ensure. For this reason, we use encryption to ensure the confidentiality of the information, maintain the integrity and authenticity of the transmitted data, as well as the certification and authentication of individuals and entities, and to ensure non-repudiation.
Encryption is the science of writing secretly, in order to hide the meaning of a message. [2]
A cryptographic system is a set of algorithms (rules / instructions) and secret keys that allow transforming information which is intended to be secret, into something incomprehensible but which can be easily decoded by someone familiar with these algorithms and keys.
Cryptology or cryptographic science is organized into two distinct interconnected areas: cryptography and cryptanalysis. Cryptography, which can be subdivided into three broad areas: symmetric ciphers, asymmetric ciphers and protocols. The cryptanalysis tries to find the fragilities of the various cryptographic systems, with the purpose of deciphering the messages, without knowing the secret keys.

2.2 Encryption of messages

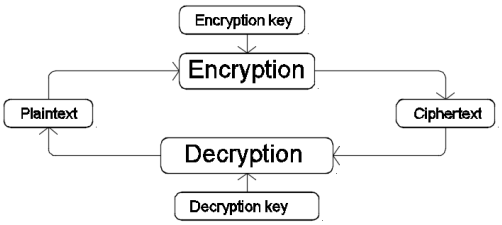

Fig1 - Encryption scheme of a message

2.3 The Classic or Symmetrical ciphers

The first ciphers were made through simple mechanisms of substitution (confusion) of characters by others or of permutation (dispersion) by changing the order of their elements. The secret keys were previously combined by the two interested parties through a secure channel, considering that the transmission channel might not be safe. [2] These figures can be divided into stream ciphers (Caesar cipher), which are characterized by coding one character at a time and in block ciphers (DES - Data Encryption Standard and AES - Advance Encryption Standard) that instead of coding each character sequentially, encrypt blocks, i.e., character sets. These types of ciphers are characterized by using confusion steps and diffusion steps successively, so that the diffusion spreads the confusion as best as possible.

2.4 The Asymmetric or Public Key ciphers

The distribution of keys has always plagued cryptographers, and there are many examples of these encountered difficulties throughout history [4]. Frequent switch of keys is a strong point in favour of security.

The cryptographic public-key system was introduced in 1976 by Whitfield Diffie and Martin Hellman, based on the idea that the encryption key doesn't have to be secret and that it may be publicly known. The receiver can only decode the message with its secret key which is private, and is not exchanged on an insecure channel.

2.5 Protocols

2.5.1 The Diffie-Hellman key exchange

The exchange of keys proposed by Diffie-Hellman came to enable communication without prior exchange of secret keys. The art of this game, is the use of an encryption algorithm with a one-way function, i.e., a function that is easily computable and simple and fast to encrypt the message, but it is extremely complicated to revert, except when a decipher key is provided, making it simple and fast to verify, thus allowing to decipher the received message.
The basic idea of Diffie-Hellman's proposition is that exponentiation of prime numbers is a one-way function, which enjoys commutative property, i.e., changing the order of its arguments does not change the final result. Thinking about these assumptions they defined a safe exchange of keys.
The process starts with the choice of a very large prime number p and another integer α such that α ∈ {1, 2, ..., p-2} that are publicly known, and the method develops as follows:

Alice		Bob
Knows **p** e $\boldsymbol{\alpha}$ Chose an integer $\boldsymbol{a} \in \{1,2,\dots,p-2\}$ and calculates $\boldsymbol{A} = \boldsymbol{\alpha}^{\boldsymbol{a}}_{mod_p}$ $\boldsymbol{a} = k_{prv,\boldsymbol{A}}$ is her private key $\boldsymbol{A} = k_{pub,\boldsymbol{A}} \equiv \boldsymbol{\alpha}^{\boldsymbol{a}} mod_{\boldsymbol{p}}$ *is* her public key		Knows **p** e $\boldsymbol{\alpha}$ Chose an integer $\boldsymbol{b} \in \{1,2,\dots,p-2\}$ and calculates $\boldsymbol{B} = \boldsymbol{\alpha}^{\boldsymbol{b}}_{mod_{\boldsymbol{p}}}$ $\boldsymbol{b} = k_{prv,\boldsymbol{B}}$ is his private key $\boldsymbol{B} = k_{pub,\boldsymbol{B}} \equiv \boldsymbol{\alpha}^{\boldsymbol{b}} mod_p$ is his public key
Sends A	A →	
	B ←	Sends B
Calculates $\boldsymbol{k_{ab}} = \boldsymbol{B}^{\boldsymbol{a}} = {k_{pub}^{\boldsymbol{b}}}^{\boldsymbol{a}}$		Calculates $\boldsymbol{k_{ba}} = \boldsymbol{A}^{\boldsymbol{b}} = {k_{pub}^{\boldsymbol{a}}}^{\boldsymbol{b}}$

Fig2 - Diffie-Hellman Key Exchange

$$B^a = (\alpha^b_{mod_p})^a = \alpha^{ba}_{mod_p} = \alpha^{ab}_{mod_p} = (\alpha^a_{mod_p})^b = A^b$$

3 The RSA cryptographic system

3.1 The RSA Algorithm

In 1977, Ronald Rivest, Adi Shamir and Leonard Adleman proposed a new cryptographic system, the RSA, based on asymmetric ciphers and which is nowadays widely used.
The one-way function underlying the RSA is based on the problem of factorizing large integers into prime numbers.
The algorithm of the cryptographic system can be summarized as follows [2]

