Data security in theory and practice
Possibile security holes

Keszthelyi András

Universitas Budensis, Faculty of Economics,
Institute for Organizing and Management
address: H-1081 Budapest, Népszínház u. 8., Hungary.
e-mail: Keszthelyi.Andras@kgk.uni-obuda.hu

There are no enterprises which would not use computers to fulfill their administrative tasks. Computers became part of everyday administration, or, better to say: they became part of everyday life. This is why our age is called 'information age'. Both the amount of digital data and our dependency from these data has been growing intensively so digital data is high-valued as resource. Data owners and/or managers must, or at least ought to, protect their data from stealing or tampering with. Luckily standard communication protocols and methods have been developed for this purpose, they only ought to be used. If not, that will result in a high level of risk. We, at Óbuda University, have been using a computer based system called Neptun for about ten years to manage the scholar records of the students. In this paper I show some possible motivations and technical solutions why and how one could gain unauthorized access to such a system.

Keywords: data security, database cracking, certificate spoofing, arp poisoning, man in the middle attack, MITM

EconLit subject descriptor: L860 - Information and Internet Services; Computer Software;
JEL code: L860 - Information and Internet Services; Computer Software

1 Data must be protected first of all

Data protection should cover two fields. One field is to protect the data against data loss or corruption. The other, and more problematic, field is the protection against unauthorized access. This paper is about the second field.

The problem of the protection against unauthorized access can also be divided into two main fields, of which the first is the protection of the stored data and the other is the protection of the data communication between two computers. We will discuss the latter problem.

Because of historical reasons all the (historical) network protocols are plain text ones, i.e. all of the data of the communication travels via the network as plain text, including user names and passwords as well (see fig. 1.). The http protocol our browser uses when surfing the internet is also such a protocol. [2] HTTP is unsecure and is subject to eavesdropping attacks, which can let attackers get the whole conent of the data exchange.

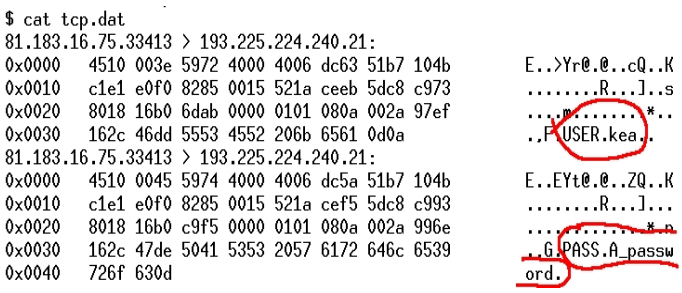

Figure 1.
Example for plain text protocol

Data encryption is nearly as old as the human communication itself. We know a large number of methods from the history to hide the plain text data. Or better to say: we know a large number of methods to *try* to hide it. Computers with their unbelievable computational power began a new era in both encryption and decryption.

1.1 Secure Communication via Untrusted Network

Mathematicians could provide different computer based methods for data encryption in private and in business life as well. These methods may even be 100% fathomless at least in a mathematical meaning. There are two main groups of these methods, one-key and two-key encryptions.

1.2 Method 1: One-key Encryption

The two communicators use the same key, in other words the same key is used both for encryption and decryption. The algorithm can be any simple and bijective operation which needs two bytes (plain text and key) to produce a third (ciphertext) one. Of course the operation should have an inverse one. E.g. an addition of character codes as bytes and key bytes modulo 256 will do.

The two main rules of such a method are the following: a) the key must be a series of real random numbers, b) the key should be kept in total secret.

For an early application of this kind of encryption (and, of course, decryption) see the well known novel 800 miles in the Amazons by Verne.

This method is very simple, easy to use, can guarantee a full 100% safety, but has one disadvantage: needs a secure channel for key exchange, which practically means that the two participants must personally meat somewhere. This cannot be a problem e.g. in the diplomatic corps, but business life demands other methods.

1.3 Method 2: Two-key Encryption

Mathematicians could and can provide us methods which do not need a secure channel for key exchange. This simplifies the use of data encryption both in private and business life.

