DOI https://doi.org/10.36059/978-966-397-104-9/139-161

CRYPTOSYSTEMS AND INFORMATION SECURITY IN INTRANET

Muliava O. M.

1. Evaluation of the reliability of cryptosystems

A group of well-known cryptographic experts, created under the auspices of the Business Software Alliance (an industry organization that prevents software misuse), has come to the conclusion that the required key length should now be at least 75 bits with a further increase over the next 20 up to 90 bits.

Let's check this statement.

The problem of finding keys in a symmetric cryptosystem by sorting through all the possible keys belongs to a class of problems that allow parallelization. Applying Distributed Computing to Organize the Search of Such Keys[1].

The exponential growth dynamics of computing systems' performance over time (10 times in 5 years) has an even more significant impact on the overall performance of the system.

Thus, progress in this area is possible due to:

1) use of advances in scientific and technological progress and the use of technological innovations to increase the productivity of an individual device;

2) increasing the number of such devices in the system. From a physical point of view, that type of transistor, which is the basis of the modern integrated circuit, can be reduced by about 10 times, to a size of 0.03 microns. At this limit, the process of switching on/off the microscopic switches will be practically impossible. Thus, the maximum speed will be 1016 operations/second, and the limit of growth will come in about 2030.

There are no other ways to increase the computing power.

Thus, from the point of view of information security by cryptographic methods, the analysis of potential capabilities of the distributed computing method is of great interest for both crypto-analysts and developers of

[1] Немет Э., Снайдер Г., Сибасс С., Хейн Г.Р. UNIX: руководство системного администратора. Пер. с англ. Киев: BHV, 1996. 270 с.

cryptographic systems. So let's try to analyze the limit values of two of these trends.

From the list that appeared in the summer of 1999, it follows that supercomputers are distributed as follows:

with a power of about 10^{12} FLOPS 3 approx.;

with a power of about 10^{11} FLOPS 54 approx.;

with a power of about 10^{10} FLOPS 428 approx.;

with a power of about 10^{9} FLOPS 251 approx.;

The first place in the world in the number of supercomputers is occupied by the USA 254 (51%), followed by Japan 87 (17.5%), Germany 45 (9%), Great Britain 24 (4.8%), France 18 (3.6%)), Korea 8 (1.6%), Canada 7 (1.4%), Sweden, Switzerland and Norway 6 each (1.2%). Russia is mentioned only once in this list: the 156th place is the LDC Ultra 10000 computer (peak performance 16600 MFLOPS), manufactured by SUN and installed in the National Reserve Bank of Russia.

Десять самых мощных суперкомпьютеров в мире по состоянию на 1999 г.

Рейтинг	Наименование машины	Страна-обладатель	Фирма-производитель	Количество процессоров	Мощность (GFLOPS)
1	Intel ASCI Red	США	Intel (США)	9125	1333
2	Hitachi/Tsukuba CP-PACS	Япония	Hitachi/Tsukuba (Япония)	2048	368
3	SGI/Cray T3E	Великобритания	Cray (США)	696	265
4	Fujitsu Numerical Wind Tunnel	Япония	Fujitsu (Япония)	167	230
5	Hitachi SR2201	Япония	Hitachi (Япония)	1024	220
6	SGI/Cray T3E	Германия	Cray (США)	512	176
7	SGI/Cray T3E	США	Cray (США)	512	176
8	SGI/Cray T3E	Германия	Cray (США)	512	176
9	SGI/Cray T3E	США	Cray (США)	512	176
10	SGI/Cray T3E	США	Cray (США)	512	176

Interesting detail: there are no foreign computers in the US. Americans only work on domestic machines and also supply them to the whole world.

Power attack on cryptosystems is futile. However, the disadvantages of the algorithms can significantly reduce the number of search options.

Use as meaningful words as a key allows you to use dictionary attack. Therefore, further development of cryptography will occur in the field of cryptanalysis.

2. Why are cryptos not reliable?

In modern software, crypto algorithms are widely used not only for data encryption tasks but also for authentication and integrity checking. To date, there are well-known and proven crypto-algorithms (with symmetric and asymmetric keys), whose cryptocurrency is either proved mathematically or based on the need to solve a mathematically difficult problem (factorization, discrete logarithm, etc.).

The most famous of them are DES, GOST, RSA. Thus, they can not be disclosed other than a complete search or solution of the specified problem.

On the other hand, in the computer world, information about errors or "holes" in a particular program (including using crypto algorithms) or that it has been cracked is constantly being displayed. This creates a distrust of both specific programs and the ability to protect anything by cryptographic methods not only from special services, but also from simple hackers[2].

Therefore, knowledge of the history of attacks and "holes" in cryptosystems, as well as understanding the reasons for their occurrence, is one of the necessary conditions for the development of secure systems. A promising area of research in this area is the analysis of successfully conducted attacks or revealed vulnerabilities in cryptosystems in order to generalize, classify and identify the causes and patterns of their appearance and existence.