1 – Two large prime numbers are chosen, $\boldsymbol{p}$ e $\boldsymbol{q}$.
2 – $\mathbf{n} = \mathbf{p} \cdot \mathbf{q}$
3 – $\boldsymbol{\phi}(\mathbf{n}) = (\mathbf{p} - 1).(\mathbf{q} - 1)$
4 – An exponent is chosen $\mathbf{e} \in \{1,2, \ldots \phi(\mathbf{n}) - 1\}$ such that $gcd(\mathbf{e}, \boldsymbol{\phi}(\mathbf{n})) = 1$
5 – The private key is given by $\mathbf{d} \cdot \mathbf{e} = 1 \mathrm{mod}_{\boldsymbol{\phi}(\mathbf{n})}$
As a result, the keys are defined:
The public $\mathbf{k_{pub}} = (\mathbf{n}, \mathbf{e})$
and the private $\mathbf{k_{pr}} = (\phi(\mathbf{n}), \mathbf{d})$,
6 – With this information it is possible to encrypt the message

$$\boldsymbol{y} = x^e \mathrm{mod}_{\mathrm{n}}$$

7 – With the private key it is possible to discover the original message $\boldsymbol{x} = y^e \mathrm{mod}_{\mathrm{n}}$

RSA is an algorithm that requires high data processing and therefore is slow, but its robustness is mainly used for the exchange of secret keys. Once the confidentiality of the keys is guaranteed, it is often replaced by other symmetric key algorithms, such as AES, which are significantly faster. [2].

3.2 Cryptanalysis

Cryptanalysis seeks to find vulnerabilities in cryptographic algorithms that allow the deciphering of messages or discovering secret keys. In order to do so, it uses the algorithms and tries to find weaknesses that allow reversing the cipher process. With the increase of the algorithms complexity, even accompanied by the technological evolution that allowed an extraordinary increase of the speed of verification of all the hypotheses, the time necessary to break the most complex ciphers became sometimes too long, preventing the task to be done successfully in a timely way. There are no unbreakable ciphers, however some are so arduous to break, requiring such a long time that in practice they could be considered safe.

3.3 Computability and Complexity

The Theories of Complexity and Computability are related. The first seeks to classify the problems into difficult or easy and the second search classifies them according to whether or not they have a solution. In most fields, researchers' efforts are made to reach simple and fast algorithms to solve all kinds of problems, but in cryptography these efforts go in the opposite direction, looking for complex algorithms that although computable, without a key, require such an effort and such a long time that in practice they are unable to find a solution in a timely manner.

The robustness of the RSA cryptographic system is based on the problem of factorization, i.e., on the difficulty of factorizing a number that is the product of two very large prime numbers. So, the choice of small p and q can dangerously facilitate the ciphers break.

3.4 Symmetric Key Security vs Public Key

The strength of a cipher is related to the difficulty of finding its key. This obstacle depends on both the complexity and efficiency of its algorithm, and the length of the key. To achieve the same level of security, different ciphers with distinct algorithms require different key lengths.
According to Bob Cromwell, citing Schneier's and NIST / NSA comparative study of the key of the various systems, to achieve the same level of security, it is possible to observe that a 1024-bit RSA key offers an equivalent security to an 80-bit key used in a symmetric key cryptographic system.

Key Length in Bits for Approximately Equal Resistance to Brute-Force Attacks, per Schneier

Symmetric Encryption	56	64	80	112	128	256	448	1024
Cryptographic Hash	112	128	160	224	256	512	896	2048
Asymmetric Encryption	384	494	768	1792	2304	10753	39031	234205

Key Length in Bits for Approximately Equal Resistance to Brute-Force Attacks, per NIST/NSA

Symmetric Encryption	80	112	128	192	256
Elliptic Curve encryption	160	224	256	384	512
RSA (asymmetric encryption)	1024	2048	3072	7680	15380

Fig3- Comparison of key length for the same resistance to brute force attacks.[5]

4 Attacks on the RSA cryptographic system

4.1 Wiener's Attack

This attack on the RSA cryptographic system, described by Michael J. Wiener, shows vulnerability detected when the private key is small. [5] [6]

The conditions for this attack to be successful are as follows:

$$3 \cdot \mathbf{d} < \mathbf{n}^{1/4} \quad \text{and} \quad \mathbf{q} < \mathbf{p} < 2 \cdot \mathbf{q}$$

If $\mathbf{n}$ has $\boldsymbol{l}$ bits in binary representation, then the attack results when $\mathbf{d}$ has less than $\boldsymbol{l}/\mathbf{4} - \mathbf{1}$ bits and if $\mathbf{p}$ and $\mathbf{q}$ are not too far from each other.

The option of choosing a small decipher key refers to the speed that can be gained in deciphering. [2].

Demonstration:

$\boldsymbol{K}_{\mathbf{pub}}$ $(\mathbf{n},\mathbf{e})$ are public; $3 \cdot \boldsymbol{d} < \boldsymbol{n}^{1/4}$ and $\boldsymbol{q} < \boldsymbol{p} < 2 \cdot \boldsymbol{q}$

Given that $\mathrm{d} \equiv e^{-1} \mathrm{mod}_{\phi(\mathrm{n})}$ or written in another form $\mathrm{d} \cdot \mathrm{e} \equiv 1 \mathrm{mod}_{\phi(\mathrm{n})}$, then:

$\exists_{t \in \mathbb{N}} : \boldsymbol{e} \cdot \boldsymbol{d} - \boldsymbol{t}\boldsymbol{\phi}(\boldsymbol{n}) = 1$

And d can be found

Being $n = p.q$, then:

$$q < p < 2 \cdot q \qquad q < \sqrt{n} < q \cdot \sqrt{2}$$

Consequently:

$$0 < n - \phi(n) = p + q - 1 < 2 \cdot q + q - 1 < 3\sqrt{n}$$

So:

$$0 < n - \phi(n) < 3\sqrt{n}$$

On the other hand,

$$\left|\frac{e}{n} - \frac{t}{d}\right| = \left|\frac{e \cdot d - t \cdot n}{n \cdot d}\right|$$

