University of Amsterdam

System and Network Engineering

Master Thesis

Feasibility of attacks against weak SSL/TLS ciphers

Student:
Kim van Erkelens

Supervisors:
Jeroen van der Ham
Marc Smeets

July 2014

UNIVERSITY OF AMSTERDAM

Abstract

Graduate School of Informatics
System and Network Engineering

Master of Science

Feasibility of attacks against weak SSL/TLS ciphers

by Kim van Erkelens

Weak ciphers as used in Secure Sockets Layer (SSL) and Transport Layer Security (TLS) have been around since many years. In theory those ciphers are feasible to crack, but in current networks the usage of weak ciphers is still very common. This research strives to evaluate the practical feasibility of cracking weak SSL/TLS ciphers in order to find out what the real risk is that the usage of such ciphers expose. Investigated is the decryption of HTTPS and Microsofts' Remote Desktop Protocol (RDP). Finally, the feasibility for cracking the encryption keys is defined.

Acknowledgements

I would like to thank Jeroen van der Ham and Marc Smeets for their supervision of this project and the critical reading of earlier drafts of this thesis. I greatly benefited from their suggestions.

Contents

Chapter 1

Introduction

Weak ciphers as used in Secure Sockets Layer (SSL) and Transport Layer Security (TLS) have been around since many years. In theory many ciphers are feasible to crack. But in current networks the usage of weak ciphers is still very common. According to SSL Pulse one third of the sites surveyed still use RC4 with modern browsers, and almost 60 percent have some RC4 suites enabled [1]. Suprisingly, in practice only a few attempts of cracking weak ciphers have been successful. Many other theoretically weak ciphers are still not easy to crack. For instance cracking them would require a lot of money and resources, and is therefore only within reach of large organisations.

SSL/TLS is widely used for securing traffic over the Internet and used for various applications. Besides web and HTTPS traffic the protocol can be used for SMTP, instant messaging, and VPNs among many other applications. This project investigates both HTTPS and Remote Desktop Protocol (RDP). Web servers using HTTPS are often target of scanners like SSL Labs. Such tools rank the security based on the ciphers available among other settings in the SSL implementation. Therefore it is of interest to know what the real value is of such ranking. Implementations of RDP have vulnerabilities related to weak cryptography. It is known that the decryption of HTTPS traffic is easily when in the possession of a private key, but it is not known whether cracking encrypted RDP traffic has the same feasibility. This research aims to evaluate the practical feasibility of cracking weak SSL/TLS ciphers in order to find out what the real risk is that the usage of weak ciphers expose.

1.1 Research question

The research question for this project is: *What is the feasibility of cracking weak SSL/TLS ciphers based on resources required?*

In order to answer this question the following sub-questions are set:

1. Which SSL/TLS ciphers are considered weak?
2. How can intercepted traffic be decoded when the key is known, and which tools can be used to achieve that goal?
3. What are the requirements for decryption?
4. How can the attack be classified based on time, money, and skills?

1.2 Related work

Many studies concern recommendations for cryptographic algorithms and key sizes. A bit outdated but still relevant is a study by Blaze et al. [2]. The subject concerns minimal key lengths for symmetric ciphers that are adequate enough to use in commercial enterprises or government intelligence agencies. The study describes that brute-force attacks can be performed fast and cheap against the key lengths that were common back in 1995. Those 40-bit and 56-bit keys were already considered weak. The time and money required to brute-force these keys is explained. The report advises the use of at least 90 bits keys, which is also considered weak nowadays.

Similar research is performed by Lenstra [3] and Orman et al. [4] in 2004. Both also cover asymmetric methods for key exchange, which influences the strength of the cryptography. Symmetric/asymmetric size-equivalence relations are provided. More recent work is done by ECRYPT II (European Network of Excellence in Cryptology II) [5] and provides an extensive report on algorithms and key lengths. All of these recommendations are summarised on keylength.com, a calculator for secure key lengths.

Research that resulted in commercially available hardware for cracking DES is performed by Kumar et al. [6]. The machine named COPACOBANA (Cost-Optimized Parallel Code Breaker) hosts 120 low-cost Field Programmable Gate Arrays (FPGAs). It is optimized for running crypt-analytical algorithms and can be realised for less than $10,000. The system can perform an exhaustive key search for DES in less than 9 days on average. Also, other efforts for cracking DES are described. These are able to crack DES in even less time but are more expensive.

This research project will expand on the previous research by investigating the practical feasibility of cracking encrypted SSL/TLS session data.

1.3 Organisation

This report is organised as follows. In Chapter 2 some background information on TLS and RDP is provided. Chapter 3 describes our approach, and Chapter 4 presents our results. Finally, we conclude and describe possibilities for future work in Chapter 5.

Chapter 2

Background

In this chapter, a theoretical background is provided. Two aspects of TLS are highlighted because of their importance for this research, namely key exchange and encryption algorithms.

The terms SSL and TLS are often used interchangeable. Because TLS 1.2 is the most recent version specified in RFC 5246 [7], we will further refer to it as TLS. The protocol uses both asymmetric and symmetric cryptography. Authentication of the counterparty is performed by means of X.509 certificates. This asymmetric key is used for the exchange of a symmetric key. That is the session key used for the encryption of the data sent. The TLS Handshake Protocol, one of the two layers of which TLS is composed of, is responsible for this authentication and key exchange at the start of a secure session.