By, by analogy with the taxonomy of the causes of OS security breach, let us distinguish the following reasons for the unreliability of cryptographic programs:

- inability to use stable crypto algorithms;
- errors in the implementation of crypto algorithms;
- improper use of crypto algorithms:
- human factor.

Note that the following causes cover only two types of potential threats: disclosure and integrity, leaving aside the threat of denial of

[2] Гайкович В., Першин А. Безопасность электронных банковских систем. Москва: Единая Европа, 1994. 264 с.

service, which is increasingly important as distributed cryptosystems develop.

Inability to use persistent crypto algorithms

This group of reasons is most common because of the following factors.

The speed of stable crypto-algorithms is low

This is a major factor complicating the use of good algorithms, for example, in "total" encryption or "on-the-fly" encryption systems. In particular, Norton DiskReet, although it has a DES implementation, may not re-encrypt the entire disk when it is changed by the user, since this will take a very long time. Similarly, on-the-fly Stacker compression software from Stac Electronics has the option of closing off the offset data. However, it has no physical ability to encrypt its file with this password, it is usually several hundred megabytes in size, so it is limited by a very weak algorithm and stores the hash function against the password along with the protected data.

Export restrictions

This is due to the export of crypto algorithms or the need to buy a patent or rights. In particular, the export of crypto algorithms with key lengths greater than 40 bits is prohibited from the US. Obviously, such crypto-stability cannot be considered reliable with modern computing power and even on a personal computer. Putting the speed of the search at 50,000 passwords/sec, we get a search time of an average of about 4 months.

Well-known examples of programs that are subject to export restrictions are recent browsers of the Internet, including Netscape Navigator from Netscape Communications and Microsoft's Internet Explorer. They provide encryption with a 128-bit key for users in the US and a 40-bit key for everyone else. Also in this group is the latest version of the ARJ 2.60 archiver. known for its weak archive encryption algorithm. known for its weak archive encryption algorithm.

Users in the US can now use the GOST crypto-algorithm. The comism of the situation is that, although this algorithm is Russian, even Russians under US law can still not use it in the ARJ program.

Using your own crypto algorithms

Ignorance or unwillingness to use known algorithms is a paradox, especially in Freeware and Shareware applications, such as archivers.

As already mentioned, the ARJ archiver (up to version 2.60 inclusive) uses (by default) a very weak encryption algorithm – simple gamming. It would seem that in this case its use is permissible, since the archived text should be completely redundant and statistical methods of cryptanalysis are not suitable here. However, upon closer examination, it appeared that some non-random information is present in the archived text (and this is true for any archiver) – for example, the Huffman table and some other official information. Therefore, knowing exactly or predicting with some probability the values of these service variables, it is equally possible to determine the corresponding password characters.

Further, the use of weak algorithms often leads to the success of a plaintext attack. In the case of an ARJ archiver, if an attacker knows at least one file from an encrypted archive, it can easily determine the password of the archive and extract from it all other files (ARJ cryptocurrency with open text – 2^0!). Even if there is no encrypted file, it is still simple gamification allows you to reach a speed of 350,000 passwords/sec on a Pentium machine.

The same is true for popular Microsoft Office applications – you only need to know the 16 bytes of a .doc or .xls file to determine your password, and then just browse through 24 options. Microsoft Office 97 made significant improvements to the encryption algorithms, leaving only the possibility of a complete search, but ... not everywhere – MS Access 97 uses a primitive algorithm, and not the data itself, but the password itself with a fixed constant XOR operation!

Novell Netware's Novell networking system (version 3.x and 4.x) also has its own hashing algorithm. At the input, the hash function receives a 32-byte value obtained in the original user password by either compressing a password longer than 32 characters by XOR operation, or multiplying a password less than 32 characters, and outputting a 16-byte hash value (Hash 16). It (for Novell Netware 3.x) is stored in the database bindery as a property of "PASSWORD".

One of the key features of a hash function is that it must not allow easy collision building (such as the crypt() function used in UNIX, which is DES-based). This property is broken in the hash function used in Novell Netware.

The hash algorithm under consideration has remained in version 4 of Novell Netware.

Microsoft, in turn, also has major shortcomings in its mainstream hash algorithm, which is applicable to all its operating systems, starting with Windows 3.11, when authenticated on local (NetBIOS) and global (CIFS and http) networks, called LM (Lan Manager) – X3in. (However, Microsoft refers to the fact that it has remained since OS / 2 and that it was developed by IBM).

It is calculated as follows: the password is converted to a 14-character string by either cutting off long passwords or supplementing short passwords with null elements.

All lowercase characters are replaced with uppercase characters. Numbers and special characters remain unchanged.

The 14-byte string is split into two seven-byte halves.

Using each half of the string as a DES key, it encrypts a fixed constant, yielding two 8-byte strings at the output.