Bearing in mind that $e \cdot d = 1 + t\phi(n)$, it is possible to replace

$$\left|\frac{1+t\cdot\phi(n)-t\cdot n}{n\cdot d}\right| < \left|\frac{1-t(3\sqrt{n})}{n\cdot d}\right|$$

$$\left|\frac{1-3t\sqrt{n}}{n\cdot d}\right| < \left|\frac{3t}{\sqrt{n}\cdot d}\right|$$

Because$t < d$then$3\cdot t < 3\cdot d$,

in turn, one of the conditions of applicability of this attack is

$$3\cdot d < n^{1/4},$$

so $3\cdot t < 3\cdot d < n^{1/4}$,

and it is possible to say:

$$\left|\frac{e}{n}-\frac{t}{d}\right| < \left|\frac{3t}{\sqrt{n}\cdot d}\right| < \left|\frac{\sqrt[4]{n}}{\sqrt{n}\cdot d}\right| = \left|\frac{1}{\sqrt[4]{n}\cdot d}\right|$$

Finally, it is possible to conclude

$$\left|\frac{e}{n}-\frac{t}{d}\right| < \left|\frac{1}{\sqrt[4]{n}\cdot d}\right| < \left|\frac{1}{3\cdot d\cdot d}\right|$$

$$\left|\frac{e}{n}-\frac{t}{d}\right| < \left|\frac{1}{3\cdot d^2}\right| < \left|\frac{1}{2\cdot d^2}\right|$$

There is a property of the theory of continuous fractions, which defines that:

If gcd(a,b) = gcd(c,d) = 1 and if

$$\left|\frac{a}{b}-\frac{c}{d}\right| < \left|\frac{1}{2\cdot d^2}\right|$$

then $\frac{c}{d}$ is a convergence of the continuous fraction expansion $\frac{a}{b}$

.

Knowing that

$$\left|\frac{e}{n}-\frac{t}{d}\right| < \left|\frac{1}{2\cdot d^2}\right|$$

it is possible to calculate $\frac{t}{d}$ as a convergence of the continuous fraction expansion of $\frac{e}{n}$, given that **e** and **n** are public information.

$$e\cdot d - t\phi(n) = 1$$

$$\frac{e}{\emptyset(n)}-\frac{t}{d} = \frac{1}{\emptyset(n)\cdot d}$$

Knowing that **p** and **q** are large numbers, it is possible to conclude that:

$$p-1 \approx p \quad \text{and} \quad q-1 \approx q,$$

$$\text{then } \phi(n) = (p-1)(q-1) \cong p\cdot q = n,$$

on the other hand being $\boldsymbol{\phi(n)}$ a big number and **d** an integer, then it is legitimate to conclude that their product is a large number, so being

$$\frac{1}{\emptyset(n)\cdot d} \approx 0$$

Rewriting the equation

$$\frac{e}{\emptyset(n)}-\frac{t}{d} = \frac{1}{\emptyset(n)\cdot d}$$

that is

$$\frac{e}{n} \approx \frac{t}{d}$$

Representation of numbers through continuous fractions

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cdots + \cfrac{1}{q_n}}}$$

In this way it is possible to determine $\boldsymbol{\phi(n)}$

Knowing in advance that:

$$\phi(n) = \frac{(e\cdot d-1)}{t}$$

it is then possible to factorize n by solving the quadratic equation:

$$\begin{cases} n = p\cdot q \\ \phi(n) = (p-1)\cdot(q-1) \\ \cdots \\ p^2 - [n-\phi(n)+1]\cdot p + n = 0 \end{cases}$$

This equation can be calculated using the resolvent formula.

Thus, by factoring n into the prime numbers, **p** and **q**, it is possible to determine $\boldsymbol{\phi(n)}$ and consequently calculate the private key $\mathbf{d} \equiv \boldsymbol{e^{-1}}\mathrm{mod}_{\boldsymbol{\phi(n)}}$. [6]

4.2 Common Modulus Attack

If a message is encrypted and sent to two users whose public keys are formed by the same module **n** and their cipher keys are co-primes, then any other user who has access to those cipher messages can decipher the text [7] [8] [9]

n = p · q; $k_{pub}^{A}(e_A, n)$; $k_{pub}^{B}(e_B, n)$ and $mdc(e_A, e_B) = 1$

m – Plaintext;

$$\begin{cases} c_A = m^{e_A} mod_n \\ c_B = m^{e_B} mod_n \end{cases}$$ Cyphertexts

Then it is possible to know the message without prior knowledge of private keys

$k_{pr}^{A}(d_A, \emptyset(n))$, or $k_{pr}^{B}(d_B, \emptyset(n))$.

$C_A^{a_A}\cdot C_B^{a_B} \cong m^{a\cdot e_A}\cdot m^{b\cdot e_B} \cong m^{a\cdot e_A + b\cdot e_B} \cong m^1 \cong m_{mod_n}$

Using Extended Euclidean Algorithm to determine a_A e a_B

$$a\cdot e_{A+} b\cdot e_B = gcd(e_A, e_B) = 1,$$

4.3 Least-Significant Bit Attack

This attack uses public data from the RSA cryptographic system, $k_{pub}^{A}(\mathbf{e}, \boldsymbol{n})$, to encrypt the message 2.m. It presupposes the previous knowledge of the cyphertext. Using a parity oracle $\boldsymbol{PN_o}$ that can determine whether the value is even or odd Thus it is possible to go dividing in half the interval at which the message is, and then after several iterations, it can determine the least significant bit of the original message [3].