TLS supports an extensive amount of algorithms for authentication, key exchange, encryption, and integrity. One combination of algorithms is called a cipher suite. It is negotiated between client and server within the TLS Handshake. The client specifies a list with supported cipher suites in the ClientHello message in order of preference. Then the server replies with the cipher suite that it has selected in the ServerHello message. Each cipher suite defines a key exchange algorithm, a bulk encryption algorithm, a Message Authentication Code (MAC) algorithm, and a pseudorandom function (PRF). The first two are further explained in the next sections.

2.1 Key exchange algorithms

Different algorithms exist for exchanging or generating the unique session key between the two sides of the communication channel. These algorithms are important if one wants to decrypt a session. Two of the most common algorithms are RSA and Ephemeral Diffie-Hellman (DHE in the cipher suite name).

Part of the RSA key exchange mechanism is the generation of a pre-master secret by the client. This is sent to the server encrypted with the servers' public key. Then the pre-master secret is used for computing the 48-byte master secret on both ends. Finally the master secret is used for generating the actual session keys. Access to the TLS handshake and the pre-master secret allows an attacker to compute the master secret. RSA encrypts the master secret with the private key in order to protect it.

Diffie-Hellman uses key generation with the use of public and private components. Client and server both compute the master secret. Only the public components are sent over the network, but interception does not provide enough information to compute the master secret or the session keys. Ephemeral suites use session keys that do not include the servers' private key in the computation. Therefore, compromise of the private key does not result in the compromise of all previously intercepted sessions. This characteristic provides perfect forward secrecy [8].

2.2 Bulk encryption algorithms

The bulk encryption algorithm specifies the cipher used to encrypt the message stream. Some of the most common ciphers used in TLS[1] applications are displayed in Table 2.1. DES (Data Encryption Standard) is also included, but is no longer supported since TLS 1.2. Each cipher has a specific key length, which is important for its strength.

Cipher	**Key length (bits)**
DES	46
3DES	168
RC4	40 - 128
AES	128 - 256

Table 2.1: The key space of common ciphers

Another "cipher" that can be negotiated is NULL encryption, which means that no encryption is applied, and therefore data can be obtained in plain text.

2.3 RDP encryption

Because TLS is application protocol independent, other protocols can built on top of it. Among those protocols is RDP, which is specified by Microsoft as an open protocol [9]. RDP provides two levels of security, called enhanced and standard security. TLS 1.0, 1.1, 1.2, and Credential Security Support Provider (CredSSP) are used for the *enhanced security* level. Older versions of Windows use the *standard security* level by default. This security level does not verify the identity of the server.

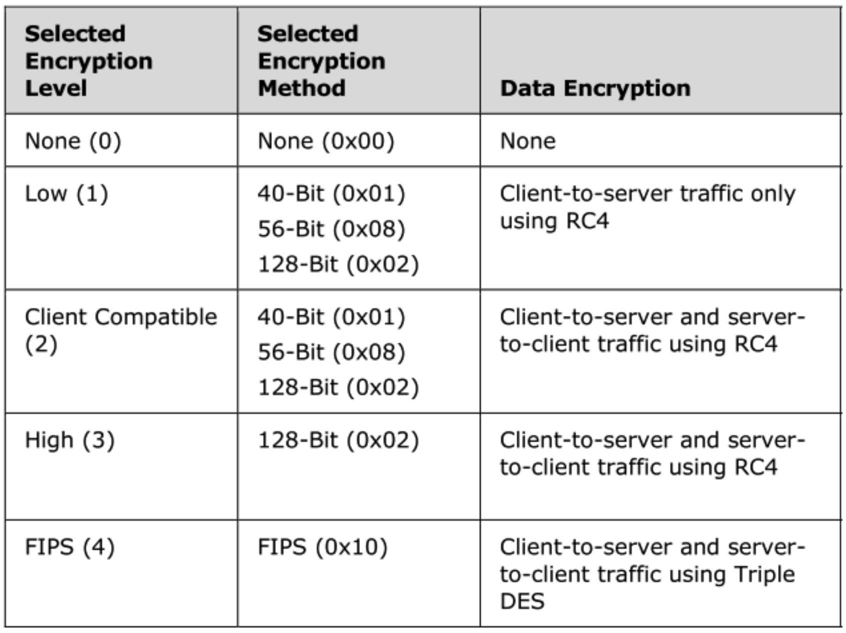
A simplified view of the RDP network stack is shown in Table 2.2. An RDP session starts with the establishment of a TLS channel. For enhanced security the server is authenticated. CredSSP is used for authentication of the user that connects to the remote desktop computer. CredSSP uses Kerberos or NT LAN Manager (NTLM) for that case. On the right side, the protocol stack for RDP traffic is shown, which uses several protocols for transport and communication, such as the transport service named Transport Packet (TPKT).

[1]Information based on ClientHello packets from several browsers

Kerberos / NTLM	RDP
CredSSP	Transport & Communication
TLS	
TCP	

Table 2.2: Network stack of RDP authentication and data

Standard security supports five different encryption levels as displayed in Figure 2.1. Enhanced security supports a subset of the encryption levels, namely Client Compatible, High, and FIPS (Federal Information Processing Standard). Except for FIPS, which only allows 3DES, the encryption algorithm is negotiated between client and server.



