CHAPTER

3

How Does Computer Security Work?

It's Harder Than It Looks

Cybersecurity is a complex and highly technical subject that uses many tools. It starts with the security provisions built into the chips that power the computer. Software must comply with rules established in hardware and use those rules to enforce more complex security policies. Much of security also depends on the way the software is used. This chapter explains some of the basic tools and principles that are used in secure computing.

Security in Hardware

Computer hardware enforces most security. One or more central processors are at the heart of every computer. The processor manipulates digits, calculating the results of moving and combining digital values. Although the digital values are sequences of zeroes and ones, they are interpreted as numbers or characters and the ways in which they combine can be arithmetic or logical.

M. Waschke, *Personal Cybersecurity*, DOI 10.1007/978-1-4842-2430-4_3

The processor does this by following rules built into the processor chip and following the instructions in the programs that run on the computer. The program instructions are digital values themselves. Instructions are carried out at blazing speed, measured in billions of instructions per second. A 2.4 GHz processor can execute 2.4 billion instructions per second.

The processor in most personal computers is based on the Intel x86 architecture, the architecture of the processor in first IBM PC. Processor architectures determine the instructions the processor will execute and how it controls and moves values from place to place in the processor and memory. The x86 architecture of the first PCs has been expanded to be faster and do more, but the basics have remained the same and most instructions that would execute on the early x86 processors are in use today.

One of the key features of x86 is *protection rings*. Programs are assigned modes, usually called *supervisor* and *user*. These are similar to account types like *administrator* and *user*. A process in supervisor mode operates in what is called Ring 0 and can execute all instructions. A process in user mode, Ring 3, cannot execute some instructions that are called *privileged instructions*. Rings 1 and 2 are for code that interacts directly with hardware and can execute instructions not available in user mode, but fewer than the instructions available in supervisor mode.

The foremost job of protection rings is to keep the system from crashing. A user mode process must not crash the system and destroy the work of other processes. Some program instructions can do this easily when misused. These are the privileged instructions that can only be executed by processes in supervisor mode. Ring 0 supervisor processes are usually part of the operating system. Neither Windows nor Linux use Ring 1 or 2. Applications, the programs that ordinary users execute, are limited to safe instructions and run in user mode, Ring 3. If a Ring 3 process needs to use a privileged instruction, it must ask the operating system to perform it. Protection rings permit ill-conceived Ring 3 applications to execute instructions that may cause their own process to self-destruct or die, but, if the protection rings are doing what they are supposed to, they cannot execute instructions that invade reserved resources or crash the computer.[1]

[1]There are exceptions. The earliest PC x86 processors did not have protection rings. Any MS-DOS program could execute any instruction, which gave DOS programs wonderfully destructive power. When protection rings were added to the x86 architecture, Windows still allowed every program to execute every instruction because some old programs required privileged instructions to work. Since some of these old programs were important to customers, ring protection was not added to Windows when rings appeared. Microsoft engineers added partially effective code to protect the system, but many "blue screen" crashes could not be prevented. These issues were resolved over time. Now Windows uses ring protection and the system is more reliable. Linux never had the backwards compatibility issue because it is based on UNIX, which used ring protection early.

Protection rings also prevent users from interfering with each other. Users cannot affect processes or resources they do not own unless they request privileged instructions. Sometimes an application must execute instructions that could be dangerous to other users. To execute the privileged instructions, the Ring 3 application must ask the Ring 0 operating system to do it. The operating system will examine the authorization of the user and what they are requesting, and then reject unauthorized or dangerous requests. Hackers try to get around these restrictions, and sometimes they do. One way is to feed a program input that will cause a privileged call to crash in such a way that the hacker gets Ring 0 privileges. At that point, the hacker has complete control of the computer for any havoc they care to perpetrate.

The ring mechanism is at the heart of the distinction between administrative and user accounts. As users, taking more privilege than needed is a temptation. An increase in authority over the computer can save time, but it invites dangerous mistakes, and users with unnecessarily high levels of privilege are vulnerable to hackers who want to usurp the power to themselves.