Applying this algorithm to each bit of the message, i.e., $\log_2 n$ times, it is possible to find the plaintext.

$$(2\cdot m)^e mod_n = 2^e m^e mod_n = 2^e \cdot C mod_n$$

If it is an even number, then 2.**m** lies in the first congruence interval

$$0 < m < \frac{n}{2}$$

Else, if it is an odd number then 2.**m** lies in the second congruence interval

$$\frac{n}{2} < m < n$$

By successively dividing these intervals it is possible to determine the plaintext.

4.4 Prevention and countermeasures of RSA attacks

RSA is the standard cryptographic system for public key encryption and signatures and until now, no effective way of breaking it has yet been found. However, in particular circumstances, as previously described, it is possible to break this robust cryptosystem based on an unfortunate choice of its parameters.

According to the attacks already mentioned, the choice of parameters should take into account the following aspects:

- Module **n**, resulting from the product of two prime numbers, must be large, i.e., must be at least 1024 bits and should not be used as part of the public key for multiple users.

- The private key must not be small. Being $\boldsymbol{l}$ the number of bits of **n**, then **d** must have at least

$$^{l}/_{4} - 1 \text{ bits, i.e., } ^{1024}/_{4} - 1 = 255 \text{ bits,}$$

- When the relationship between **p** e **q** is:

$$\boldsymbol{q} < \boldsymbol{p} < 2 \cdot \boldsymbol{q},$$

the private key must be such that:

$$3 \cdot \boldsymbol{d} > \boldsymbol{n}^{1/4},$$

- In the case of attacks on RSA implementation, the careful choice of parameters is not enough to avoid them. However, in particular cases, it is possible to adopt some measures to avoid this kind of flaw.

There are many other attacks that exploit various weaknesses of the parameters chosen, namely the choice of a small public key, which allow breaking the RSA. However, it has not been possible to find an algorithm that breaks this cryptographic system effectively, in a timely manner for all situations.

5 Security Laboratory Prototype- Cryptography

As an integral part of this thesis, a security laboratory prototype, based on the three previously described attacks, was developed. It decrypts the messages that were intercepted, when the parameter choices were wrongly done or when they were used repeatedly for several users. This simulation was performed in virtual machines and developed in Python, using the Github source code disclosure platform [7].

5.1 Roadmap for resolution

This application was performed in a virtual machine (Oracle's VirtulBox), with the Linux Ubuntu operating system (16.04.2, 64-bit) and with installation of modules Python and Sage.
These attacks were simulated using a server application, that for each case generates the public key and a random message, according to the fragilities described above, and uses the algorithms described for each one that allow the decipher of the messages. The hacker makes the attack and the server verifies that it was successful.

5.2 Wiener's Attack

Description: The generated data (**c**, **e**, **n**) are used by the hacker to decipher the message, according to the algorithm described above. This message is sent to the server and verified. If it is correct the following comment is returned "Correct.".
Dialogue between the hacker and the server:

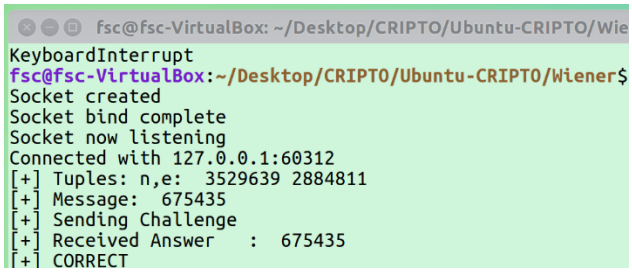

Fig4 – Wiener's Attack – Server's Terminal

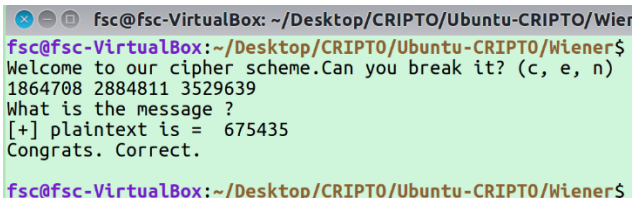

Fig5 - Wiener's Attack –Hacker's Terminal

5.3 Common Modulus Attack

Description: The generated data (**n**, $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{e}_1$, $\mathbf{e}_2$) are used by the hacker to decipher the message, according to the algorithm described above. This message is sent to the server and verified. If it is correct the following comment is returned "CORRECT
Dialogue between the hacker and the server:

