First International Conference
Modelling and Development of Intelligent Systems
Sibiu - Romania, 22-25 October, 2009

The computer's vulnerabilities

Mircea Iosif Neamţu

Abstract

The vulnerabilities are „exchange tricks" used by hackers, which allow them to enhance access possibilities by exploiting a logic error in one of the computer's code. A vulnerability can be caused by a project-error, by an error occurring after using some COM type objects, by an limit-checking-error that can occur while interpreting HTML tags to which a large number of scrip-functions has been associated to (thousands) etc. We can say that the different types of vulnerabilities and the ways in which they are exploited are random and chaotic while every way has, apparently, an unexpected outcome. My intention, through this article, is to draw attention upon the fact that vulnerabilies are an imminent threat and at the same time, to convince that this threat can be removed.

1. Weaknesses in computers

By analysing vulnerabilities, we observe that these are weaknesses within the security-plan, an error that can be exploited to overcome security.

Security is the basic structure that prevents unauthorized access to the system. When a vulnerability is exploited, the person using this vulnerability can gain some extra-influences within the system and thus, compromising its integrity.

In conclusion, there are four types of basic vulnerabilities: logic errors (e.g. an attack on a protocol), weaknesses (e.g. a program that captures packages from the web), social engineering (e.g. false phone calls) and surveillance-politics – while being related with two factors:

- the specific target of the vulnerabilities in terms of the computer or the person,
- how fast a vulnerability can be achieved.

Therefore the current features of these attributes are independent and some problems can be identified the same as the four types, mentioned above.

Most software-types inherit the vulnerabilities of the operating system. There are vulnerabilities of the operating system that represent the most direct attack-methods, showing almost instant reaction-types with unpredictable results.

A vulnerability-example for an operating-system is the ***chroot*** vulnerability. Before reporting and solving this vulnerability, it could be found within Unix systems. This vulnerability could have been used for changing access-privileges, by creating a password-file and offering a neutral password for the root account.

Applications can be represented starting from a game all the way to a web-server. The programs can also be written both by programmers with exceptional abilities and by amateurs.

Let's take into consideration an error within the Microsoft internet browser Internet Explorer.

Both the application-errors, as well as the errors from within the operating system can be found on the highest executing speed. There are errors that don't appear until a certain condition is fulfilled that the attacker requires specifying the computer that runs the application-error.
As an example we use the game-application LARN (in a package of more versions of the operating system BSD), accidentally programmed with a vulnerability that allows the administrator access to host.

Description of the game's LARN vulnerability; BSD 4.4. For e.g.: if a person scores 263 points, the game forces the system to send an e-mail to the user. The e-mail-sending process causes an IFS vulnerability that can be used to exploit the root-access.
This attack-example doesn't happen instantaneously, although it was given as an example to show that situations that are not automatically provoked can also occur.

Another feature that generates discussions, similar with vulnerabilities, is the notion of weakness of computers that often creates confusions.
Vulnerability always has a remedy, while a weakness might never have a remedy. Regarding the computer weaknesses, they say that: „a chain is only as strong as its weakest link".
Critical security-elements targeted by weaknesses are:

- security by obscurity
- encrypting
- password-security
- secured hash-functions
- old software
- old hardware
- people

Each of those are elements will break down in time, although they can be upgraded to correct that weakness.

2. The WI-FI - WEP, WPA, WPA2 vulnerabilities

WI-FI (Wireless Fidelity) is part of the top-technologies from the gamma transmissions without a cable. The support for the Wi-Fi technology is already inserted within a multitude of equipments: laptops, PDAs, mobile phones, etc., but unfortunately, the security-aspect has passed on unobserved. Let's now closely analyze the level of safety for the encrypting methods used within modern implementations of the Wi-Fi technology.

Even if the security methods are activated on the Wi-Fi equipments, in most cases, the used WEP protocol is a low level encrypting-protocol. An analysis of the WEP protocol demonstrates the vulnerability of this system by using a WEP-key. We observe the need for introducing a new security-architecture in the form of the 802.11i standard for solving some of the vulnerabilities of the Wi-Fi technology.