The first and well known method of this kind is the RSA-algorithm which was published in 1978 by Ron Rivest, Adi Shamir and Leonard Adleman at MIT; the letters RSA are the initials of their family names, listed in the same order as on their paper. [3]

The first public key application based upon the RSA-algorithm was originally created by Philip Zimmermann in 1991. [4]

No need for a secure channel to distribute keys is a great advantage, but it has a price: these method are not 100% safe, theoretically they can be deciphered in certain conditions but it would need unreal amount of resources.

1.4 Some Basics

In public key cryptosystems everyone has two related complementary keys: a publicly revealed key, called public key and a secret (or private) key. Each of the keys unlocks the code that the other key makes. Knowing the public key does not help you find the corresponding secret key. The public key can be published and widely disseminated. The private key must be kept in total secret. Two-key, or public key cryptosystems provide privacy without the need for the same kind of secure channel that a conventional, one-key cryptosystem requires for key exchange.

Anyone can use a recipient's public key to encrypt a message to that person, and the recipient uses his or her own corresponding secret key to decrypt that message. No one but the recipient can decrypt it, because no one else has access to that secret key (at least according to rules;). Not even the person who encrypted the message can decrypt it.

Message authentication is also provided. The sender's own secret key can be used to encrypt a message, thereby signing it. This creates a digital signature of a message, which anybody can check by using the sender's public key. This process

proves that the sender was the true originator of the message, and that the message has not been altered by anyone else, because the sender alone possesses the secret key that made that signature. Forgery of a signed message is not possible, and the sender cannot later disavow his signature.

Public keys are kept in individual key certificates that include the key owner's name, a timestamp of when the key pair was generated, and the actual key material (and other possible fields).

1.5 Security Rules

No data security system is impenetrable. Public key cryptosystems can be circumvented in a variety of ways. Potential vulnerabilities including compromising of the secret key and public key tampering should be avoided. There are many other ways or by-pass roads, of course, to penetrate such a cryptosystem, e.g. deleted files which are still somewhere on the disk, viruses and Trojan horses, electromagnetic emissions, exposure on multi-user systems, or even doing a traffic analysis. Let us see how we can use public key cryptography according to Phil Zimmermann, developer of PGP. [7]

The first rule of security is to keep your secret key, according to its name, in secret. If someone gets your secret key, not only can they read your messages but they can make signatures in your name as well.

The second: When you use someone's public key, make certain it has not been tampered with. A new public key from someone else should be trusted if, and only if, you got it directly from its owner (this would mean you have a secure channel for key exchange), or if it has been signed by someone else you trust. Make sure no one else can tamper with your own public keys. Maintain uninterruptible physical control of both the public keys you collected and your secret key and keep a backup copy of them.

1.6 Web of Trust

Anybody can sign digitally someone else's public key as a a so called introducer. You collect signed public keys. "As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys." [7]

The above mentioned introducers can be enterprises as well. It is a good business opportunity to digitally sign as many public key as possible, while the enterprise

has an efficient way to distribute it's own authentic public key in all over the world. These enterprises are called certificate authorities or certification authorities (CAs). Some of them are worldwide known and many of them are local Cas.

What is the difference between public keys and certificates? A certificate is a digitally signed document to validate its owner's authorization and name. The document consists of a specially formatted block of data that contains the name of the certificate holder (which may be either a user or a system name), the holder's public key, the begining and end date of validity, as well as the digital signature of a certification authority who digitally signed the certificate. The certification authority attests that the sender's name is the one associated with the public key in the document.

1.7 Certificates and HTTPS

HTTPS connections are often used for payment transactions on the World Wide Web or for sensitive transactions in corporate information systems or even in a scholar information system. HTTPS stands for the term Hypertext Transfer Protocol Secure, which is a combination of the Hypertext Transfer Protocol with the SSL/TLS protocol to provide encryption and secure identification of the server.