Authentication and authorization are aspects of a system that amplify and rely upon protection rings to provide the finely articulated security we see today. This will be covered in detail in a later section.

Encryption

Encryption converts a readable document to an unreadable one. Decryption converts a previously encrypted document back to the original readable text. Unencrypted text is often called *clear*. Most encryption schemes use a secret key that the receiver of an encrypted document uses to restore the encrypted document to readability. The secret key can take many forms. Sometimes it is a code book that converts words and phrases to other words and phrases. Or it can be a rule like "replace each character with the character four places ahead in the alphabet." In computing, a complex mathematical formula transforms the message to and from the encrypted form. The key is a value that the encryption formula processes with the clear text to generate the encrypted text. The key must be supplied to reverse the formula and return the message to readability.

Encryption is used to protect data in transit between computers. Speedy encryption and decryption is especially important when transferring data. Encryption is also used to protect data at rest, which is usually data that is stored on a disk. Frequently, the encrypted data resides on a remote disk in the cloud.

A good encryption scheme for computers must meet several criteria. First, encrypting and decrypting a message must be fast and efficient; neither encryption or decryption can take too much memory or processor time. Some decrease in computer performance is tolerable for increased security, but no one is happy with sluggish performance. The encryption algorithm must also be hard to crack. Choosing an encryption algorithm is tricky. An algorithm that is slow to decrypt is also slow to crack by the brute force method (trying every possible key until the right one pops out). Thus, it requires a performance and security tradeoff. A rule of thumb is that an encryption scheme is good if the time it takes to break the encryption without the key is longer than the shelf-life of the data it protects, but that is a difficult requirement to meet.

The National Institute of Standards and Technology (NIST) is a part of the United States Department of Commerce, which provides guidance on encryption and other aspects of computer security. It performs intensive testing on encryption schemes and publishes the results. The federal government requires compliance with many NIST recommendations for systems used by the government.[2]

Symmetric vs. Asymmetric

There are three broad types of encryption that are important in computer security: symmetric, asymmetric, and hashes. Hashes are closely related to encryption, but they have somewhat different purposes.

Most people are familiar with the first: symmetric encryption. A single key is used for both encryption and decryption. This is what we ordinarily expect from encryption. See Figure 3-1.

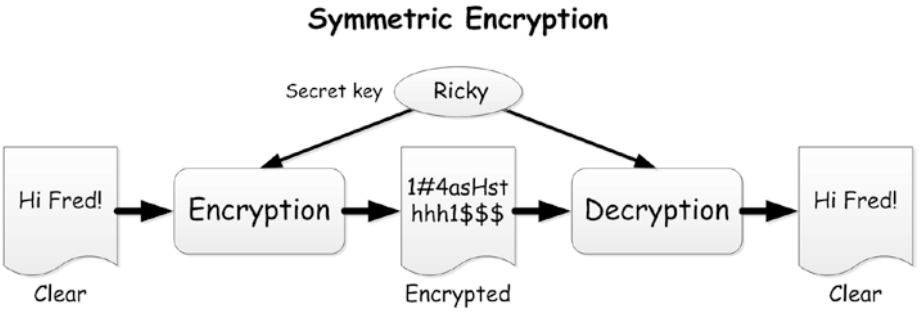

Figure 3-1. Symmetric encryption uses the same key for encrypting and decrypting

[2]See "NIST Cybersecurity Portal," www.nist.gov/cybersecurity-portal.cfm. Accessed February 2016.

For example, naval headquarters and a ship at sea share the secret key. Headquarters encrypts a message with the secret key. The ship receives the message, decodes it with the key, and responds with a message encoded with the same secret key. This system works well when there is a clear sender and receiver who can safely share a secret key. Symmetric encryption/decryption is generally faster than the alternative, asymmetric encryption, so it often used safe transmission and storage of data where speed is important. But other purposes for encryption have other requirements.

Asymmetric encryption uses different keys for encryption and decryption. Asymmetric keys are generated in pairs using special algorithms. These paired keys have the property that a message encrypted using one of the keys can only be decrypted using the other key in the pair. The encrypting key will not decrypt the message it encrypted. Only the other key in the pair will decrypt the message. See Figure 3-2.