Selected Encryption Level	Selected Encryption Method	Data Encryption
None (0)	None (0x00)	None
Low (1)	40-Bit (0x01) 56-Bit (0x08) 128-Bit (0x02)	Client-to-server traffic only using RC4
Client Compatible (2)	40-Bit (0x01) 56-Bit (0x08) 128-Bit (0x02)	Client-to-server and server-to-client traffic using RC4
High (3)	128-Bit (0x02)	Client-to-server and server-to-client traffic using RC4
FIPS (4)	FIPS (0x10)	Client-to-server and server-to-client traffic using Triple DES

Figure 2.1: Standard RDP Security [9]

Chapter 3

Methodology

This chapter describes the approach of this research. In order to answer the research questions the following steps are performed:

1. Identification of weak cryptography;
2. Decoding of encrypted traffic;
3. Identification of attack requirements;
4. Classification of attack feasibility.

3.1 Identification of weak cryptography

In order to identify the cipher suites that are considered weak, security and governmental organisations are consulted. These organisations have recommendations about the ciphers that should not be used according to the security level an organisation has to meet. Both PCI DSS (Payment Card Industry Data Security Standard) [10] and ISO/IEC 27001 recommend the use of strong cryptography but do not specify key lengths or algorithms[1]. Therefore the definition of weak ciphers is mainly based on the more specific and practical SSL Labs and OWASP, which describe best practises.

3.2 Decoding of encrypted traffic

The practical feasibility of decryption is experimentally verified. For generating network traffic a test lab with multiple virtual servers was created. The virtual machines were running Ubuntu 12.10, Windows Server 2003, 2008, and 2012. Using the tools as described by Vandeven [8], several setups are investigated in order to identify requirements for traffic decoding. Our starting point is traffic that is already captured. In that case no interference with the handshake is possible anymore. As a consequence there is no possibility to downgrade the security level, SSL version or cipher suite.

[1]The versions consulted are PCI DSS version 3.0 (November 2013), ISO/IEC 27001:2013(E), and ISO/IEC 27002:2013(E)

3.2.1 Experimental setup

We first verified the feasibility of decrypting HTTPS traffic. The Ubuntu server running a standard Apache configuration with `mod_ssl` was used for this. For the decryption of RDP traffic, remote desktop was enabled on the Windows Server 2012 machine. The experiments conducted are displayed in Table 3.1.

System	Traffic type
Ubuntu	HTTPS
Windows Server 2012	RDP - Enhanced Security (TLS)
Windows Server 2012	RDP - Standard Security (Low)
Windows Server 2012	RDP - Standard Security (High)

Table 3.1: Decryption experiments

The web page was requested using the OpenSSL `s_client`, because that allowed for full control over the cipher suite and it has verbose output regarding the session. To ensure that standard or enhanced RDP security was used, it was explicitly configured to be enforced on both server and client. On Windows Server, those settings are configured in the group policy editor. The Server 2003 and 2008 instances were used for analysis of their default settings.

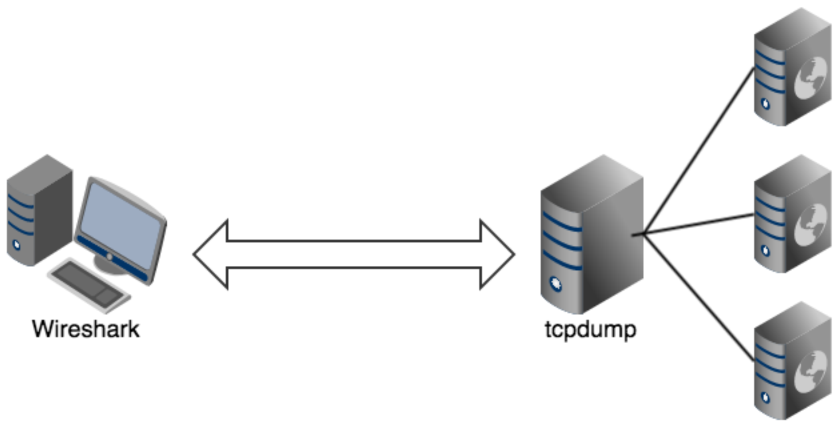


Figure 3.1: Network setup

As shown in Figure 3.1, the machine in the middle of the network topology acted as sniffer. It listened on the bridge interface that connected the servers with the rest of the network. For this part of the research tcpdump was used, but alternatives as Wireshark, tshark or ssldump can be used as well. Traffic was captured with tcpdump as follows:

```
# tcpdump -w traffic.pcap
```

3.2.2 Decryption of traffic

Wireshark, an open-source packet analyser, was used for the analysis and decryption of the captured traffic. The newest version (1.10.7) during the time of research was used as well as an older version (1.8.2). Decryption results differed considerably. The newer version was better in dissecting RDP packets. Some of the decrypted packets could be read only with this version.

Known private keys are used in this research project in order to analyse the decryption process. For Apache this key can be copied from the file system. Exporting the private key of a Windows server cannot be done directly from the operating system. For Server 2003 it can be exported via the management interface, but for newer versions that is not allowed. A tool for invoking Windows security functions, Mimikatz[2], could be used for this purpose. The procedure followed is described on the FreeRDP wiki [11].

3.3 Identification of attack requirements

A division is made between Wireshark-specific requirements and requirements to the traffic and session.