The main idea of HTTPS is to create a secure channel over an insecure network for communication. This ensures reasonable protection from eavesdropping and man-in-the-middle attacks (see below), provided that the server certificate is verified and trusted. HTTPS is designed to withstand such attacks and is considered secure against such attacks (with the exception of older deprecated versions of SSL). While HTTP URLs begin with "http://" and use port 80 by default, HTTPS URLs begin with the string "https://" and use port 443 by default. The details of HTTPS protocol is described in RFC 2818. [8]

The trust inherent in HTTPS is based on major certificate authorities whose public keys come pre-installed in browser software to be used for signature checking (this is equivalent to saying "I trust certificate authority (e.g. VeriSign) to tell me who I should trust"). Therefore an HTTPS connection to a website can be trusted if (and only if) all of the following are true:

a) The website provides a valid certificate (an invalid certificate shows a warning pop-up window in most browsers), which means it was signed by a trusted authority;

b) The certificate correctly identifies the website (e.g. visiting https://www.uni-obuda.hu and receiving a certificate for "uni-obuda.hu" and not "uni-oduba.hu" or "uni-obuda.hu.com").

It is possible, of course, that the biggest CA signs public keys only for the bigger CAs, bigger ones for the smaller ones, the smaller ones for the local ones etc.

1.8 Man in the Middle

It is possible to eavesdrop such a should-be-secure connection in certain conditions, if the public key is tampered with, i.e. the owner of the public key used in the connection is not the same as it should be. Just as if an interpreter stood between the two parties, as the old joke illustrates it:

A Spanish speaking bandit held up a bank in Tucson. The sheriff and his deputy chased him. When they captured him, and the sheriff, who couldn't speak Spanish, asked the bandit, who couldn't speak English, where he'd hidden the money. "I will not tell it you", he replied in Spanish. The sheriff put a gun to the bandit's head and said to his bi-lingual deputy: "Tell him that if he doesn't tell us where the money is right now, I'll blow his brains out." Upon receiving the translation, the bandit became very animated. "I've hidden it under the oak tree", he answered in Spanish. The sheriff leaned forward. "Yeah? Well..?" The deputy translated: "He says he wants to die like a man."

Technically a man in the middle attack can be performed by somebody (be its name: Middle) who can redirect the data flow in the network between the two original persons (be the names the classical Alice and Bob). In such a case when Alice sends her public key (P_A) to Bob, , or Bob downloads it from Alice's homepage, Middle can capture and store for himself the authentic P_A key. Then Middle generates a pair of keys (let these be P_M and S_M as the public and secure key of Middle, respectively). Middle replaces the original P_A key with his own P_M key and sends it forward to Bob. Bob thinks the received P_M key to be P_A (but he is wrong, of course). He uses this fake key to encrypt his message to Alice. So the encrypted message can easily be decrypted by Middle and only by him. Middle decrypts the redirected message, reads it, alters it if he wants, then re-encrypts it with the original P_A public key of Alice and sends it forward to Alice.

Casting: Alice stands for the sheriff, Bob for the bandit and Middle for the deputy. None of Alice and Bob knows that Middle is in the middle. This is why the authenticity of the public keys or certificates must be verified very carefully.

1.9 Failed Public Key Authenticity

If somebody tries to browse to an https site, there are two possibilities. In the normal case the site sends its certificate to the client. If the browser knows the CA who signed the certificate, i.e. has its authentic public key, everything is right,

there cannot be anyone in the middle. If not, the browser tries to check the CA who signed the certificate of the given https site. There may be a whole chain of digital signatures of different level CAs. As it was discussed above, a new certificate (public key) from someone else should be trusted (if it hasn't got directly from its owner) if it has been signed by someone else you trust. If I can trust CA_1, then I can trust everybody who's certificate is signed by CA_1. If the certificate of CA_2 is signed by CA_1, it also can be considered as trustworthy and so on. This procedure is based on the public key of some top level CAs whose public keys are built in the browsers by their developers.