These lines merge to create a 16-bit hash value.

Obviously, attacks on the LM hash are easily successful for the following reasons:

Converting all characters to uppercase limits the already small number of possible combinations for each (26 +10 + 32 = 68).

Two seven-byte "half" passwords are hashed independently. Thus, the two halves can be sorted independently, and passwords longer than seven characters are no stronger than passwords with a length of seven characters.

There is no salt element, as in crypt() – two users with the same password will always have the same hash value.

Thus, you can pre-compile a dictionary of hashed passwords and search for an unknown password in it.

Wrong implementation of crypto algorithms

Although crypto-resistant or certified algorithms are used in this case, this group of reasons leads to a security breach of cryptosystems due to their incorrect implementation.

Decrease in cryptocurrency during key generation

This is because of the many examples where the crypto system either cuts the user's password or generates data with fewer bits than the password itself. Examples:

In many (older) versions of UNIX, the user password is truncated to 8 bytes before the hash. Interestingly, for example, Linux 2.0, requiring

users to enter passwords containing necessarily letters and numbers, does not verify that the 8-character password start also consists of letters and numbers. Therefore, by asking, for example, a sufficiently strong password 'passwordIsgoodlÇ', one would be very surprised to learn that a hacker has logged in under his name with a simple password 'password'.

Novell Netware allows users to have passwords up to 128 bytes, which gives (including Latin letters without case, numbers and special characters) $68^{128} \sim 2^{779}$ combinations. But first, the hash function (see above). It only receives a 32-byte value at the input, which limits the effective length of the password to the same value. Moreover, secondly, the output hash value is only 128 bits long, which corresponds to 2^{128} combinations. This further reduces the effective length to =21 символа[3], ie 6 times compared to the original length.

The situation with the RAR 1.5 * archiver is quite similar – choosing a password of more than 10 characters does not increase the time it takes to open it.

If the length of the "top" password in this case is determined by the implementation of cryptographic algorithms, then the restriction on the length of the "bottom" is already associated with the concept of unit of information or entropy. In the considered example from Novell Netware, to create a hash value with an entropy of 128 bits, the password length must be at least 22 characters. The fact that many cryptosystems do not limit the minimum length of the password, which in turn leads to the success of attacks not by keys and passwords.

Lack of checking for weak keys

Some crypto-algorithms (such as DES, IDEA), when encrypting with specific keys, may not provide the right level of cryptocurrency. Such keys are called weak. DES is known for 4 weak and 12 semi-weak. (Semi-weak) keys. Although the probability of getting into them is $\sim 2 * 10^{-16}$, for serious cryptographic systems it can not be neglected.

The power of many IDEA weak keys is 2^{51} (however, because of the total number of keys 2^{128}, the likelihood of getting into it is 333 times less than in DES).

Insufficient security against MS

MS (malicious software) are computer viruses, Trojan horses, software bookmarks, etc. applications that can intercept the private key or the unencrypted data themselves, and simply replace the algorithm with

non-encrypted ones. If the programmer has not provided sufficient security against the MS, they are easily capable of negatively affecting the security of the cryptosystem. This is especially true for operating systems that do not have built-in security or access control features, such as MS DOS or Windows 95:

Password Interception

An example is the oldest method of password stealing, known since the days of major computers, when the phantom program emulates the OS invitation, offering to enter a user name and password, memorize it in some file and stop working with the message 'Invalid password'. For MS DOS and Windows there are many bookmarks for reading and saving passwords that are typed on the keyboard (by intercepting the corresponding interrupt), for example, when using Diskreet V. 6.0.

Crypto algorithm replacement

An example of this is the bookmark masked under the Turbo Krypton-type accelerator application. This tab replaces the GOST 28147-89 encryption algorithm implemented by the Kgurton-C board (demo version) with another, simple and easily decrypted algorithm.

Trojan horse in email

A recent example is the June 1998 attempt to infiltrate a Trojan horse via email. The letter contained a pornographic image and an EXEC file FREECD.EXE, which, while the user was having fun with the letter, decrypted the passwords to the provider (Dial-Up) and sent them to ispp@usa.net.

Time limit for key processing

This is a relatively new aspect of the lack of correct implementation of crypto algorithms discussed in the article. There it is shown that many cryptosystems process different input data differently quickly. This is due to both hardware (different cycles per operation, CPU cache, etc.) and software reasons (especially when optimizing the program over time). Time may depend on both the encryption key and (de) encrypted data.

Therefore, the attacker, having detailed information about the implementation of the crypto algorithm, having encrypted data and being able to somehow measure the processing time of this data (for example, analyzing the time of sending packets with data), can try to pick up a secret key. The paper describes in detail the tactics of attacks on systems that implement algorithms RSA, Diffie-Hellman and DSS. moreover, the key

can be obtained by specifying bit by bit, and the number of required measurements of time is directly proportional to the length of the key.