3.3.1 Wireshark-specific requirements

The decryption of TLS traffic is explained by Wireshark [12]. Based on this information requirements for decryption are identified:

- Wireshark must be compiled against GnuTLS (for Linux systems);
- The RSA key file must be in PEM format or be a PKCS#12 keystore with password provided;
- The RSA key file must only contain the private key with the correct header and footer;
- The format of the master secret must be in the correct format. For RSA key exchanges a Session-ID is required.

3.3.2 Session requirements

From the background information about TLS it follows that the requirements to the TLS session are:

- Knowledge of the private key of the server and a handshake that does not use Diffie-Hellman or the master secret used to encrypt the session data;
- No ephemeral suite is used;
- The session is not reused.

[2]https://github.com/gentilkiwi/mimikatz

3.4 Classification of attack feasibility

The method for classifying the cost of brute-force attacks as described by Blaze et al. [2] has been used for the classification of symmetric and factorisation based systems. The type of attackers that they have defined are explained in Section 3.4.1. The approach from Lenstra [3] for calculating the cost of an attack has been used in our research. This is described in Section 3.4.2. The classification for crypto-analytical attacks is based on specific literature concerning those attacks.

3.4.1 Type of attackers

Blaze et al. [2] describe different type of attackers plus the budget they have at their disposal, together with the equipment they use. These categories have been used in this research project:

- Hacker: $400;
- Small organisation: $10k;
- Medium organisation: $300k;
- Large organisation: $10M;
- Intelligence agency: $300M.

These numbers are used in the calculations that are described in the next section.

3.4.2 Calculation

Developments in hardware and cryptanalysis result in a decrease of the attack effort over a certain amount of time. Because studies were used that describe the attack effort in previous years, we have to take this into account. The figures from these studies have to be adjusted for new developments. Besides that, the attack effort has to be calculated for the different budgets as described in the previous section.

In order to have up-to-date figures for the developments in technology Moore's law is used. This law says that computing power doubles every 18 month. Lenstra [3] applied Moore's law as follows: "The cost of any fixed attack effort drops by a factor 2 every 18 months". Integer factorisation poses a problem that differs from exhaustive key search. Not only the cost of factoring drops by a factor 2 in 18 months, but the cost of equipment also drops with the same rate. Therefore, the Double Moore factoring law is applied on these calculations, which Lenstra defined as: "The cost of factoring any fixed modulus drops by a factor 2 every 9 months".

Lenstra [3] uses the same formula for the cost of breaking symmetric and factorisation based systems. With his method the time of a key search can be calculated for different budgets. This same method is also used in the estimations by Gehrmann et al. [5]. Lenstra describes the cost of an attack as being c dollars, to be realised in d days. For any reasonable w, the attack can be realised in d/w days by a device costing cw dollars. This implies that the cost-performance ratio is constant, i.e. with a larger budget available, the number of days needed to break a key decreases with the same factor. The cost of an attack can then be specified in *dollardays*: the amount of dollars it costs to break a key in one day.

The calculations are automated in a Python script. Below is a part of the script. The full version can be found in Appendix A. The results obtained with the help of these formulae can be found in Section 4.3.

```
x = [400, 10000, 300000, 10000000, 300000000]

# dollardays (getting the cost for cracking in one day):
# d/w = 1, thus d = w, thus c*w = c*d)
cd = c * d

# Apply Moore's law
a = (y2 - y1) * 12
c = cd / (2 ** (a/18))
d = 1

# b = cw, thus w = b / c
i = 0
for b in x:
        w = b / c
        t = d / w
...
```

Chapter 4

Results

This chapter describes the results of this research. First, weak cryptography as described by security organisations is identified. Then, the practical feasibility of decoding encrypted traffic is verified. Finally, a classification is created for the feasibility of cracking the weak ciphers.

4.1 Identification of weak cryptography

The usage of weak cryptography in applications invites an attacker to conduct attacks against it. The following categories of weak cryptography have been identified:

1. Cipher suites with key sizes smaller than 128 bits;
2. Ciphers with cryptographic weaknesses;
3. RSA keys with short length.

4.1.1 Cipher suites with key sizes smaller than 128 bits

Cipher suites define the size of the symmetric session key. SSL Labs describes that 128 bits is the recommended minimum key length today to provide enough protection against brute force attacks [13]. As shown in Table 2.1, DES and the RC4 types that use 40 or 56 bits keys have a key size that is smaller than 128 bits. That results in an attack effort that is equal to the key space of the ciphers as can be seen in Table 4.1. 3DES provides only 108 or 112 bits of security and therefore is below the recommend minimum. This is due to the two keys that are being used for encryption. Also Export level (EXP/EXPORT in cipher suite name) cipher suites belong to this category. Those cipher suites have a restricted length of 40 bits, designed for US export regulations.

Cipher	Attack effort
DES	2ˆ40
3DES	2ˆ112
RC4	2ˆ40 - 2ˆ128
AES	2ˆ128 - 2ˆ256

Table 4.1: Attack effort

4.1.2 Ciphers with cryptographic weaknesses

RC4 was first recommended for mitigation of the BEAST attack. Now, this advice has been withdrawn because of the discovery of crypto-analytical attacks [13]. Parts of the plain text can be recovered because of statistical biases in the RC4 key table. However, $13 * 2^{20}$ encrypted sessions are needed to perform the most recent attack, a double-byte bias attack. It is expected that the attacks on RC4 will improve in the future [14].