If the chain of the digital signatures cannot be followed to one of the built-in top level CAs, the verification is failed so the client must not trust the site he or she wanted to browse. In such a case web browsers usually open a window which says that the browser is unable to verify the identity of the website to be browsed as a trusted site.

2 The case of Óbuda University Neptun

2.1 The situation

We at Óbuda University have been using Neptun for about a decade as a scholar information system. The Neptun server can be contacted for teachers at the url https://neptun.uni-obuda.hu/oktato/login.aspx, for students at the url https://neptun.bmf.hu/hallgato/login.aspx. Browsers say that they cannot confirm that the connection is secure (See fig. 2.), because the issuer is unknown. Browsers cannot check the certificate of the Neptun server because it is issued by a GeoTrust Inc., and there is no chain of trust to any top-level certificate authorities, certificates of whom browsers have, i.e. there is no chain of signatures which would lead to any of the issuers of the builtin certificates.

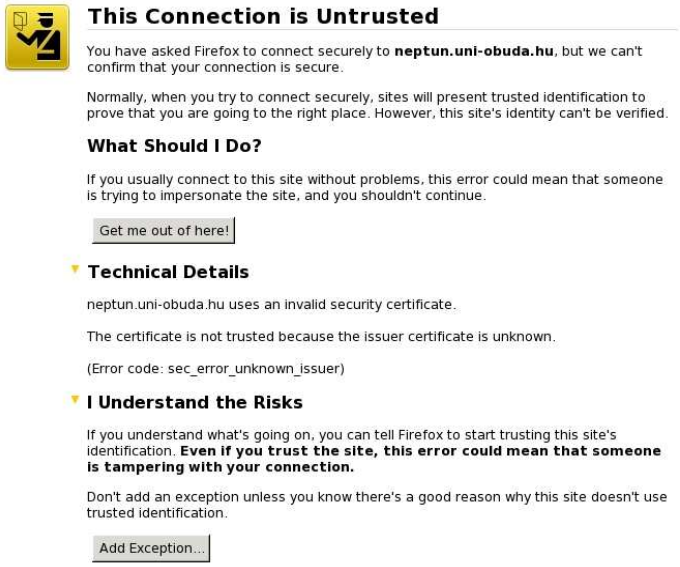

Figure 2.

Certificate verify error

At this point there is no way for the users to decide whether their browsers talk to the real neptun.uni-obuda.hu or to a pirate server which personalizes the real neptun.uni-obuda.hu server.

There exist possibilities to sort out this problem, of course. First of all Óbuda University ought to get a digital signature which could be verified by most browsers, if not all of them. The second possibility is to give the students and teachers a piece of paper holding the fingerprint of the certificate of neptun.uni-obuda.hu. In the first case the problem would not even exist any more without any user action. In the latter case users could verify the fingerprint and if (and only if) it was correct they could accept it manually (See fig. 3.) once and for all. Students could, of course, make a telephone call to the system administrator in order to check the fingerprint. Try to imagine the situation when about 12 thousands students ring the sysadmin up with the same question...

Figure 3.
The fingerprint(s) of the certificate of neptun.uni-obuda.hu

Without one of these steps there can be no guarantee for the user that his or her browser communicates with the original and official neptun server of Óbuda University. In spite of this risk not only students usually accept the certificate without verifying but teachers do the same as well. So a man in the middle attack can be performed not only theoretically but practically, too. What leads us to face this situation is that that our certificate cannot be verified automatically, so browsers produce a pop-up window. It soon becomes an everyday routine to say O.K. on the pop-up window, of course, without a proper personal check. In this case it will not be realised by anybody if the certificate (and the server itself, of course) is changed. Let us see at least one example, but before that take a look at the possible motivations.

2.2 Possible motivations

The first step in risk analysing to see what kind of motivations could be taken into account. Why would anybody try to realise a man-in-the-middle attack, why

would anybody try to capture the data of the normal and everyday data flow of the scholar administration?

First of all: the only valuable element of data traffic to or from the neptun server are the username and password pairs, so it is enough to get only them. What are thely good for?