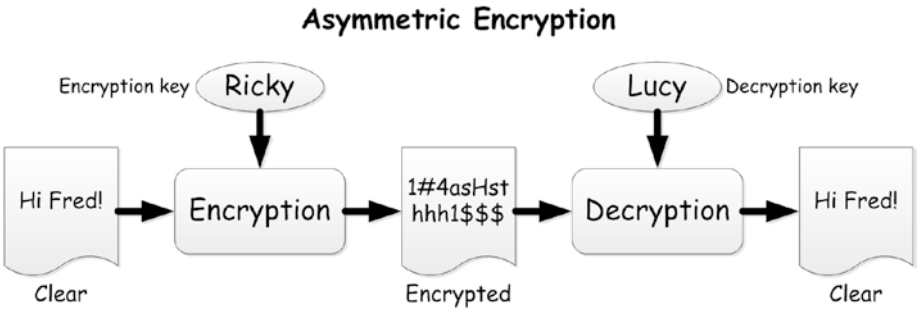

Figure 3-2. Asymmetric encryption uses different keys for encryption and decryption

Requiring separate keys means that sender and receiver do not share a secret key that they both know. Under some circumstances, this is a big advantage.

THE MAGIC OF ASYMMETRIC ENCRYPTION

The algorithms used for asymmetric encryption are complex but the idea behind them is simple. Start with a hard-to-factor number, such as 4757. It is the product of two prime numbers: 67 and 71. We can use these facts to perform an elementary, if trivial, asymmetric encryption.

Ricky wants to send a secret numeric message to Lucy. He can't use symmetric encryption because he has no way to get a shared secret key to Lucy. He goes to an encryption master to set up an asymmetric encryption system. The master gives the key 67 to Ricky. He sends Lucy a postcard bearing the key 71. Ricky may or may not peek at the postcard with Lucy's key; Lucy doesn't know Ricky's key. The encryption master gives Ricky a black box for encrypting numeric messages. Ricky can input

a message and his key to the encryption black box. Inside the box an algorithm multiplies the message by the input key and divides by 4757. The result is the encrypted message. The encryption master gives Lucy a similar box for decrypting. If she inputs the encrypted message and her key to the box, out will come the decrypted message. Inside the black box, the encrypted message is multiplied by the key and the result is the decrypted message.

Ricky's romantic secret message is 11. (I told you this would be trivial.) He uses the encryption box to encrypt the message with his key (67). Inside the encryption box the calculation is

```
11 * 67 / 4757 = 0.154929577464788732394366197183l
```

Ricky's encrypted message is "0.154929577464788732394366197183l" which he paints on a rock and throws through Lucy's window, an insecure transmission, but the message is safely encrypted.

Lucy decrypts the message using the decryption box and her key (71). The result is

```
0.154929577464788732394366197183l * 71 = 11
```

Note that if Ricky forgot the contents of his message and tried to decrypt it with his key, the result would be

```
0.154929577464788732394366197183l * 67 =
                10.380281690140845070422535211268
```

A failure. All Ricky can do with the tools given him by the encryption master is encrypt messages for Lucy. Similarly, Lucy can only decrypt messages from Ricky with her key. Only her key, and no other, will decrypt a message from Ricky.

This is, of course, a weak algorithm and not very useful example, but it illustrates how different keys can be used for encryption and decryption. Non-trivial asymmetric encryption encrypts complex messages and is not easily cracked. Producing the key pairs is a challenging aspect of asymmetric encryption.

Public and Private Keys

Usually, one asymmetric key is designated *public* and the other key designated *private*. Depending on the goal of the encryption, either the encryption key or the decryption key can be designated public. The public key is broadcast to a large group and is not secret. The private key is secret and is kept secret by an individual or special group.

When the encryption key is public, anyone in the public group can use it to encrypt a message, but only a holder of the private decryption key can decrypt the message. Therefore, the members of the public group can send secret messages that can only be decoded by the private group because the decryption key remains secret.