4.1.3 RSA keys with short length

Public RSA keys that are used for the authentication of the server should be at least 2048 bits [15]. RSA keys with a smaller size are considered weak because they are vulnerable for integer factorisation. This attack can recover the private key by factorisation of the modulus N, which is part of the public key. The largest key size that is publicly known to be factorised is RSA-768 [16].

4.2 Decoding of encrypted traffic

This section describes the process of decoding encrypted traffic. First, methods for obtaining the private or session key are described. Then the results of the decryption expirements are analysed.

4.2.1 Obtaining the key

Before traffic can be decrypted, the session or private key has to be obtained first. Several methods are identified for achieving that. Based on Section 4.1 the following methods can be identified:

- Brute force (exhaustive key search);
- RSA factorisation;
- Crypto-analytical attacks.

Exhaustive key search or brute force is a crypto-analytical attack that searches the key space of a cipher in order to find the correct key of encrypted data. Usually special purpose hardware, such as FPGA designs, is used for optimising this process in terms of speed.

For RSA factorisation, several tools are available that can run on desktop computers. Sage[1], a mathematical software package based on Python, can be used for small numbers. Msieve[2] is a C library with a suite of algorithms for factoring large integers. Lastly we note that crypto-analytical attacks exist in different variations. Not all of them are usable against weak ciphers, or result in decryption of the whole stream of traffic. For instance the attacks may result in recovering a web cookie only. Therefore, not every attack is covered in this report. The feasibility of the attacks identified is considered in Section 4.3.

4.2.2 Experimental results

During this research project traffic was intercepted and decrypted successful with Wireshark, except for the HTTPS session with its session key. The results are displayed in Table 4.2. These results are further explained below.

System	**Traffic type**	**Decoding successful with**	
		Private key	Master secret
Ubuntu	HTTPS	yes	no
Server 2012	TLS	yes	yes
Server 2012	RDP - Low	yes	yes
Server 2012	RDP - High	yes	yes

Table 4.2: Results of decryption experiments

4.2.3 Decryption of HTTPS traffic

The packet capture can be analysed to answer the question whether the session meets the requirements as described in 3.3.2. The first step is identifying the used cipher suite in order to look for possible weak ciphers. The ServerHello packet can be used to identify the negotiated cipher suite. This information is included in the Cipher Suite field of this packet. When the key exchange method is RSA, the attacker needs to be in the possession of either the private key or the session key in order to decrypt the traffic. For a session with Diffie-Hellman key exchange, it is only possible to decrypt using the symmetric session key.

```
TLSv1.1 Record Layer: Handshake Protocol: Server Hello
        Content Type: Handshake (22)
        Version: TLS 1.1 (0x0302)
        Length: 58
        Handshake Protocol: Server Hello
            Handshake Type: Server Hello (2)
            Length: 54
            Version: TLS 1.1 (0x0302)
            Random
            Session ID Length: 0
            Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a) ...
```

[1]www.sagemath.org
[2]sourceforge.net/projects/msieve/

4.2.3.1 Using private key

After the private key has been retrieved, the process for decrypting HTTPS traffic is straight forward. All settings regarding TLS traffic can be found in the Wireshark preference panel under Protocols, SSL. The information that needs to be provided under the RSA keys list is the IP address of the server, the port number, the protocol, and the private key. For HTTPS, the following parameters are specified: `<server-ip>,443,http,<path to key>`. This results in plain HTTP data, which can be shown with Follow SSL stream (a related packet must be selected).

4.2.3.2 Using session key

It turned out that the Export SSL Session Keys option that is available in Wireshark, does not export the session key, but the master secret. The master secret is not the same as the actual encryption keys. This is described in section 6.3 Key Calculation of RFC 5246 [7]. A consequence for the brute-force attack is that the session key cannot be imported directly into Wireshark. Further research has to be performed for solving this problem. Decryption with the master secret will be described instead.

The master secret for RSA key exchanges is configured in the preference panel for the SSL dissector in the following format[3]:

```
 *   - "RSA Session-ID:xxxx Master-Key:yyyy"
     *     Where xxxx is the SSL session ID (hex-encoded)
     *     Where yyyy is the cleartext master secret (hex-encoded)
```

For using the master secret with an RSA cipher suite, also the SSL Session ID is needed. This value can be found in the same ServerHello packet. When it is absent, the value of Session ID Length is equal to zero as is the case for the packet shown. That resulted in traffic that could not be decrypted. Unfortunately Wireshark needs the Session ID for mapping the traffic flow to the private key. Decryption of the traffic should be possible without providing the Session ID. One option is to use the format for non-RSA traffic (shown below). Another option is to use a different tool such as Unsniff[4] that can decrypt the traffic with the master secret only.