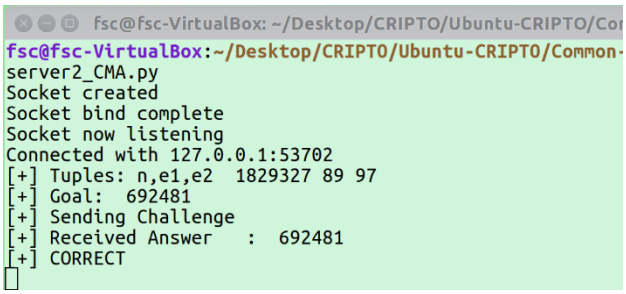

Fig6- Common-modulus Attack –Server's Terminal

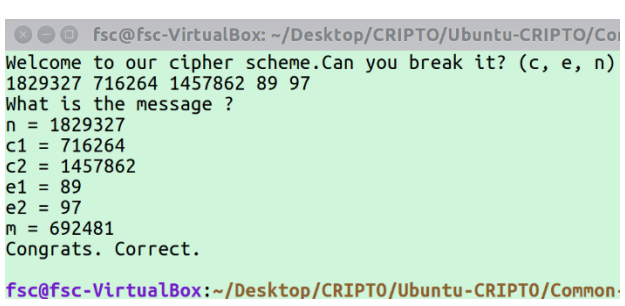

Fig7 - Common-modulus Attack –Hacker's Terminal

5.4 Least Significant Bit Attack

Description: The generated data (**c**, **e**, **n**) are used by the hacker to decrypt the message, according to the algorithm described above and each iteration is sent to the Oracle (server) that returns the parity. This dialog ends when the range contains only one possible message and returns the comment "CORRECT".
Dialogue between the hacker and the server:

```
fsc@fsc-VirtualBox: ~/Desktop/CRIPTO/Ubu
fsc@fsc-VirtualBox:~/Desktop/CRIPTO/Ubuntu-CR
IPTO/LSB$ python server2_LSB.py
Socket created
Socket bind complete
Socket now listening
Connected with 127.0.0.1:51900
cyphertext =  29409
[+] Tuples: n,c,e 2052683 29409 13
[+] Goal:  1767190
[+] Sending Challenge
[+] Received Answer   :  754617
1
[+] Received Answer   :  1193951
1
...
[+] Received Answer   :  1420195
0
[+] Received Answer   :  1767190
[+] plaintext  :  1767190
[+] CORRECT
**********************************************
```

Fig8 - Least Significant Bit Attack –Server's Terminal

```
fsc@fsc-VirtualBox: ~/Desktop/CRIPTO/Ubuntu-CRIPTO/LSB
fsc@fsc-VirtualBox:~/Desktop/CRIPTO/Ubuntu-CRIPTO/LSB$ sage LS
B-Attack.py
n =  2052683
c =  29409
e =  13
[ 0 2052683 ]
754617
i =  0   parity  1 [ 1026341 2052683 ]
1193951
i =  1   parity  1 [ 1539512 2052683 ]
...
i =  18   parity  0 [ 1767188 1767191 ]
703528
i =  19   parity  1 [ 1767190 1767191 ]
1420195
i =  20   parity  0 [ 1767190 1767190 ]
fsc@fsc-VirtualBox:~/Desktop/CRIPTO/Ubuntu-CRIPTO/LSB$
```

Fig9 - Least Significant bit Attack –hacker's Terminal

6 Web Attacks

6.1 Data Bases

Databases (DB) are an efficient way to store, manage and relate information. At present, databases are mostly used by various sites to interact and exchange information with users. These pages generally have an intuitive interface for communicating with clients, where they can search, or authenticate, i.e. where they can do some type of data input. Subsequently, the page communicates with its databases and returns information, or validates permission to access a reserved area [8].

6.2 Web Attacks

In Web pages where some kind of data can be input, it can be used to trick and manipulate the websites, to reveal and manipulate the information, or to allow inadvertent access.

6.3 SQL

Structured Query Language (SQL) is a standard programming language for storing, manipulating, and returning database information (DB). With SQL you can create databases and their tables, enter data, change them, delete them, and also do research. For researching examples on SQL queries it is advisable to look at sites like [9], [10] or [11], among many others.

6.4 SQL injection (SQLi)

SQL injection is a technique that uses the inputs of data from on a web page, for example a user's request for authentication, or a website's search engine, to enter code in order to manipulate that information. It is the most common form of attack on these pages [12].Throughout these fields it is also possible to introduce malware to control other elements of the system, such as gaining administrator privileges.

6.4.1 Classic SQL Injection Attacks

When some type of authorization is required to access a reserved area of a site, this check is usually done by checking the user name and password in the web site database.

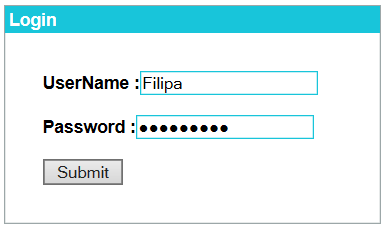

Fig10–Example of an authentication page

SQL injection uses insertion of authentication fields, to introduce code that SQL recognizes as commands, or instructions, and causes it to inadvertently reveal or change database information.

In general, verification of the authentication is done through a logical condition:

> **"SELECT** * **FROM** tabela_de_utilizadores **WHERE** UserName = 'Filipa' **AND** Password = 'password' ";

If there is a register whose UserName is 'Filipa' and its Password is 'password', then the access to the reserved area is given

A typical attack is to introduce a character set that will cause the program error (Error-based SQLi), i.e., to lead the machine to conclude that the condition you are checking is true. The art of these attacks is to modify the condition to be verified by making use of the syntax of the SQL language

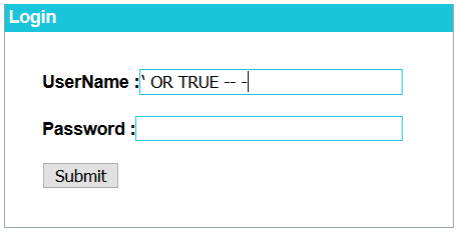

Fig11 – Classic attack of SQL Injection.

> **"SELECT * FROM** tabela_de_utilizadores **WHERE** UserName='' or true -- -' **AND** Pass =' ' **";**

SQL uses the string "-- "(two dashes and one space) to start a comment. By these way the original query statement is changed, because all the conditions that were found after these special characters are ignored.

This is the most frequent type of SQL Injection attack. The attacker changes the query syntax so as to be able to manipulate SQL queries, obtaining permissions, information and various privileges. Observing and compiling the results of this trial-and-error technique is often the route used by hackers to attack a site.

It should be noted that SQL is very sensitive to syntax rules, so it is not always easy to find the correct string to do the intended manipulation. Querying sites, [10], [11], [12], is useful for identifying SQL and SQLi language samples that can be exploited in an attack on a web page.