Suppose a detective agency is doing secret work and needs to be careful with communications with their operatives. The agency could issue a public encryption key to all of their operatives. Then their operatives could send private encrypted messages back to the agency, certain that the messages were secret from everyone, even the other public key holding operatives, because the agency holds the only private key that will decrypt the messages. But there's a problem: the agency can't be sure which operative sent the message.

Electronic Signatures

Electronic signatures solve the message source problem. Signatures use asymmetric encryption, but the keys are reversed. The decryption key is the public key and the encryption key is held private.

In the agency example, the operatives each receive a unique private encryption key. The agency makes each operative's public key. Operatives send a message to the agency that is certain to be from them by encrypting the message with their private encryption key. When the message gets back to the agency, they can verify who sent the message by checking whose public key will decrypt it. Now each operative can send messages that are guaranteed to come from them.

There is still a problem. The message sent by the agent is not private because the decryption key is public. Anyone with the agent's public key can read the message. The encryption assures the reader of the message source but not its secrecy. However, the message can be kept secret by asking the agent to encrypt a short text (their name, for example) with their personal private encryption key, insert the encrypted text into the message, and encrypt the entire message with the agency's public key. The privately encrypted text in the larger publicly encrypted text is the agent's signature.

When the message arrives at the agency, the agency uses its private decryption key to decrypt the message. Then the agency verifies the source of the message by decrypting the message with the agent's public decryption key. This way, the agency receives a secret message from a guaranteed source.

Electronic signatures are useful in many ways. Electronically signed software assures users that the software is not pirated or tampered with by a cybercriminal. They are also used to verify that the websites are not fakes set up to trap the unwary. For this purpose, they usually are combined with hashes or digests.

HTTP and HTTPS

Hypertext Transmission Protocol (HTTP) is the protocol that controls messages traveling in the World Wide Web. HTTP is not a secure protocol. A network snooper can easily intercept and read the information in any message. This is a gaping security hole for critical information. The hole was closed by an addition to the HTTP standard, which is called Secure HTTP or HTTPS.

HTTPS encrypts the message and verifies that the receiver of a message is who it appears to be using electronic signatures. The protocol is rather complex. The sender and receiver negotiate the encryption algorithm to be used and exchange encryption keys securely. See Figure 3-3.

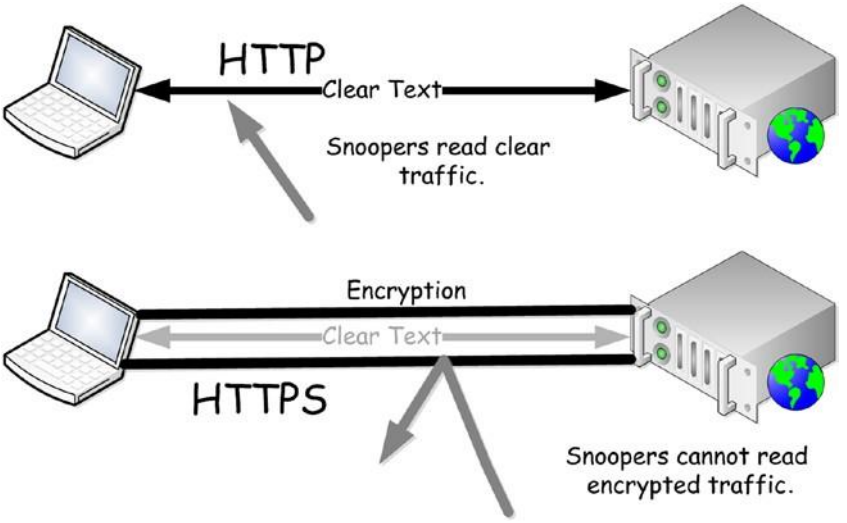

Figure 3-3. HTTPS protects messages with encryption

Hashes and Digests

Hashes are related to encryption, but hashed messages are not intended to be decrypted. In fact, a hash is one-way encryption; a hash that can be decrypted is not useful. A more precise name for the hashes that are used in security are *one-way hashes*. They are also called *digests* or *message digests*.