For Diffie-Hellman key exchanges Wireshark requires the master secret to be in a file with the client random, which can be copied from the ClientHello packet. The master secret for Diffie-Hellman needs to be provided in the following format:

```
*   - "CLIENT_RANDOM xxxx yyyy"
     *     Where xxxx is the client_random from the ClientHello (hex-encoded)
     *     Where yyy is the cleartext master secret (hex-encoded)
```

[3]Code snippets from the Wireshark source code (version 1.10.8)
[4]www.unleashnetworks.com/blog/?p=574

4.2.4 Decryption of RDP traffic

Decryption of RDP traffic does in general not differ much from HTTPS traffic. Therefore, only the differences are highlighted in this section. When viewing the encrypted RDP traffic no human-readable information can be derived. This is in contrast with HTTPS traffic where TLS Handshake packets are dissected by Wireshark. For the *standard security* level the packets are recognised as TCP streams. This may differ per Windows Server version as a capture for Server 2008 was interpreted differently by Wireshark. A couple of SSL packets were present in that capture, but without any meaning. For the *enhanced security* level also TPKT packets are dissected.

The downside of this is that the cipher suite used cannot be identified. However, as described in Section 2.3, there are only a couple of possibilities for the cipher suite. A common approach for identifying the cipher suites supported by the server is the usage of an SSL scanner. SSLMap[5] was used against the Windows Server 2012 machine. The results are included in Appendix B. Log entries for Schannel, a Windows Security Support Provider for encryption, were consulted to verify that the cipher suite shown in Wireshark was correct. This was indeed the case.

In order to decrypt RDP traffic with the private key, one should configure the SSL dissector with the following setting [17]: `<server-ip>,3389,tpkt,<path to key>`. Decryption with the master secret is exactly the same as for HTTPS traffic. Decryption for RDP is successful when CredSSP packets are shown. Then, user name and password can be viewed in plain text. However, the remainder of the traffic is compressed. As stated in a Microsoft article about RDP encryption [18], compression needs to be disabled in order to view the unencrypted traffic.

```
Remote Desktop Protocol
    SendData
        clientInfoPDU
            flags: 0x0040
            flagsHi: 0x0000
            codePage: 1043
            optionFlags: 0x002103fb
            cbDomain: 0
            cbUserName: 26
            cbPassword: 24
            cbAlternateShell: 0
            cbWorkingDir: 0
            domain: 0000 ()
            userName: 410064006d0069006e006900730074007200... (Administrator)
            password: 70006100730073077006f00720064000000... (password)
```

The decrypted ClientInfoPDU with username and password.

[5]thesprawl.org/projects/sslmap/

4.3 Classification of attack feasibility

First the cost for breaking cryptographic systems is estimated. Then the feasibility of other crypto-analytical attacks is described. Our results are discussed in the end of this section.

Many estimations have been done for the costs of breaking symmetric and factorisation based keys. Those figures are renewed by applying (double) Moore's law. The recovery time for different key lenghts is calculated for each type of attacker. The estimations do not include the one time costs for design.

4.3.1 Symmetric systems

Two key lengths are chosen for symmetric systems: 56-bit and 83-bit. In 2008, ECRYPT described that a 56-bit key can be cracked in one month for only 750 dollar. For 83-bit already a machine costing $300M is needed. This machine can find the key in 73 days. A machine that can perform a 128-bit key search in one month, would still cost $2.8 * 10ˆ24 [5]. This is even infeasible for an intelligence agency with the budget as used in the calculations.

The previous developments are translated to this year, which is displayed in Table 4.3. Then cracking 56-bit session keys is feasible for the normal hacker, because it can be performed in 3.5 days on average. But cracking a symmetric system with 83-bit session keys becomes infeasible for the hacker. For an intelligence agency the attack is still feasible in around 4.6 days.

Attacker	**Budget**	**Recovery time**	
		56bits	83bits
Hacker	$400	3.52 days	3421875 days
Small org.	$10k	202.5 minutes	136875 days
Medium org.	$300k	6.75 minutes	4562.5 days
Large org.	$10M	0.20 minutes	136.88 days
Intelligence agency	$300M	0.01 minutes	4.56 days

Table 4.3: Time and cost for breaking symmetric keys in 2014

That means that for 3DES with a key space around 110 bits, exhaustive key search is still impractical for an attacker. Only DES and RC4 (40 or 56 bits) are feasible, while cracking RC4 (128 bit) and AES are far beyond his capabilities.

4.3.2 Factorisation based systems

Relatively recent is the factorisation of RSA-512 keys used for signing the operating systems of Texas Instrument calculators. One of the keys is factored in 73 days using a desktop computer (2009). That resulted in several successful distributed attempts of other signing keys [19, 20]. The first successful attempt of the factorisation of a 768-bit RSA key was announced in 2010 by Bos et al. [16].

Preneel [21] estimated the budget that would be needed for the factorisation of RSA-768 keys. A 768-bit modulus could be factored in 95 days with a budget of 10,000 dollar.

The time and cost for RSA factorisation in 2014 is shown in Table 4.4. 512-bit RSA keys can be cracked even by a hacker. 768-bit factorisation would take the hacker about two months, but is not impossible.

Attacker	**Budget**	**Recovery time**	
		512bits	768bits
Hacker	$400	1034.71 minutes	58.91 days
Small org.	$100k	41.39 minutes	2.36 days
Medium org.	$300k	1.38 minutes	113.10 minutes
Large org.	$10M	0.04 minutes	3.39 minutes
Intelligence agency	$300M	0.001 minutes	0.11 minutes

Table 4.4: Time and cost for RSA factorisation in 2014

1024-bit RSA is not factorised yet. Several estimates has been done over the years as described by Coffey [20]. One of the most recent estimates dates from 2009 by Bos et al. [22]. Although these estimates are hypothetical, it can be assumed that 1024-bit factorisation is feasible for an intelligence agency. In 2012 Bernstein et al. have shown that it is also feasible for large organisations [23].