2.1.1. WEP (Wired Equivalent Privacy)

The WEP encrypting-protocol wasn't created by experts from the security field or from the encrypting area, fact that has thus generated the vulnerability within the RC4 algorithm problem. This problem was first described by David Wagner, Scott **F**luher, Itsik **M**antin and Adi **S**hamir who presented the two vulnerabilities of the RC4 algorithm. The two attacking methods rely on the fact that for some key-values it's possible that some bits from the actual bit depend only on a couple of bits from the encrypting-key (normally, each key has a 50% chance to be different from the latter). WEP has been the implicit encrypting protocol first introduces in the first standard IEEE 802.11 in 1999. At the base of the WEP protocol lies the RC4 encrypting algorithm, this includes a secret key on 40 or on 104 bits combined with an initializing vector (IV) on 24 bits for encrypting the text message (M) and the control bits (checksum, ICV–Integrity Check Value).

The encrypted message (C) was determined by using the following formula:

$$C = [M \,||\, ICV(M)] + [RC4(K \,||\, IV)] \quad (1)$$

Where: „||" is the concatenating operator and „+" is the XOR operator.

The initializing vector (IV) is the WEP security-key and is used for maintaining a medium security level and to minimize detection.

The IVs must be enhanced within every package so that all following packages are encrypting with different encrypting-keys. Unfortunately for the WEP security, the IVs are transmitted only in a simple text within the 802.11 standard and don't require incrementing the IVs allowing the implementers of the particular wireless terminals to chose the needed security measure (access points and WLAN board).

After presenting this vulnerability, the fact that the WEP protocol offers an acceptable level of security only for its use within applications of lesser importance, was realized. Afterwards, this level of security was crushed once attacks appeared that allow arbitrary packages to be decrypted without knowing the key (KoreK and Arbaugh) by using the injection-method. It was proven that cracking tools such as Aircrack and Weplab using this method succeed in recuperating the key encrypted by 128 bits in less than 10 minutes. As a recommendation, the WEP protocol shouldn't be used, not even with the rotation of the encrypting key.

2.1.2. Treating vulnerabilities within the WiFi 802.11i technology

January 2001 the „i" group within the IEEE was created to enhance security of the authentication and encryption of data within the 802.11 standard. April 2003, the WiFi alliance published a recommendation regarding wireless security within companies. Concern was shown regarding the consumers' availability to change their existing equipments. June 2004 the last update of the IEEE 802.11i standard was adopted, being given the commercial name WPA2 from the WiFi alliance. This standard introduced some fundamental changes amongst which the most important being the separation of the user's authentication from the integrity of the message. This creates a robust security-architecture and scalable applicable to any kind of network. By using the new standard, the new architecture for wireless networks called Robust Security Network (RSN) uses another key distribution and a new integrity mechanism. Because the RSN architecture is more complex, it offers safety-solutions and scalability for wireless communication. Generally, a RSN only accepts RSN-capable equipments. The new standard defines also a network security architecture to which both RSN and WEP systems can participate, thus allowing users to change their equipment in time. The authentication procedures or the association procedures that are used between stations for the communication method in 4ways (4-way handshake) are called RSNA (Robust Security Network Association).

In order to achieve a secured connection, four well defined fazes must be run through.

1. Convention upon the security policy.
2. Authentication 802.1x.
3. Grading keys and distribution.
4. RSNA, confidentiality and integrity of data.

2.1.3. Weaknesses of the WPA/WPA2 protocols

Since publishing the WPA/WPA2 protocols, a small number of minor security deficiencies were found, none being though too dangerous while respecting the conditions for a simple security recommendation.

Regarding vulnerabilities, the fiercest is the attack against the PSK key for WPA/WPA2 protocols. The PSK offers an alternative to the 802.1 X PMK generations by using an authentication server. For using a known algorithm such as PSK a sequence of 256 bits is needed or an access password containing 8 to 63 characters for generating the line.

The algorithm PSK = PMK = PBKDF2 (password, SSID, length SSID, 4096, 256), where: PBKDF2 is the used method, 4096 is the number of hashes, and 256 is the exit length. Worth mentioning would be

that the second message within the information exchange in all four ways can be subjected both to dictionary attacks, as well as to the bruteforce –type ones. This is why a utilitarian was invented to exploit this error, its source-code being used and improved by Christophe Devine in Aircrack. The Aircrack utilitarian can be used for PSK type attacks (dictionary and bruteforce) on wireless networks with WPA security protocols.