Having the username and the corresponding password of someone else another student can log in to the Neptun to sign the password owner out of courses or examination dates e.g. in order to have a free place for another one, even for him/herself. This challenge is not worth the risk of some years to be spent in prison.

If teacher accounts are caught one will be able to sign in in the name of the given teacher to do any of the teacher's jobs, e.g. to give a valuable mark to a student. Knowing that there are no checks after the teacher filld in the forms of a given cours, it could be a bit more interesting possibility. This is also belongs to the category of students' tricks, which is not worth the risk of going to prison.

If an attacker can collect a large number of logins, the situation will be much more interesting.

A not little part of the students must pay a fee of about 700 EUR for their studies. Students can change their own bank account number themselves after a successful login. Students can or, better to say, must pay any kind of fees via the Neptun. In the first step they must make a money transfer to a given bank account. In the second step they must use the Neptun to indicate what purpose is the money for. After they marked the purpose the university can do a second money transfer to its own bank account. Students have the possibility to give order to the Neptun to send their money back to their own bank account, number of which can be changed by the students themselves. Let us suppose that the attacker can get the logins of about one thousand students (approximately less than 10 percent of our students) who are supposed to pay the above mentioned fee. In this case (s)he could make money transfers of about 700.000 EUR to a fake bank account. This sum is worth a bit of risk to some people, I think.

Of course, all the personal data of the victims could be collected for other purposes, e.g. to sell it, or to use them in fake transactions, e.g. founding phantom enterprises and so on.

Last but not least the collected passwords could be tried if their owners would use the same password at gmail.com, at facebook.com etc., so to try to steal the digital personality of the victims. Based upon broken mailboxes e.g. some nigerian type tricks could be initiated.

Summarizing the above possibilities we can state that real and serious motivations exist, so our university ought to be much more cautious.

3 An Example for MITM

3.1 Address Resolution Protocol

ARP stands for Address Resolution Protocol. This protocol is responsible for controlling the network traffic. If a computer needs to send a packet of data to another computer connected to the same subnet, first it should know the 6-byte MAC (Media Access Control) address of the network interface of the recipient. In TCP/IP networks, the MAC address of a subnet interface can be queried with the IP address using the Address Resolution Protocol (ARP).

Sender computer performs an ARP query in broadcast mode by which it asks all the computers of the subnet which MAC address belongs to the given IP address. The only machine having the appropriate IP address will give its MAC address in an ARP reply. In the next step the sender machine can send ethernet frames to the given MAC address. Machines store the appropriate IP and MAC address pairs in their ARP cache for a given time period. After that time the sender must ask the MAC address again.

3.2 Getting Access to the Local Subnet

If someone can crack a computer in a Neptun lab or can use his/her own laptop a so-called ARP poisoning can be made. The pirate computer broadcasts fake ARP replies let's say one in every second, which replies state that the IP address of the default gateway of the given subnet has the MAC address of the pirate computer. All the computers in the given subnet will store this pair of data in their ARP cache. The result of this is that all of the outgoing ethernet packets will be directed to the attacker laptop instead of the real and authentic gateway. You can download tools for that stuff from the internet, see e.g. the dsniff package of Dug Song. [1] Arpspoof as a part of that package will do the trick.

Dsniff was originally written by Dug Song. Dsniff is a collection of tools for network auditing and penetration testing. Dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspy passively monitor a network for interesting data (passwords, e-mail, files, etc.). Arpspoof, dnsspoof, and macof facilitate the interception of network traffic normally unavailable to an attacker (e.g, due to layer-2 switching). Sshmitm and webmitm implement active monkey-in-the-middle attacks against redirected SSH and HTTPS sessions by exploiting weak bindings in ad-hoc PKI.

3.3 Fakeing the DNS

If you give the name of a remote computer as a part of the url, the browser should decide the IP address of that computer. In our example the browser should trace down the IP address belongs to the name neptun.bmf.hu. This is done by DNS (Domain Name Service) servers, servers which can tell which IP address belongs to a given computer name. This is done by sending a DNS query to the udp port 53 of the nearest DNS server. Trying to do this the appropriate data packet which normally would go to the default gateway of the subnet in our case goes to the pirate laptop.