Although it has not been possible to bring these studies to a concrete result (calculate the secret key), this example shows that the programming of critical systems (including cryptosystems) should be particularly careful and may need to use special security methods. programming and specialized development tools (especially compilers).

Errors in software implementation

Clearly, as long as programs are written by people, this factor will always be the case. A good example is Novell Netware 3.12, where, despite a well-thought-out authentication system that, according to Novell, "an unencrypted password is never transmitted over the network", SYSCON v. 3.76 was found to have an error where openly falls into one of the network packets.

This is not the case with either earlier or later versions of this program, which makes it possible to speak about a purely programming error. This error occurs only if the supervisor changes the password to someone (including himself). Apparently, somehow the keyboard buffer falls into the network packet.

The presence of hatches

The reasons for the presence of hatches in cryptosystems are obvious: the developer wants to have control over the information processed in his system and leaves for himself the ability to decrypt it without knowing the user's key. It is also possible that they are used for debugging and for some reason are not removed from the final product. Naturally, this sooner or later becomes known to a large number of individuals and the value of such a cryptosystem becomes almost zero. The most famous examples here are the AWARD BIOS (up to version 4.51PG) with its universal password "A WARD SW" and Borland International's Paradox DBMS, also has "SuperPassword", "jIGGAe" and "pbbrrh".

Along with the availability of hatches in the implementation (obviously, in this case, they use explicitly unstable algorithms or store the key together with the data) are adjacent algorithms that allow a third party to read an encrypted message, as is done in a high-profile CLIPPER project, where the third party acts, always fond of stuffing his nose in the secret of his citizens.

Disadvantages of the Random Data Generator (RDG)

A good, mathematically proven and correctly implemented RDG is as important for the cryptosystem as a good, mathematically stable and correct crypto-algorithm, otherwise its disadvantages can affect the overall crypto-stability of the system. In this case, pseudorandom number sensors, typically characterized by a period, scatter, and the need for its seed initiation, are usually used to model the RDG on the computer. The use of PDG for cryptosystems is not generally considered a good solution, so good cryptosystems use physical PDG (special charge) for this purpose, or at least produce a number to initialize PDG using physical values (for example, user key press time).

The short period and the bad scatter are related to the mathematical disadvantages of the RDG and appear if for some reason your own RDG is selected. In other words, choosing your own PDG is as dangerous as choosing your own crypto algorithm.

In the case of a small period (when the pseudorandom values produced by the sensor are less than the possible key values), the attacker can shorten the search time for the key by searching not pseudorandom keys but generating keys from them.

If the sensor is badly scattered, the attacker can also reduce the average search time if the search starts with the most likely pseudorandom numbers.

The most common mistake that can be found in the case of a good PDG is its incorrect initialization. In this case, the number used for initialization has either less number of bits of information than the sensor itself, or is calculated from non-random numbers and can be predicted with varying degrees of probability.

This was the case with Netscape Navigator version 1.1. It initializes the PDG using the current time in seconds (sec) and microseconds (usee), as well as process IDs (pid and ppid). As researchers J. Goldberg and D. Vagner have found out, at such scheme as a maximum 47 significant bits of information (though this sensor was used for 40- or 128 (!) – bit keys) were obtained. But, if the attacker had the ability to intercept packets transmitted over the network and had access (account) to the computer where the program is running, then it did not cause any problems with a high degree of probability to learn sec, pid and ppid. If condition (2) was not satisfied, the attacker could still try to set the time via network time

daemons, the pid could be obtained through the SMTP daemon (usually included in the Message-ID field), a ppid or not very different from the pid, or generally equal to 1.

The researchers wrote the unssl program. which, looking through the microseconds, found a secret 40-bit key in an average of a minute.

Incorrect application of crypto algorithms

This group of reasons leads to the unreliable crypto-stability and correctly implemented algorithms.

Short key length

Its the most obvious reason. The question is: how can stable crypto algorithms have a small key length? Probably due to two factors: some algorithms can work with variable key lengths, providing different cryptocurrency – and it is the developer's job to choose the required length based on the desired cryptocurrency and efficiency. Sometimes, other circumstances, such as export restrictions, impose this desire.

Some algorithms were developed a long time ago when the length of the key used in them was considered more than sufficient to meet the required level of protection.

With a sharp leap in computing performance, the RSA algorithm was first encountered, for which it is necessary to solve the factorization problem. In March 1994, was completed, which lasted for 8 months factorization of the number of 129 digits. To this end, 600 volunteers and 1600 email-connected machines were involved. Machine time was equivalent to approximately 5000 MIPS leagues.

The progress in solving the factorization problem is largely due not only to the growth of computing power, but also to the emergence of new efficient algorithms recently. (The factoring of the next number out of 130 digits took only 500 MIPS years). To date, it's basically a matter of factoring 512-bit numbers. If you mention that such numbers have recently been used in PGP. it can be argued that this is the fastest growing field of cryptography and number theory.