4.3.3 Crypto-analytical attacks

Attacks against cryptographic weaknesses have a different feasibility. For evaluating this, we consult the paper of Bernstein et al. [14] about the security of RC4. The setup that is used only involves a recent consumer-grade computer. The attacks proposed apply on both TLS and WPA. They describe that $13 * 2^{20}$ amount of sessions are needed. And even then, only parts of the plain text can be recovered. Bernstein et al. come to the conclusion that RC4 does not provide as much security as it should, but the attacks they describe are not considered as a practical threat. Still, they point out that the attacks described could be improved in the future. Therefore, they recommended to avoid the usage of RC4 in TLS.

4.3.4 Discussion

Because the calculations make use of models that easily can turn out to be too simplified, the results obtained in this way will now be more critically discussed. As expected, the calculations correspond with the recommendations for key lengths described at the beginning of this chapter. The ciphers that are recommended are those for which it is computationally infeasible to crack. Even the removal of 3DES in the near future makes sense with these results. The estimations for factorisation based systems are more optimistic than for symmetric ones. This is because the double Moore's law has been used. When looking at developments on RSA-1024 factorisation, they are not as fast as this law implies. Also, the figures that are less than one minute could be too optimistic.

Finally, the practical feasibility is concluded. This feasibility applies for someone with low resources like the hacker with a budget around 400 dollar. With such capabilities, the practical feasibility can be classified as displayed in Table 4.5.

Weakness	**Practical feasible**
Cipher with short key length (e.g. DES and RC4)	yes*
3DES, AES	no
Weakness in RC4	no
RSA modulus 512-bit and below	yes
* When decryption with session key is possible	

Table 4.5: Practical feasibility

It should be noted that private keys can be used for sessions with RSA based key exchanges only.

Chapter 5

Conclusions

5.1 Conclusions

The central research question reads: *What is the feasibility of cracking weak SSL/TLS ciphers based on resources required?* Four sub-questions are set in order to answer this question.

1. Which SSL/TLS ciphers are considered weak?

The ciphers most commonly called weak are 3DES and RC4. Weaknesses of a cipher can be divided into ciphers that use short key lengths and ciphers with cryptographic weaknesses. DES and 3DES fall into the first category. RC4 is considered weak because of statistical biases in the key table that can help in recovering parts of the plain text. A group of cipher suites that are weak are EXPORT ciphers. They are weakened on purpose, and have short key lengths. The NULL cipher can also be seen as weak because it does not encrypt data at all. RSA public keys have weaknesses such as short key lengths. Moduli that have a length that is less than 2048-bit are considered weak because of integer factorisation.

2. How can intercepted traffic be decoded when the key is known, and which tools can be used to achieve that goal?

All these weaknesses can aim in cracking TLS traffic. The ciphers with short key lengths are susceptible to exhaustive key search, while the ciphers with cryptographic weaknesses are vulnerable for other cryptanalytic attacks. Short RSA keys can be factorised. When one of these methods can be applied for retrieving the private key, they can be used for the decryption of traffic. For the two investigated protocols, HTTPS and RDP, this can be performed with Wireshark. The tool is able to decrypt and dissect this traffic. For RDP, only parts of the decrypted traffic resulted in useful information.

3. What are the requirements for decryption?

The requirements for decryption can be divided into requirements for the intercepted traffic and Wireshark-specific requirements. Decryption with the private key is only possible for a session with an RSA based key exchange. Furthermore, the whole session needs to be captured for decryption. In order to decrypt traffic with the session key, Wireshark requires a special formatted key that includes the master secret and Session ID. When the ID is not present in the session, decryption will likely fail with Wireshark.

4. How can the attack be classified based on time, money, and skills?

The feasibility of cracking the ciphers with short key length and factorisation depends on the budget of the attacker. Cracking 56-bit session keys is feasible for the normal hacker with a budget around 400 dollar, because it can be performed in 3.5 days on average. Cracking a symmetric system with 83-bit session keys already becomes infeasible for the hacker. For an intelligence agency with an estimated budget around 300M dollar, that would be still feasible in around 4.6 days. That means that for 3DES with a key space around 110 bits, exhaustive key search is still impractical for an attacker. Only DES and RC4 (40 or 56 bits) are feasible, while cracking RC4 (128 bits) and AES are far beyond his capabilities. Only small sizes can be factorised by someone with low resources. However, such small sizes are rarely used these days. 512-bit factorisation is possible for a hacker in a couple of days, and a 768-bit key would take two months. 1024-bit RSA keys can be cracked by a large organisation or intelligence agency.

5.2 Future research

More research is needed in the light of the following areas:

- *Decryption with session key instead of master secret.* Decryption was (partially) successful with the master secret. However, brute force attacks result in the session key. More effort has to be done for solving this problem and the investigation of methods or tools that can perform decryption with the session key.
- *Decompression of RDP traffic.* Compression was applied on the RDP traffic. Therefore, no information could get extracted from the traffic. More analysis is desired for this.
- *Elliptic curve cryptography.* With elliptic curve cryptography different key sizes are applicable. The key sizes used in this research have to be converted for this type of cipher suites.
- *Other applications that use TLS.* Many other applications use TLS for secure communication. The impact of decrypting traffic from these applications still has to be investigated.