In general computer programming, a hash is a way of assigning a simple tag for each item in a list. You can hash a list of words by using the first letter of each word as its tag, which is called *the hash*. The hash of "Lupaster" is "L," the hash of "Reggie" is "R," and so on. This is not a very useful hash because it has many collisions. A collision is when two or more values hash to the same thing. For instance, "Apple," "Antique," and "Atom" all hash to "A." Second, it's too easy to reverse this hash because a computer can easily list all the values that hash to "A."

A good cryptographic hash function does the following:

- It is infeasible to reverse.
- It has no collisions.

- Its output values are all the same length.
- The hashed output from two input values that differ only slightly will be drastically different.

Hashes are called *digests* because a hash function can condense a document, even a large document, into a short digest of the original document. Since even a slight change in the original will produce a big change in a good digest, digests can be used to prove that a document has not changed when down- loaded or in transmission. The source sends a digest along with the message. The receiver calculates the digest of the document and compares it to the digest sent by the source. If the two match, the receiver knows the document has not been tampered with or corrupted. When digests and electronic signa- tures are combined, downloaded software is much safer because the receiver is certain that the software received is an exact match to the software sent by the sender who is verified with an electronic signature.

Hashes are also important in verifying passwords, which will be covered in the "Authentication" section.

Authentication

In general, authentication proves that a thing is what it is claimed to be. Authentication in computer security is a special case of general authentication that proves that a person or an entity acting as a person, such as a script, is what they claim to be. Someone enters a user or account name claiming to be the user or account owner and the system challenges the supplicant to prove that the user id is theirs.[3] See Figure 3-4.

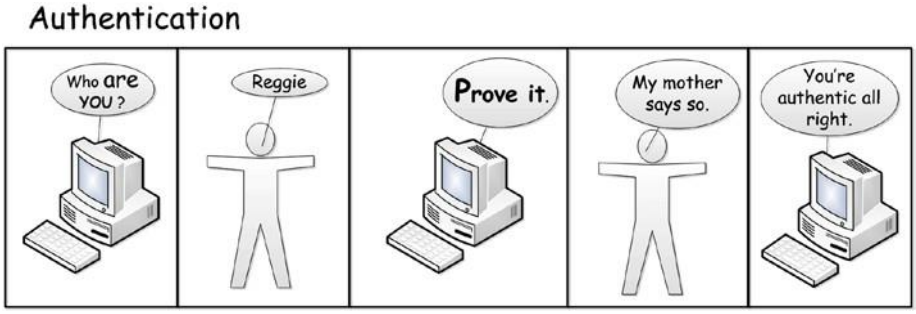

Figure 3-4. Authentication verifies the identity of the user

[3] *Supplicant* is security jargon for a person or agent attempting to authenticate.

In early days of computing, a valid user id was adequate to gain entry to the system. Systems are seldom so trusting any longer. The current equivalent of this approach is a blank password, which is a very weak form of authentication.[4]

Most of the time now, systems will not proceed without something that substantiates a user's claim to a user id. There are three types of substantiation:

- Something the user knows, such as a password or an answer to a security question.
- Something the user possesses, such as a wrist band, a cellphone, a mechanical key, or an email address.
- Some artifact inseparable and unique to the user, such as a fingerprint, retina scan, or face scan.

Several substantiating factors can be combined for increased certainty. Multi-factor authentication is often used for increased security.

Often the substantiation is a password, but not always. It depends on what the system will accept. See Figure 3-4 above. The "Password Alternates" section discusses some of them.

Passwords

Everyone knows about passwords, but some implementations of passwords are better than others. In current security practices, most passwords are a pair: a user id and a secret password. If the password goes with the id, the claimant gets into the system. That's simple.

What happens in between gets complicated. For a simple implementation, all that is necessary is a table stored in a file. Each row in the table has a user id and corresponding password, both in clear text. When a user attempts to authenticate, the system scans the table for the user id that the supplicant entered. If there is no matching user id in the table, the supplicant has no account. If the user id is in the table, the system compares the password presented by the supplicant with the password corresponding to the supplicant's user id in the table. If the two passwords match, the supplicant is allowed in. If the user id isn't present, or the password does not match, the supplicant is not authenticated. There are problems with this simple implementation, although it was used when innocence still prevailed. Later, the password file was encrypted to prevent snooping on passwords.