On January 29, 1997, RSA Labs announced a competition to open a symmetric RC5 algorithm. The 40-bit key was revealed 3.5 hours after the start of the contest! (This didn't even require connecting computers over the Internet – enough of a local network of 250 machines at Berkeley University). After 313 hours, a 48-bit key was opened.

Thus, it became obvious to everyone that the length of the key, which would satisfy the export restrictions, could not provide even the minimum reliability.

In parallel with the unveiling of the RC5, a pillar of American cryptography, a DES algorithm with a 56-bit key, was also challenged. And it fell on June 17, 1997, 140 days after the start of the contest (with about 25% of all possible keys tested and about 450 MIPS spent).

It was a remarkable achievement, which meant the actual death of DES as an encryption standard. Indeed, when in the beginning of 1998 the next DES Key Competition came to fruition in just 39 days, the National Institute of Standards (NIST) announced a competition to approve the new Advanced Encryption Standard (AES). The AES must be a fully open symmetric algorithm with a 128, 192, 256 bit key and a 128 bit encryption unit.

Invalid algorithm class selection

This is also a very common reason why a developer chooses a good, but completely inappropriate, algorithm. Most often it is to encrypt instead of hashing or to choose a symmetric algorithm instead of an algorithm with public keys.

For example, it is almost all programs that restrict access to the computer password when it is turned on or loaded, for example, AMI BIOS, which stores instead of the password hash its encrypted version, which, of course, is easily decrypted.

In all network authentication procedures, it is natural to use asymmetric cryptography, which will not allow you to pick up the key, even with full traffic capture. However, such algorithms (from network OS) so far only implement Novell Netware 4.x, others are satisfied (at best!) Standard query-response scheme, in which you can perform a fairly fast search for intercepted values of 'query' and 'feedback'.

Re-imposition of cipher gamma

Already a classic example is the encryption vulnerability in Windows 3.x and the first versions of Windows 95. In this case, Microsoft programmers, well-known for their security knowledge, used the RC4 algorithm (which is nothing like gamma encryption), without changing the gamut, several times to different data – network resources stored in .pwl files.

It turned out that one of the datasets of the .pwl file was more than specific text – a 20-character username (uppercase) and a set of pointers to resources. Thus, guessing the user (which in most cases also matches the file name), you can calculate at least 20 bytes of gamma. Since the gamma does not change when encrypting other resources (this is the main mistake of using RC4 in this case), the first 20 bytes of all resources that include the length of each can be calculated. By calculating the length, you can find the value of pointers and thus add a few tens of bytes to the guessed gamut. This algorithm is implemented in the known program elide.

Storing the key along with the data

This reason leads to the fact that data encrypted using a crypto-stable and correctly implemented algorithm can be easily decrypted. This is due to the specifics of the solved problem, in which it is impossible to enter the key from the outside and it is stored somewhere inside in almost unencrypted form. In other words, the encryption algorithm on the key, and the key (with the help of some secondary key) will be the most vulnerable here. But since (which again obviously follows from the specifics of the task) this secondary key cannot be stored from the outside, the master data will sooner or later be decrypted without the use of methods, iteration.

A typical example here would be all WWW-, ftp-, e-mail clients. The fact is that for basic (most common) authentication in these protocols, the password must be transmitted to the server in an open form. Therefore, client programs are forced to encrypt (not hashed) the password, and with a fixed key, so as not to bother the user with constant questions. It follows that somewhere inside any browser, mail or ftp client (be it Netscape Communicator, Eudora, Outlook, FAR, etc.), all your passwords are stored in a virtually open form, and that deciphering them is no problem. (Most often, by the way, the password in such programs is not even encrypted, and is encoded with a base-64 algorithm).

The human factor

In any critical system, human operator errors are by far the most expensive and widespread. In the case of cryptosystems, unprofessional actions of the user negate the most stable crypto-algorithm and its most correct implementation and application. First of all, it is related to the choice of passwords. Obviously, short or comprehending passwords are easily remembered by the person, but they are much easier to open. Using long and meaningless passwords is definitely better in terms of crypto-

persistence, but a person usually can't memorize and write them on a piece of paper, which then either gets lost or falls into the hands of the attacker.

In recent years, much attention has been paid to resolving this contradiction, but recommendations for choosing good passwords are beyond the scope of this article.

In addition to the fact that inexperienced users usually choose either short or meaningful passwords, there are two methods of their disclosure: full-blown attack and dictionary attack.

Due to the sharp increase in computing power, full-on-attack attacks are much more likely to succeed than before (see also 'Small Key Length'). If crypt(), which is responsible for password hashing, was implemented for UNIX for almost 1 second on a PDP class machine, the speed of its calculation was increased 15,000 times in twenty years (!). Therefore, if earlier hackers (and developers who limited the length of the password to 8 characters) and could not imagine a complete search, today such an attack will on average lead to success in 80 days. Below is the password speed for different cryptosystems.