Appendix A

Attack effort calculation

This appendix contains a Python script that was written for the calculations of attack efforts for breaking symmetric and asymmetric systems.

```
#!/usr/bin/python
# Output is LaTeX formatted
from __future__ import division

c =
d =
y1 =
y2 = 2014
x = [400, 10000, 300000, 10000000, 300000000]
attacker = ["Hacker", "Small org.", "Medium org.", "Large org.",
    "Intelligence agency"]
m = 18 # 18 for Moore’s law, 9 for double Moore factoring law

# dollardays (getting the cost for cracking in one day: d/w = 1, thus d = w,
    thus c*w = c*d)
cd = c * d

# Apply Moore’s law
if y1 != y2:
        a = (y2 - y1) * 12
        c = cd / (2 ** (a/m))
        d = 1

print "Attacker & Budget & Recovery time \\\\"

# b = cw, thus w = b / c
i = 0
for b in x:
        w = b / c
        t = d / w
        # if less than one day, translate to minutes
        if t < 1:
```

```
        time = str(t * (24 * 60)) + " minutes"
else:
        time = str(t) + " days"
print attacker[i] + " & \$"  + str(b) + " & " + time + " \\\\"
i = i + 1
```

Appendix B

RDP cipher suites

This appendix contains the cipher suits that are supported by the Windows Server 2012 machine retrieved with SSLMap.

```
[*] Scanning 145.100.108.164:3389 for 229 known cipher suites.
[+] TLS_RSA_WITH_AES_128_CBC_SHA (0x00002F)
[+] TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0x00C013)
[+] TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x00C014)
[+] TLS_RSA_WITH_AES_256_CBC_SHA (0x000035)
[+] TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000A)
[+] TLS_RSA_WITH_RC4_128_MD5 (0x000004)
[+] TLS_RSA_WITH_RC4_128_SHA (0x000005)

==================== Scan Results ====================
The following cipher suites were rated as HIGH:
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA

The following cipher suites were rated as MEDIUM:
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
```

Bibliography

[1] SSL Labs. Ssl pulse, 2014. URL `https://www.trustworthyinternet.org/ssl-pulse/`.

[2] M Blaze, W Diffie, RL Rivest, and Bruce Schneier. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security. Technical report, 1996.

[3] Arjen K Lenstra. Key lengths. *Handbook of Information Security*, 2:617–635, 2004.

[4] H Orman and P Hoffman. Determining Strengths For Public Keys Used For Exchanging Symmetric Keys. 2004. URL `http://tools.ietf.org/html/rfc3766.html`.

[5] C Gehrmann and M Naslund. Yearly Report on Algorithms and Keysizes. Technical report, ECRYPT II, 2012.

[6] Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking ciphers with copacobana–a cost-optimized parallel code breaker. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 101–118. Springer, 2006.

[7] T Dierks and E Rescorla. Rfc 5246: The transport layer security (tls) protocol. *The Internet Engineering Task Force*, 2008.

[8] Sally Vandeven. SSL/TLS: What's Under the Hood. Technical report, 2013.

[9] Microsoft. Remote Desktop Protocol : Basic Connectivity and Graphics Remoting. Technical report, 2014.

[10] PCI DSS. Pci security standards council, 2013. URL `http://www.pcisecuritystandards.org`.

[11] Volodymyr Lisivka. Mimikatz, 2013. URL `https://github.com/FreeRDP/FreeRDP/wiki/Mimikatz`.

[12] Wireshark. Ssl, 2014. URL `http://wiki.wireshark.org/SSL`.

[13] Ivan Risti. SSL / TLS Deployment Best Practices. 3(September):1–11, 2013.

[14] Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob C N Schuldt. On the Security of RC4 in TLS and WPA . pages 1–31, 2013.

[15] OWASP. Transport layer protection cheat sheet, 2014. URL `https://owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet`.

[16] Joppe W Bos, Pierrick Gaudry, Alexander Kruppa, Peter L Montgomery, Dag Arne Osvik, Herman Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. pages 1–22, 2010.

[17] Wireshark. Rdp, 2013. URL `http://wiki.wireshark.org/RDP`.

[18] Hongwei Sun. Encryption negotiation in rdp connection, 2011. URL `http://blogs.msdn.com/b/openspecification/archive/2011/12/08/encryption-negotiation-in-rdp-connection.aspx`.

[19] Luke O'Connor. Solo desktop factorization of an rsa-512 key, 2009. URL `http://lukenotricks.blogspot.nl/2009/08/solo-desktop-factorization-of-rsa-512.html`.

[20] Neil Coffey. Rsa key lengths, 2012. URL `http://www.javamex.com/tutorials/cryptography/rsa_key_length.shtml`.

[21] Bart Preneel. Cryptography for Network Security : Failures , Successes and Challenges. pages 36–54, 2010.

[22] JW Bos, ME Kaihara, and Thorsten Kleinjung. On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. *IACR Cryptology ePrint ...*, 57(1): 1–19, 2009. URL `http://lacal.epfl.ch/files/content/sites/lacal/files/papers/ecdl2.pdf`.

[23] Daniel J Bernstein. FactHacks: RSA factorization in the real world, 2012.