[4] The cybersecurity team usually offer the keys to the clown car to enterprise users with blank passwords.

Securing Passwords

Most password schemes now use one-way cryptographic hashes rather than encryption. If a password is encrypted rather than hashed, it can be decrypted with the key. If a hacker obtains the key, they can quickly decrypt all the passwords in the password database. A cryptographic hash has no key and cannot be reversed. Encoded passwords never need to be decoded. The system performs a hash when the supplicant offers the password and compares it to the stored hash. Since encoded passwords never need to be decoded, using an encryption with a key is an unnecessary risk. See Figure 3-5 for the entire process. Note that there is a large vulnerability: hashing the password before it is sent to the authenticator may not be practical. More typically, the clear password is sent.You will see in a moment how that gap can be closed.

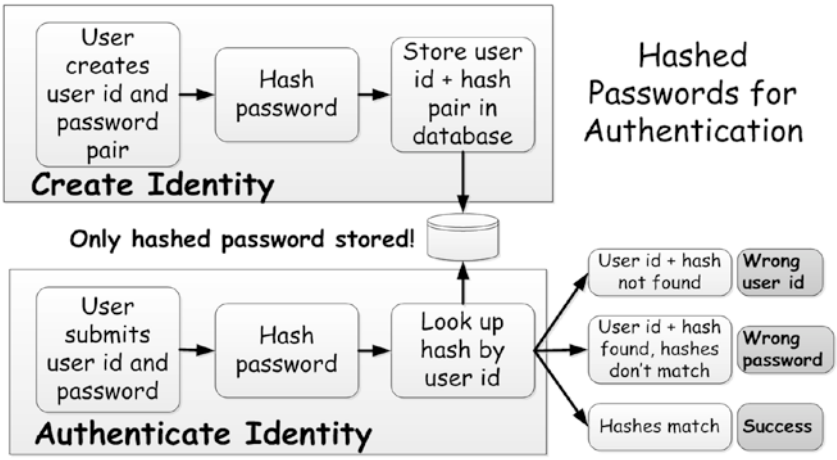

Figure 3-5. Unencoded passwords are never stored in current password practice

How Passwords Are Cracked

Passwords are not invulnerable. They can be compromised in several different ways. Computer users can reduce the chances of password compromise by following good password management practices and being aware of the threats.

Password Theft

Reliable authentication relies on passwords that are kept secret. The system must be hardened against hacker's attacks.The most vulnerable point of attack is the supplicant's practices for keeping passwords secret. If the supplicant can be forced or tricked into revealing a password, authentication is compromised.

As usual, breaching a system with social engineering is easy and quick compared to more technical methods, and very little can be done technically to harden human nature against gullibility and deceit.

Social engineering is not the only way passwords are stolen. Whenever a clear password is placed in an unsecured file, into email or some other insecure messaging system, the password can be stolen by an industrious hacker.

Snooping

Social engineering is not the only way to break a password system. Another point of vulnerability is the connection between the supplicant and the system. If the supplicant is on the same computer as the system, the connection is most likely all in memory and difficult to break into, but a remote login is subject to network snooping. Encrypting the messages interchanged in password validation and creation will prevent network snooping.[5] Secure sockets and secure HTTP are common methods, although some older applications developed their own methods of secure transmission.

Reading unencrypted network traffic is trivial for hackers. Entering a password into a website that does not use encryption is an open invitation for password snoopers. The first version of the HTTP standard documented basic authentication. The method is easy to implement but insecure because it does not protect user id and passwords from snooping in transit. A newer method, digest authentication, uses a cryptographic hash. Since digest authentication is somewhat more difficult to implement, some websites still use basic authentication. Digest authentication is not as secure as using another web security alternative, Secure HTTP (HTTPS). HTTPS encrypts the entire message as well as the password and user id and it also verifies that the message is sent to the intended server. When computers and networks were slower, HTTPS was noticeably slower than HTTP. Websites tended to use HTTPS only when exchanging credentials. This practice has declined as the HTTPS overhead has become less noticeable. Sites like Google and Facebook use HTTPS for all communication on the Web; in fact, HTTPS has become the most common way of hiding passwords from hacker and securing communications on the Web.