Full-speed search on a Pentium / 166 computer

Криптосистема	Скорость, паролей/сек.
ARJ 2.50	350 000
RC5 - 56 бит	150 000
LM-хэш	50 000
Novell Netware 3.x	25 000
MS Office 97	15 000
UNIX - crypt()	15 000
RAR 2.0	1 000
UNIX -MD5	500

However, back to a few years ago, when computing power was not enough to completely reset all passwords. However, hackers have come up with a clever method based on the fact that as a password a person selects an existing word or any information about himself or his acquaintances (name, date of birth, etc.). Well, since there are no more than 100,000 words in any language, it will take quite a bit of time to search

them, and 40 to 80% of your existing passwords can be guessed using a simple scheme called "dictionary attack." up to 80% of these passwords can be guessed using a dictionary of only 1000 words!). Even the Morris virus (1988!).

Used this way, especially since UNIX often has a dictionary file on hand, often used by proofreaders. As for "own" passwords, the /etc/passwd file can give a lot of information about the user: his input name, first name, home folder.

The Morris virus has successfully used the following assumptions:

- the input username is taken as the password;
- password is a double repeat of the username;
- same but read from right to left;
- first or last name of the user;
- same but lowercase.

Today, users already understand that it is impossible to choose such passwords, but as long as a person is working with a computer, computer security experts will not wait to use such simple and happy souls of passwords as 34jXs5U @ bTa! 6.

Therefore, even the experienced user cheats and selects such passwords as hopel, userl997, pAsSwOrD, toor, roottoor, password, gfhjkm, asxz. It can be seen that they are usually based on a meaningful word and some simple rule of its transformation: to add a number, to add a year, to translate a letter in another register, to spell the word opposite, to spell the word in Latin, to type Russian a word on a keyboard with a Latin layout, to password with a number of keys located on the keyboard, etc.

Therefore, one should not be surprised if such a "tricky" password will be revealed by hackers – they are not more stupid than the users themselves, and have already inserted into their programs the rules that can be converted words.

In the most advanced programs (John The Ripper Password Cracking Library) these rules can be programmed and set using a special language by the hacker.

Here is an example of the effectiveness of such a search strategy. Many security books suggest choosing a meaningful password for two meaningful words separated by a character, such as "good.password". We calculate how long, on average, such passwords will be cracked if such a rule is included in a cracker program (let the dictionary of 10,000 words,

punctuation marks can be 10 digits and 32 punctuation marks and special characters, Pentium class machine with a speed of 15000 crypt / sec): = 140,000 seconds or less than 1.5 days!

3. Information security in Intranet

3.1 Developing network security policies

Security policy is defined as a set of documented management decisions aimed at protecting information and its associated resources[3].

In developing and implementing it in life, it is advisable to be guided by the following principles:

- inability to pass protective equipment;
- strengthening the weakest link;
- inability to transition to a dangerous state;
- minimizing privileges;
- division of responsibilities;
- separation of defense;
- variety of protective equipment;
- simplicity and controllability of the information system;
- providing general support for security measures.

Let's explain the meaning of the above principles. If an attacker has a disgruntled user with the ability to bypass security, he will, of course, do so. With respect to firewalls, this principle means that all information flows to and from the protected network must pass through the firewall. There should be no "secret" modem or test line inputs that bypass the firewall.

The reliability of any defense is determined by the weakest link. The attacker does not fight against strength, he prefers an easy victory over weakness. Often, the weakest link is not the computer or the program, but the person, and then the problem of information security becomes non-technical.

The principle of inability to move into a dangerous state means that in all circumstances, including freelance, the protective agent either performs its functions completely or completely blocks access. Figuratively speaking, if the strength of the drawbar mechanism breaks down, the bridge must remain elevated, obstructing the passage of the enemy.

[3] URL: http://www.ksu.vntu.edu.ua/files/akit/bakalavr/14_4.pdf

The principle of minimizing privileges requires that users and administrators be granted only the access rights they need to perform their duties.

The principle of division of responsibilities implies such a division of roles and responsibilities, in which one person can not initiate a process critical to the organization. This is especially important to prevent malicious or unqualified system administrator actions.

The principle of separation of the defense dictates not to rely on one defensive line, no matter how reliable it may seem. Physical security means must be followed by software and hardware, access control and, as the last line, logging and auditing. An echelon of defense is capable of at least deterring an attacker, and the presence of a line such as logging and auditing makes it difficult to make a criminal act imperceptible.

The principle of the diversity of protective equipment recommends the organization of different defensive lines in character, so that the potential attacker would require mastering a variety and, if possible, incompatible skills (such as the ability to overcome high fencing and knowledge of the weaknesses of several operating systems).