Because of the protocol's component (4096 hashes for each password try) a bruteforce attack is very slow (with a processor of the latest generation with a single nucleus only about 100 passwords can be loaded per second).

The PMK can't be pre-calculated because the password is encrypted based on the ESSID. To protect ourselves from this vulnerability we must have a password with a minimum of 20 characters which isn't a word from a dictionary.

The attack consists in capturing the exchange of information in the 4 ways (4-way Handshake) by passively monitoring a wireless network or by using an authentication attack to speed up this process. Actually, the first two messages are required to start guessing the PSK values.

The second important weakness of the WPA protocol is the DoS possibility during the exchange of information in the 4 ways. John Mitchell noticed that the first message within the 4 ways isn't authenticated; each client must save his message until receiving a signed message leaving the client vulnerable until exhausting the memory. By listening to the first message sent by the access point, the attacker can perform a DoS to the client with more simultaneous sessions.

The last known weakness is the possibility of a theoretic attack against the temporary hashes of the WPA key protocol. This attack implies a low level complexity in some circumstances and also knowing more RC4 keys.

WPA/WPA2 also are subject to vulnerabilities that affect other mechanism of the 802.11i standard, such as: e.g. the listening-attack of 802.1X messages (EAPoL Logoff, EAPoL Start, EAP Failure etc.), first described by William Arbaugh and Arunesh Mishra; this being possible in case of a lacking authentication.

Not last, it's important to know that the usage of the WPA/WPA2 protocol doesn't provide a current protection against the attacks that stand at the base of these technologies, such as blocking radio frequencies.

2.1.4. Implementing WPA/WPA2 in SO

Within the operating system Windows XP, the support for the WPA2 protocol isn't built in. An update of the operating system Windows XP SP2 (KB893357) was performed on the 29th of April 2005 containing the support for the WPA2 protocol with an improvement of detecting wireless networks.

Other Microsoft operating systems must use an external applicant (commercial or Windows open source). The Linux and BSD platforms, *wpa_supplicant* were already prepared for WPA2 as soon as the 802.11 standard was published.

The external applicant only bears a large number of EAP methods and key management-methods for WPA, WPA2, WEP. More networks can be declared with different types of encryptions, key management and EAP methods. The default location of the wpa_supplicant configuration file is /etc/wpa_supplicant.conf and the file must only be accessed by the root.

On the Macintosh platform the WPA2 protocol already had a support at release of the 4.2 update of the Apple AirPort software.

2.1.5. Firewall protection

A frequent encountered error is considered to be the *firewall* application which can detect and block a message mail based attack.

The *firewall* applications control the network connectivity and normally don't verify the traffic taken place on the standard e-mail port (25). The network administrator can add rules that allow specifying the types of traffic that have the permission of passing the firewall. For example, the traffic

on the 25th port is seeing as an e-mail, which might allow an attacker the use of the 25th port for launching an attack.

2.2. Conclusion

It's obvious that encrypting with the WEP protocol doesn't offer a sufficient wireless security and can only be used with high level encrypting solutions (e.g.: VPN).

The WPA protocol is a secure solution for equipments that still don't have the support for the WPA2 protocol, protocol that will become a standard wireless security. Let's not forget the fact that the wireless equipment must be placed in a filtered area always prepared for a cable connection for critical networks because the blockage of radio frequencies and low level attacks can still be devastating for a wireless network with the encrypting protocol WPA2.

3. Case-study: Treating method of vulnerabilities infecting memory sticks

Because the use of memory sticks has increased, the malwares have started to take advantage of this situation to spread rapidly. Because a single memory stick can be used on many computers, this has become a problem for users. The following presented method can be helpful in diminishing the risk of infecting a memory stick used daily on more than one computer.

The malwares use two techniques for spreading. The first technique is characterized by the fact that it infects the executable files from the stick and by executing those files they migrate to the computers.

The most common and dangerous method is spreading through the ***„autorun.inf"*** file, file which the operating Windows system automatically executes after inserting the memory stick into the computer. Many malwares use this spreading technique (e.g. the *Conficker* worm).

3.1. Solution

One solution would be to deactivate the *autorun* mechanism from within the Windows SO, this requires although technical IT knowledge.

Because most users don't have this certain knowledge, preventing the execution of the *autorun.inf* file must be directly solved on the memory stick.