---

[5] I have to add that *side channel* assaults are a possibility. They are a sophisticated form of snooping that use external factors like the size and response times of messages to siphon off information about a system. The derived information may be approximate and incomplete, but perhaps enough to stage a breach. See Shou Chen, Rui Wang, XiaoFeng Wang, Kehuan Zhang. "Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow," May 2012. http://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf. Accessed February 2016.

Cryptographic Attacks

Cryptographic attacks assault the encryption or hash algorithm. They come in several forms, but they all require mathematical sophistication. One such attack is called a *collision attack*. Some hash algorithms, which were previously thought to be secure, have proven capable of producing the same hash for two passwords. If a hacker could take advantage of this, they might generate an alternate password for a user id without knowing the real password.

Cryptographic attacks can be more theoretical than real for sites following best practices because they are certainly more difficult than stealing or snooping and could, depending on the circumstances, be slower than brute force cracking. However, there have been sites that continue to use deprecated cryptographic hashes that have been shown to be insecure.

Brute Force

Brute force attacks are more common. A brute force attacker tries different passwords until they find one that works. Some finesse makes the task quicker, but brute force is not subtle. The method assumes that the hash algorithm, user id, and hashed password value are known. Usually the hacker will break into the system by some means and steal the password file or database. Personal computers usually have only a few user id/password pairs to crack and are therefore not as desirable as a corporate server with thousands of ids.

The hacker could simply begin by hashing sequential possibilities like "a","A", "ab", "Ab", "aB", "AB", "AZ" and so on until a hash pops out that matches the user's hashed password. The method is systematic and brute force in the extreme.

Guessing

Hackers know that users have favorite passwords. For instance,"password" is said to be the most used password."1234" is another chestnut. Hackers, and security researchers, compile lists of these. You can get a list of the 10,000 most common passwords hashed with a popular algorithm from the Internet.[6] Hackers use lists like this to skim off the easy candidates. The time for scanning a list like this is trivial (seconds for a smartphone) and hackers tend to favor fast, high-performance servers. Some experts estimate that 60% of passwords at most sites will appear on these lists.

[6]See www.passwordrandom.com/most-popular-passwords. Accessed February 2016. If you think you have a crack-proof password, check here. You may be surprised.

Dictionary

The next level of attack is a *dictionary attack*. A dictionary attack tries every word in a dictionary, and perhaps some common combinations of words. With faster processors, they will probably throw in a few substitutions: "kat" for "cat", random capital letters, anything obvious. For example, "E1eph@nT" might fall to a dictionary attack. For a dictionary attack, the hacker will probably use sophisticated in-memory storage techniques called *rainbow tables* to reduce the memory required and speed processing. Note that mixing uppercase and lowercase letters, numerals, and symbols is not as good a suggestion as it was when cracking machines were less powerful. The number of possibilities to try in a dictionary attack gets large, but it is nothing compared to the huge number of random choices.

Ultimate Force

If a dictionary attack fails, the next stage is true brute force, which is required to crack passwords that are neither common nor in the hacker's dictionary.

At this point, the hacker needs heavy duty resources. A hacker will probably not make this effort without strong motivation because the time and computing resources are expensive. Nevertheless, such an effort is possible.

An ordinary computer is not adequate at this stage. Hackers build special computers that can execute hash algorithms at very high speeds. Graphics processors happen to be well-suited to this. Calculating the next set of pixels to display for a high-resolution, fast-moving subject requires similar capacities to hashing an arbitrary string of characters. Password cracking computers use many graphics processors to process candidate passwords very rapidly. One example, reported on in 2012, uses 25 graphics processors to execute 63 billion guesses per second. If this specialized computer were to test the 10,000 popular passwords referred to above, it would finish in two millionths of a second. An ordinary dictionary attack would still take less than a second.[7]