6.4.2 SQLi inference attacks

Sometimes the resulting error from the manipulation of the query is not visible for attacker, although it is possible to make some revealing deductions of information or the characteristics of the DB. This type of onslaught called Blind SQL Injection can reveal useful information, either through the logical result of the evaluated condition (Boolean-based), or through the time it takes to perform a task (Time-based)

An example of the application of this technique can be described as follow:

The hacker types the sequence in the user's field:

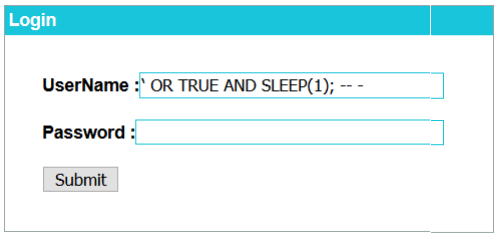

Fig12 – Inference attack SQL Injection

That applied to the original query, changes it and builds the following query:

"SELECT * FROM **family.familia** WHERE **NAME** ='OR TRUE AND SLEEP(1); -- -"

In this case the search delays the response as many seconds as the number of rows of the table.

The sleep function is recognized by Mysql, a database management program that uses the SQL language, and in this case delays its task 1 second. The time that the search takes to go through all the rows of the table, without any retarder appears to be null, so the amount of seconds that this delayed search takes to execute, is approximately identical to the number of rows of the table.

Non-validation of inputs is a vast area that can be applied to several languages. The detection and exploitation of vulnerabilities depends on the system used in the DB, the target of the attack, and the programming developed for the application, but in a generic way a hacker's approach is done by following these steps [13]:

1 –Tests the introduction of data that may cause misinterpretation, or that provokes errors in the language found;

2 – Identifies and evaluates the anomalies of the responses that can be indicators of the existence of vulnerabilities;

3 – If there are explicit error messages, interprets them carefully;

4 – Introduces variants of the inserted data to verify the existence of the vulnerability that the hacker suspects to exist;

5 – Formulates the hypothesis of the existence of vulnerability and proves, testing as many times as possible;

6 – Explores vulnerability variants found to achieve his goal.

6.5 Countermeasures to avoid SQL Injections

As mentioned above, there are no measures to ensure 100% effective safety, but there are, however, rules of good practice for the development of such risk-minimizing systems which can and should be kept up to date. A simple way to avoid the situations described above, and that follows the recommendations of the lemma Safe by Design and Safe by Default, is to prevent the introduction of some special characters, or to treat this input as a mere text, without any other meaning beyond a sequence of characters, thus preventing it from being interpreted as a command.

During the development of an application the exposure of the errors found is a fundamental tool for debugging, verifying that the program proceeds correctly and does what it proposes. These comments should, however, be hidden or even deleted in the final versions so that they do not become a good source of information, especially on its structure, for a potential attacker.

Safe by Design and Safe by Default are two concepts that can be described in a simplified way: all software must be thought and developed avoiding the existence of vulnerabilities, must do exactly what it proposes and by default must have the maximum security possible. In addition to these rules, users should have only the access privileges strictly necessary to perform their functions.

6.6 Mod Security

A Firewall is a device that monitors and filters incoming and outgoing traffic from a network according to pre-defined security rules. This filter, or shield, can be accomplished through hardware, software, or both. As the name implies it is an effective way to defend a network of known and unwanted intrusions.

Mod Security is an open source web application firewall (WAF) developed by Trustwave's SpiderLabs [14].It is a tool that monitors HTTP traffic, web applications, logins (authentications) and access controls. It is flexible and allows to choose the rules someone want to adopt for detection or blocking.

This application analyses the data entry and checks if any of the predefined rules have been broken. When so it registers this data in its log file (modsec_audit.log). In this file it can be find information about which rules have been violated [15].

This application is in constant evolution, through the contribution of users from all over the world, who in this way help to eliminate the false positives detected (traffic that is classified as transgressor, but after being analysed it proves not to be) and to define new rules.

7 Security Laboratory Prototype- SQL Injection

As part of this thesis, a security lab prototype was developed, which uses a user authentication page and in which data such as malicious code can be entered. This laboratory was simulated through two virtual machines, one that allows unrestricted data entry and which results in the possibility of undue authorization in a reserved area, or in the disclosure of information in the database. The second one, protected by the Mod application Security, a tool that is compatible with the Apache2 server, detects and blocks (chosen option) data entry that, according to pre-established rules, is identified as malicious code.

7.1 Resolution script

This testing environment was run on two virtual machines (Oracle's VirtulBox), with the Linux Ubuntu operating system (16.04.2, 64-bit). In one of the machines the Mod Security application was also installed. This simulation was developed in MySQL (data base), PHP (communication with the Apache2 server) e HTML e CSS (web page).

In these pages it is possible to input data and observe the result of some SQL Injections

7.2 Web Attacks

Description: This attack was simulated using a database (family) that consists of two tables (familia and guitarras) and an authentication web page with two fields: username and password. The purpose is to test strings that cause a SQLi attack, which eludes the original query, allowing authentication and fraudulent access to the Welcome page, changing the DB, or revealing some kind of information.

Query for the authentication:

"SELECT * FROM family.familia WHERE NAME = '$myusername' and PASSWORD = '$mypassword' ";

Inputs suggestions:

1) Username : Vasco
   Password : vasco123
2) Username : ' or true --
   Password :
3) Username : ' OR true ORDER BY NAME; -- -
   Password : 1234
4) Username: ‘ OR TRUE AND SLEEP(1); -- -
   Password :

Expected results:

1) Vasco is an authorized user. This is why he is redirected to the Welcome page. The "Sign Out" link returns to the Login page.