The pirate laptop (or desktop computer) can send a fake answer using the above mentioned dnsspoof to the client browser which states that the IP address of neptun.bmf.hu is that of the fake laptop itself. The original DNS queries for neptun.bmf.hu must not be forwarded to the real DNS server while any other requests are to be forwarded to the original gateway, so it is necessary to enable IP forwarding on the attacking machine. By this time we succeeded in becoming a man (or woman) in the middle. At this point we redirected http(s) requests to the pirate laptop instead of the original and authentic neptun.bmf.hu.

3.4 Personalizing the Original Server

At this moment the situation is the following in the computer lab, more precisely on the subnet which the pirate machine belongs to. All the data transfer goes through the attacking machine because of the fake ARP answers. DNS queries for neptun.bmf.hu are also faked by dnsspoof, so https requests for neptun.bmf.hu and only for that goes to the attacking machine instead of the original one. All other traffic is redirected to the original gateway of the subnet.

The attacker saves the original opening pages of neptun.bmf.hu, at https://neptun.bmf.hu/oktato/login.aspx for teachers, and at the url https://neptun.bmf.hu/hallgato/login.aspx for students which is not a complex task. By the help of the webmitm program (web monkey in the middle, part of the dsniff package of Dug Song), the pirate laptop can be used as a transparent https proxy with the addition that it logs the user names (neptun codes) and the belonging passwords. So the situation is just like in the story of the bandit, the sheriff and the deputy, but neither the students nor the teachers will know that they have a deputy as an interpreter.

Only a fake certificate is needed which can be produced by openssl which contains the same names than the original certificate of the authentic neptun.bmf.hu. Of course the value of the public key will be different, so the fingerprint of the certificate will differ as well but nobody will recognise it because everybody has accustomed to the annoying warnings about the certificate.

5 Summary

If the certificate of neptun.uni-obuda.hu was issued by a verifiable certificate authority (CA) then no man in the middle attacks could successfully be performed without the serious carelessness of the end-users. But students and teachers has got used to those windows of the browser in which it complains on the certificate. So they will enter an OK, as they did it before so many times as well without noticing that the fingerprint of the certificate has changed.

This is a serious security hole which must not exist at our university especially because one and a half year ago we were in the same situation for many months.

Of course I did not make the above described procedure to steal passwords and other personal data. I can only hope that nobody else did, do nor will try it. Or could this backdoor be closed?

REFERENCES

[1] Dug Song: dsniff. without place, 2000.; http://monkey.org/~dugsong/dsniff/ (download: 09-09-2009)

[2] Fielding, R. - Irvine, UC - Gettys J. - Mogul J. - DEC - Frystyk H. - Berners-Lee T.: Hypertext Transfer Protocol - HTTP/1.1.. MIT/LCS, 1997.; http://www.rfc-editor.org/rfc/rfc2068.txt (download: 09-09-2009)

[3] Robinson, Sarah: Still Guarding Secrets after Years of Attacks, RSA Earns Accolades for its Founders. SIAM News, Volume 36, Number 5, June 2003. pp. 1-4.

[4] Schneier, Bruce: Applied cryptography: Protocols, algorithms, and source code in C. Wiley & Sons, New York, 1996. (2nd ed.) pp. 265-301.

[5] Szikora Péter: Measured Performance of an Information System. 7th International Conference on Management, Enterprise and Benchmarking, Budapest, 2009. pp.

[6] Szikora Péter: The Role of the Tools and Methods of Implementation in Information System Efficiency. 2nd International Conference for Theory and Practice in Education, Budapest, 2009. p. 50.

[7] Zimmermann, Philip: The official PGP user's guide. MIT Press (Cambridge, Mass), 1996. ISBN 0262740176. Originally part of the PGP program package: ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/IntroToCrypto.pdf pp. 47-50.

[8] HTTP Over TLS. http://www.rfc-archive.org/getrfc.php?rfc=2818