A very important principle is the simplicity and manageability of the information system as a whole and security in particular. Only for a simple safeguard can it be formally or informally proven correct. Only in a simple and manageable system can you check the consistency of the configuration of the various components and perform centralized administration.

In this regard, it is important to note the integrating role of the Web service, hiding the diversity of objects and providing a single, visual interface. Accordingly, if objects of some kind (say database tables) are accessible through the Web, you must block direct access to them, otherwise the system will be complicated.

The last principle – general support for security measures – is non-technical. If users and / or system administrators consider information security to be superfluous or even hostile, the security mode is deliberately failed. From the outset, a set of measures aimed at ensuring the loyalty of staff, continuous training, theoretical and, most importantly, practical, should be envisaged.

Risk analysis is the most important step in developing a security policy. In assessing the risks to which the Internet system is exposed, the following circumstances should be considered:

- New threats to old services that result from the ability to passively or actively listen to the network. Passive listening means reading the network traffic, while active listening means changing it (theft, duplicate modification of transmitted data). For example, authentication of a remote client using a reusable password cannot be considered reliable in a network environment, regardless of the length of the password;
- New (network) services and associated threats.

As a rule, Internet systems should adhere to the principle of "all that is not allowed, forbidden", since "unnecessary" network service can provide a channel of penetration into the corporate system. In principle, the same view expresses the statement "all incomprehensibly dangerous."

3.2 Software environment security

The idea of networks with so-called active agents, where not only passive but active data (ie programs) are transmitted between computers, is certainly not new. Initially, the goal was to reduce network traffic by performing the bulk of the processing where the data is located (approximating programs to the data). In practice, this meant moving applications to servers. A classic example of implementing this approach is the stored procedures in the DBMS[4].

For Web servers, programs that support the Common Gateway Interface (CGI) are analogous to stored procedures.

CGI procedures are hosted on servers and are commonly used to dynamically generate HTML documents. Organization security policies and procedures should determine who is allowed to connect to the CGI server. Rigid control is necessary here, as executing the wrong program by the server can lead to any serious consequences. A reasonable measure of a technical nature is to minimize the privileges of the user on whose behalf the Web server is running.

In Intranet technology, if you care about the quality and expressive power of the user interface, there is a need to move applications from Web servers to client computers – to create animations, perform semantic controls when entering data, etc. In general, active agents are an integral part of the Internet technology.

[4] URL: https://issuu.com/alex.voronkin/docs/

In whatever direction programs are moved over the network, these actions are of great danger because a program obtained from an unreliable source may contain unintentionally made malicious intentionally generated malicious code.

This program potentially threatens all major aspects of information security:

- accessibility (the program can absorb all available resources);
- integrity (the program can delete corrupted data);
- privacy (the application can read the data and transmit it over the network).

The problem of unreliable programs was recognized for a long time, but apparently only within the programming system.

Java is the first to offer a holistic concept for its solution.

Java offers three defensive lines:

- reliability of language;
- control upon receipt of programs;
- control when executing programs.

However, there is another, very important way to ensure information security – the unprecedented openness of the Java system. The source code of the Java compiler and interpreter is available for verification, so it is likely that honest experts, not malicious users, will be the first to detect errors and shortcomings.

In conceptual terms, the greatest difficulty is the controlled execution of programs downloaded over the network. First of all, it is necessary to determine what actions are considered acceptable for such programs. Considering that Java is a language for writing client parts of applications, one of the basic requirements for which is mobility, the downloaded program can only serve the user interface and to communicate with the server. The program cannot handle the files at least because there may not be any files on the Java terminal. More meaningful actions must be performed on the server side or performed by programs local to the client system.

An interesting approach is offered by Sun Microsystems specialists to ensure the safe execution of batch files. It's about Safe-Tcl (Tool Comman Language). Sun has proposed a so-called cellular command file interpretation model. There is a master interpreter to whom all language capabilities are available.

If you need to execute a questionable batch file while the application is running, a subordinate command interpreter is created that has limited functionality (for example, files and network capabilities can be removed from it).

As a result, potentially dangerous programs find themselves trapped in cells that protect user systems from hostile action. To perform actions that are considered privileged, the slave interpreter may request the principal.

Obviously, here is an analogy with the separation of operating system address space and user processes and the use of recent system calls. This model has been standard for the OS for about 30 years.

3.3 Authentication in Open Networks

Methods used in open networks to validate and validate entities must be robust to passive and active network listening.

Their essence is as follows.

- The subject demonstrates knowledge of the secret key, with the key either not being transmitted over the network or transmitted in encrypted form.
- The subject demonstrates mastery of the software or hardware for generating one-time passwords by means of a request-response mode. It is easy to see that the intruder and subsequent playback of a one-time password to answer the request does not give the attacker.

3.4 The concept of data transmission in open networks

One of the most important tasks is protecting the flow of corporate data transmitted over open networks. Open channels can be securely protected by only one method – cryptographic.