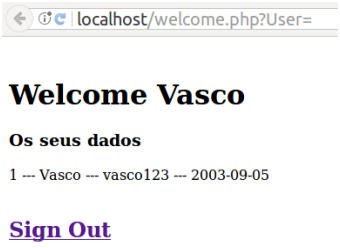

Fig13 - Result of legitimate authentication

2) This attack reveals all information in the database and has the particularity of assuming that the user is the first name of the list. This point is particularly interesting because this position often corresponds to the database administrator, who has special privileges and can manipulate this information with few or no restrictions

Fig14–Extract from SQLi result Username = ‘ or true –

3) This attack reveals all information in the database, sorts it alphabetically from the NAME column and assumes the user is the first name in the list

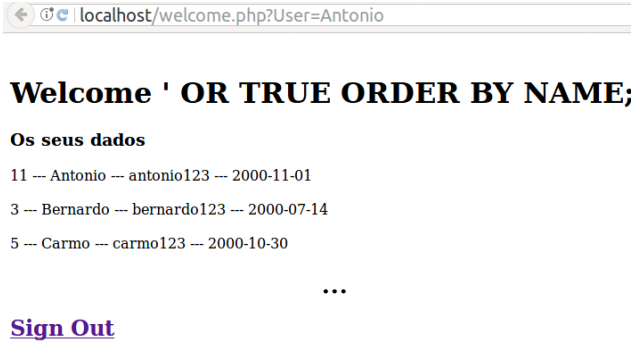

Fig15- Extract from SQLi result Username = ' OR TRUE ORDER BY NAME; -- -

4) In this case, entry into the reserved area is not allowed.

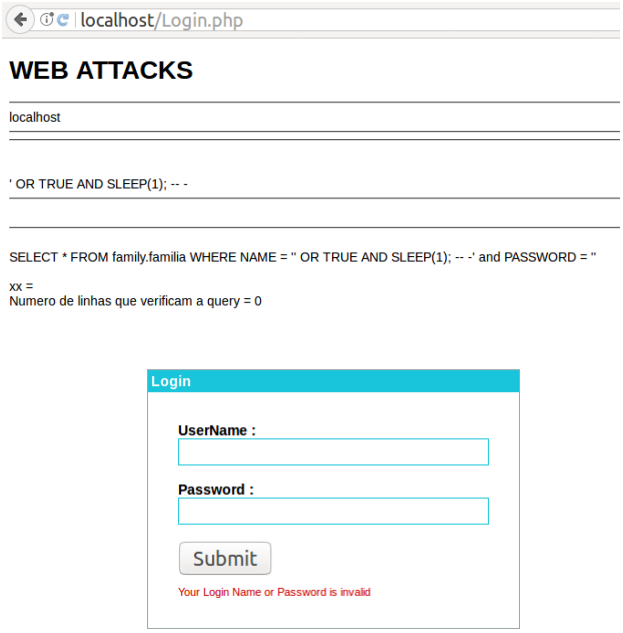

Fig16 - SQLi result Username = ' OR TRUE AND SLEEP(1); -- -

In this last attack it is possible to observe that although the conditions for accessing a reserved area are not met, the time to return the denial of access response is of the order of 18 seconds, indicating that the table should have about 17 or 18 registers.

Although not part of the scope of this thesis, this application is also vulnerable to other types of attacks such as XSS (Cross-site scripting Injection), as it can be seen by typing the following JavaScript code in the form

http://localhost/?param="><script>alert("XSS INJECTION !!!");</script> [16]

In this case it opens a box with the message "XSS INJECTION!!!".

7.3 Web Attacks + ModSecurity.

Description: This attack was simulated using a virtual machine identical to the first one, but in which the web application firewall (WAF) Mod. was installed.

By making the same inputs it is possible to observe:

1) O Vasco is an authorized user. He is redirected to the Welcome page again. In this case the Mod Security application do not display any message.

Fig17 – Result of legitimate authentication in Mod Security presence

2) In this attack at least one of the predefined rules is violated. Mod Security locks the page and logs the event in the modsec_audit.log file.

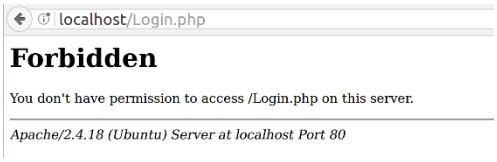

Fig18 - SQLi result Username = ' or true -- with Mod Security

3) As in the previous case Mod Security blocks the page and once again registers the event in the modsec_audit.log file.

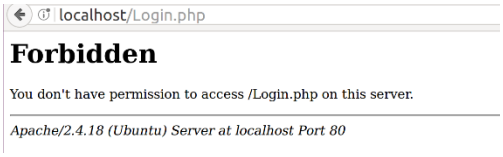

Fig19- SQLi result Username = ' OR TRUE ORDER BY NAME; -- - with Mod Security

4) In this case, the response is instantaneous; it does not take several seconds as it happens on the machine without Mod Security installed. This locks the page and once again logs the event in the modsec_audit.log file.

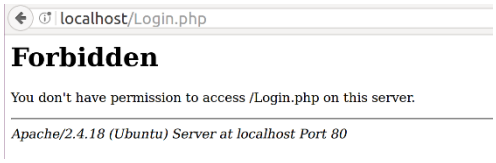

Fig20 - SQLi result Username = ' OR TRUE AND SLEEP(1); -- - with Mod Security

When scanning the modsec_audit.log file, which registers malicious data entry, it can be verify that Mod Security identifies these entries as SQLi, with the maximum rating (10).

8 Conclusion

Technological evolution in the digital area has revolutionized the way people communicate, the internet has accelerated its relationship with the world, bringing enormous advantages in communication in many areas, but for this type of system to work and to be trusted, it is necessary that confidentiality, authenticity and integrity are ensured. Encryption is a powerful weapon to achieve these goals. This ancient science has developed complex ciphers, whose ciphering algorithms, though computable, are extremely difficult to reverse without the knowledge of secret keys. Despite this difficulty, there are situations in which the robustness of a cipher, and in particular of the RSA cryptographic system, can be compromised when a selection of parameters is not very judicious. The Wiener, Common Modules and LSB attacks are examples of weaknesses that arise from choosing a small private key, or the option of a **n** module, common to several users whose public

keys are co-primes, or to vulnerabilities, different from the parameters choice of, but made directly to the cryptographic implementation, which weaken the RSA and enable the timely deciphering of intercepted messages.