Before applying this method we can also say that there also exists a possibility of acquiring a type of memory sticks that contain a switch for „read only" or „write-read" modes. These sticks thus can be blocked so that transfer of viruses can't be made from the computer to the stick or the other way around. If the file system of the stick is a FAT32 (< 8GB+) type, than a small trick can be applied presented next. This trick needs a hexa editor and some basic knowledge regarding the director table of the FAT32 filing system.

Step 1. A blank file is created on the stick, called „autorun.inf",

Step 2. The stick is being opened with the hexa editing program.

Observation: It's not important whether the physical disc or the local partition is being opened but we must assure that the stick was opened with the required read/write permissions and that no other program is accessing it in that moment.

As an example, I used an HxD, a portable hexa editor. If we don't have a new stick and our stick already contains information, it's recommended that this information is copied on another support and the stick to be formatted. After creating the empty file called „autorun.inf" and opening the stick with HxD, the hexa editor, we than start a search for „AUTORUN", as a Unicode string text. If the stick was empty, then it must be found right at the beginning of the disc.

The interest area is the following:

41 55 54 4F 52 55 4E 20 49 4E 46 20
A U T O R U N I N F

The first 8 bytes stand for the name of the file (followed by a space), followed by 3 bytes (the file extension), and followed by 1 byte containing the attributes of the file. This last byte is the relevant one. The current value of this byte is (0x20), meaning that the file has only the archive byte set.
Our objective is to change this byte into 0x40 (setting the bit of the equipment).
The area then changes into:

41 55 54 4F 52 55 4E 20 49 4E 46 40
A U T O R U N I N F @

Once this change has been saved on the disc, ignoring any warning that might corrupt the disc, we first remove it from the computer and then we reattach it to the computer.
Now, the file *autorun.inf* is visible but can't be deleted, edited, overwritten or have its attributes changed.

When this stick is connected to an infected computer that tries creating a *autorun.inf* file on the stick, it will inform that the „file can't be created" which means that it can't be infected through this method (*autorun.inf)* and can't spread an infection to other computers.

4. Conclusion

This method is addressed to all users of memory-sticks. If this technique will start being used by more and more users, it will surely assure a simple but efficient method of protection against the distribution of the M-stick viruses.

References

1. Protected Access and 802.11i (John Edney, William A. Arbaugh) – Addison Wesley – ISBN: 0-321-13620-9,
2. The Insecurity of 802.11 (Borisov, Goldberg, Wagner),
3. Criptografia şi Securitatea Reţelelor de Calculator (Victor-Valeriu Patriciu), Editura Tehnică
4. Protecţia şi Securitatea Informaţiilor (Dumitru Oprea), Editura Polirom
5. Insider Attack and Cyber Security Beyond the Hacker (Salvatore J. Stolfo, Steven M. Bellovin), Springer
6. Securitatea Sistemelor Linux (Dragos Acostachioaie), Editura Polirom
7. Assessing Network Security (Kevin Lam, David LeBlanc, Ben Smith), Microsoft Press
8. Hack Proofing Your Network (Internet Tradecraft), Syngress Publishing
9. The Hacker Crackdown. Law and Disorder on the Electronic Frontier (Sterling B.)
10. Building Internet Firewalls (Chapman B.), O'Reilly and Associates
11. Practical UNIX and Internet Security (Garfinkel S., Spafford G.), O'Reilly and Association
12. Hackers Beware (Cole E.), New Riders Publishing
13. Designing Secure Web-Based Applications for Windows 2000 (Howard), Microsoft Press

Weblinks

1. http://standards.ieee.org/getieee802/download/802.11i-2004.pdf – IEEE 802.11i standard,
2. http://www.awprofessional.com/title/0321136209 – Real 802.11 Security Wi-Fi
   http://www.cs.umd.edu/~waa/attack/v3dcmnt.htm – An inductive chosen plaintext attack against WEP/WEP2 (Arbaugh),
3. http://www.cr0.net:8040/code/network/aircrack/ – Aircrack (Devine)

Mircea Iosif Neamţu
"Lucian Blaga" University
Computer Science Department
Str. Dr. Ioan Raţiu nr.5-7 ROMÂNIA
e-mail: neamtu12@yahoo.com