Of course it is natural to put on the firewall the task of encrypting and decrypting corporate traffic on the way to and from the external network. In order for such encryption/decryption to be possible, an initial allocation of keys must take place. Modern cryptographic technologies offer a number of methods for this purpose.

After the firewalls have encrypted the closure of corporate data streams, the territorial location of the network segments is detected only at different rates of exchange with different segments. The rest of the network looks like a whole, and subscribers do not need to bring any additional security.

Simplicity and homogeneity of architecture

The most important aspect of information security is the manageability of the system. Manageability is both the maintenance of high system availability through early detection and troubleshooting, the ability to change hardware and software configurations according to changed or needs, and notification of attempts to breach information security almost in real time, and reducing the number of administration errors, and many, much more.

Simplicity and homogeneity of architecture

The most important aspect of information security is the manageability of the system. Manageability is both the maintenance of high system availability through early detection and troubleshooting, the ability to change hardware and software configurations according to changed or needs, and notification of attempts to breach information security almost in real time, and reducing the number of administration errors, and many, much more.

Internet technology, at the expense of simplicity and homogeneity of architecture, makes the cost of administering a client workplace virtually nil.

It is also important that the replacement and re-commissioning of the client computer can be accomplished very quickly, since these are "clients without status", they have nothing that requires a long-term recovery or configuration.

At the junction of the client and server parts of the Internet-system is a Web-server. This allows for a single mechanism for registering users and granting them access rights, followed by centralized administration. Interaction with numerous heterogeneous services is hidden not only from users, but also to a great extent from the system administrator.

The task of providing information security on the Internet is simpler than in the case of arbitrary distributed systems built in the client / server architecture. The reason is the homogeneity and simplicity of the Internet architecture.

If application developers are able to take full advantage of this advantage, then at the software and technical level, they will be quite a few inexpensive and easy to learn products. However, a thoughtful security policy and a comprehensive set of procedural measures should be added to this.

Some recommendations

A comprehensive approach to information security is needed.

Information security should be considered as an integral part of the overall security of the bank – and as an important and integral part of it. The development of the concept of information security should necessarily involve the management of the bank's security. This concept should not only encompass information technology-related measures (crypto-protection, user rights management software, their identification and authentication, firewalls to protect network I / O, etc.), but also administrative and technical, including rigid procedures for controlling physical access to the automated banking system, as well as means of synchronization and communication between the banking security module and the security system.

It is necessary for the security management staff to participate in the selection-acquisition-development phase of the automated banking system. This participation should not be limited to verification by the supplier. The security management must monitor the availability of appropriate means of differentiating access to information on the system being purchased.

CONCLUSION

Note that encryption and decryption are required in society not by themselves, but only because they can bring profit or avoid losses, so it is always necessary to know what is the value of one character encrypted and decrypt information and what is the cost?

Are the organizations involved in intercepting or decrypting information profitable, or are they deliberately unprofitable?

The most interesting comparative analysis of data to scientifically justify the share of information security costs. It should also be borne in mind that a significant number of attacks are carried out internally by staff from institutions, which are much more difficult to defend against.

In particular, the problem of key storage is currently the most acute and, if using public keys solves the problem of key distribution and user authentication, then a more efficient way of storing keys than memorizing not found, and the use of memorable passwords allows you to apply dictionary attack.

In addition, the use of reliable cryptographic methods does not guarantee protection against software attacks.

Therefore, when creating computer cryptosystems, it is necessary to provide security at the operating system level, which is more difficult than creating a cryptosystem itself.

SUMMARY

There are 4 main groups of reasons for the unreliability of cryptographic systems: the use of unstable algorithms, the incorrect implementation or application of crypto algorithms, as well as the human factor.

This shows a clear parallel between them and the reasons for the breach of security of computer systems.

For the reasons described, there have been or are security concerns in all classes of software using crypto algorithms, be they operating systems; cryptocurrencies; clients and servers that support them; office applications; user-friendly encryption utilities; popular archives.

In order to intelligently implement your own cryptosystem, it is necessary not only to become acquainted with the errors of the Others and to understand the reasons for their occurrence, but also, perhaps, to apply special protective techniques of programming and specialized development tools.

REFERENCES

1. Немет Э., Снайдер Г., Сибасс С., Хейн Г.Р. UNIX: руководство системного администратора. Пер. с англ. Киев: BHV, 1996. 270 с.
2. Гайкович В., Першин А. Безопасность электронных банковских систем. Москва: Единая Европа, 1994. 264 с.
3. URL: http://www.ksu.vntu.edu.ua/files/akit/bakalavr/14_4.pdf
4. URL: https://issuu.com/alex.voronkin/docs/

Information about the author:
Muliava O. M.
Candidate of Physical and Mathematical Sciences, Associate Professor, Deputy Dean of the National University of Food Technology