In this work, a test environment was developed that simulates the Wiener, Common Modulus and LSB attacks, in which the various parameters are generated, according to the characteristics of each one, and where it is possible to demonstrate that in the presence of the described vulnerabilities it is feasible to decipher the messages.

The security of communications can be doubted in many other ways. Information is a valuable asset. Databases are an efficient way to store, manage and relate information. The websites and their pages usually use databases to interact with users, allowing them to authenticate or respond to searches, among other features, always limited by the privileges of access granted to each one. The most common relationship is done through authentication to a reserved area, or in a search space. In response, they are given the privilege of entering those areas or receiving some kind of information that they are allowed to see. These input fields can be used by a hacker to deceive the machine by forcing it to reveal information. SQL Injection is an effective technique for manipulating searches performed in a database, in a way that inadvertently reveals and manipulates information or allows undue access. Building applications that comply with the Safe by Default and Safe by Design rules is a good principle. It is an issue that should always be present in the programming, but it is even better to combine these rules with a firewall that detects or blocks any malicious code, such as the Mod Security application. This application can help programmers bridge the vulnerabilities that arise over the life of the programs by leveraging the community's contribution to discovering new vulnerabilities. The application of these measures helps to make security of database access more robust and helps to ensure that the information is not inadvertently revealed.

Information security should not be reduced to a simple computer problem. The cooperation of efforts of the areas of Information Security in Organizations and Law with the technical area are essential to complement computer security. Last but not least, it is impossible to leave out the invaluable contribution of continuous user training, which is crucial to enhancing more effective security and minimizing the consequences of incidents, even though the objective of making safety foolproof is difficult to achieve. Test environments are an effective way to learn and test weaknesses in various systems. The theoretical and practical learning of rules that contribute to the safe use of computer systems is essential. The ability to experiment and verify the extent of errors in test environments without real consequences is an effective way to learn. Targeting biginner users in the area of Information Security, and after presenting the theoretical foundations that help to understand the proposed laboratories, test environments have been developed to simulate the vulnerabilities of the RSA cryptographic system (the Wiener attacks, Common Modulus and LSB) and to simulate a web aplication vulnerable to SQLi attacks. In the first, the various parameters are generated, according to the characteristics of each one, and it is possible to demonstrate that, in the presence of the described vulnerabilities, it is possible to decipher the messages. In the second, an authentication page was created that allows input data and where it is possible to test SQLi attacks. The developed prototype allows the future construction of new test modules for carry out trials with other types of vulnerabilities.

The files of this applications can be found in https://fenix.tecnico.ulisboa.pt/cursos/msidc/dissertacoes

Bibliography

[1] ANACOM, 14 Agosto 2017. [Online]. Available: https://www.anacom.pt/render.jsp?contentId=1374434.

[2] C. Paar and J. Pelzl, Understanding Cryptography, Springer, 2010.

[3] S. Singh, The Code Book, London: Fourth Estate, 1999.

[4] B. Cromwell, “Cipher Strength and Key Length,” 28 Outubro 2017. [Online]. Available: https://cromwell-intl.com/cybersecurity/crypto/cipher-strength.html.

[5] D. R. Stinson, Cryptography. Theory and Practice (Third Edition), CRC Press, 2005.

[6] M. J. Wiener, “Cryptanalysis of Short RSA Secret Exponents,” *IEEE Transaction of Information Theory, vol 36, nº3,* Maio 1990.

[7] J. Katz and Y. Lindell, Introduction to Modern Cryptography, USA: Chapman & Hall/CRC, 2008.

[8] F. d. C. Boucinha, *A Survey of Cryptanalytic Attacks on RSA,* 2011.

[9] H. I. d. S. Q. Leite, *Ataques ao Sistema Criptográfico RSA,* 2009.

[10] W. Mao, Modern Cryptography, Theory & Practice, U.S.A.: Hewllet-Packard Company - Prentice Hall, 2004.

[11] GitHub, Janeiro a Outubro 2017. [Online]. Available: https://github.com/.

[12] Microsoft, “Noções básicas da base de dados,” 5 Agosto 2017. [Online]. Available: https://support.office.com/pt-pt/article/No%C3%A7%C3%B5es-b%C3%A1sicas-da-base-de-dados-a849ac16-07c7-4a31-9948-3c8c94a7c204.

[13] KhanAcademy, “Noções básicas de SQL,” 27 Outubro 2017. [Online]. Available: https://pt.khanacademy.org/computing/computer-programming/sql/sql-basics/v/welcome-to-sql.

[14] W3school, “SQL Tutorial,” Maio a Outubro 2017. [Online]. Available: https://www.w3schools.com/sql/.

[15] Codecademy, “SQL,” Maio a Outubro 2017. [Online]. Available: https://www.codecademy.com/catalog/language/sql.

[16] OWASP, “Testing for SQL Injection (OTG-INPVAL-005),” 19 Setembro 2017. [Online]. Available: https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)#Detection_Techniques.

[17] D. Stuttard and M. Pinto, The Web Application Hacker's Handbook, Indianapolis: Wiley Publishig, Inc., 2011.

[18] I. Trustwave Holdings, “Mode Security - Open Source Application Firewall,” 29 Setembro 2017. [Online]. Available: https://modsecurity.org/download.html.

[19] I. Trustwave Holdings, “SpiderLabs/ModSecurity,” 30 Setembro 2017. [Online]. Available: https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-2-Data-Formats#Alert_Action_Description.

[20] S. d. Paola and L. Carettoni, “Testing for Reflected Cross site scripting,” 31 Outubo 2017. [Online]. Available: https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001).