Figure 3-6 shows an important characteristic of brute force attacks. Brute force cracking of a random password that is not in a popular choice table or subject to a dictionary attack is subject to simple mathematical rules. The table below crunches the numbers for a simple example. The password char- acter set is a common one: digits, lowercase and uppercase letters, and a few special symbols. Altogether, there are 64 possible characters. For a password

[7] Dan Goodin, "25-GPU cluster cracks every standard Windows password in <6 hours," Ars Technica, December 2012. http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/. Accessed February 2016. Note that this reference was three years old when it was accessed. Hardware has advanced in the mean time.

of a single character, there are 64 possibilities. For two character passwords, for each possible first character, there are 64 possible second characters. With each added character, the number of possibilities is multiplied by 64. The number of possibilities rises rapidly. The table shows that a 5-character password has over a trillion possibilities. The number may seem formidable, but a 63 billion guess per second cracking machine can try them all in less than a second.

HOW TO BUILD A CRACKING MACHINE

Today, building a high-speed brute-force cracking machine is not extremely hard. It relies on two technologies. First, big data analysis has developed very effective algorithms for combining the efforts of large numbers of relatively small computers to perform gigantic tasks. The second is the availability of relatively low-cost, high-speed processors that are specifically designed to efficiently perform the mathematical manipulations involved in encryption. These processors are readily available because both encryption and high resolution moving computer graphic images require high-speed, high-throughput numeric processing. In other words, a high-end graphics processor (GPU) is just the thing for cracking passwords. Combining a large number of GPUs into a big data-style parallel processing array for cracking processing is not a job for a first-year computer science student, but by their second or third year, some could handle it. The cost would be in five figures perhaps—but well under the "governments only" price range. It's certainly possible with some help from the "dark side."

Looking down the table, a 7-character password will take a little over a minute, but a 14-character password will take close to ten million years. In short, longer is better. The exceptions are choices from the table of popular choices or the dictionary table. From the table, a 10-character password is quite safe. No hacker would be likely to be willing to wait for 200 days to crack a password. Since cracking machines are bound to get faster, a 12-character password would be a safer choice.[8]

This table brings out an important aspect of password strength. The strength of a password depends on the character set the hacker thinks you are using, not the character set in the password. In Figure 3-6, the length of the password determines the probability that the password will be cracked, not the mixture of characters in the password. If the password does not fall under guessing and dictionary attacks, an all-lowercase 12-character password would still be very time consuming to crack. Stringing together random word combinations into long passwords can be easy-to-remember and strong.

[8]I've skipped over another factor for simplicity. Even though there are trillions of possibilities, there is always a chance that the hacker will win the lottery and get a hit early in its process. The probability of a hit within an interval can be calculated, but I don't want to discuss it here. Longer is still better.

Password Length	Possibilities	Seconds	Minutes	Days	Years
1	64	> 1			
2	4096	> 1			
3	262144	> 1			
4	16777216	> 1			
5	1.07E+09	> 1			
6	6.87E+10	1.1			
7	4.40E+12	69.8	1.2		
8	2.81E+14		74.5	0.1	
9	1.80E+16			3.3	
10	1.15E+18			211.8	0.6
11	7.38E+19				37
12	4.72E+21				2,375
13	3.02E+23				152,018
14	1.93E+25				9,729,155
Character Set: 0-9 a-z A-Z !@#$%& (64 possible characters)					
Password cracking computer processing 63 billion guesses/sec					

Figure 3-6. Password length determines the difficulty of brute force cracking

Death of Passwords

The death of passwords seems to be announced by some important person in the computing industry every few months. For good reason. Strong passwords are a human-computer interaction catastrophe.

Memorizing a 12-character random password, say 2kLc%Arr3$7#, is not easy for most people and accurately typing a random sequence can be frustrating. Not only that, but passwords should be changed frequently in case the password has been stolen through social engineering or snooping.

The result is that most people ignore safer practices. Most passwords in use are on the popular passwords list or subject to dictionary attack, are seldom changed, and are often used for several accounts.

There are alternatives to passwords, but they have their own drawbacks. Possessions, like cellphones, can be lost or stolen. Personal physical artifacts, like fingerprints or face scans, cannot be changed if they are compromised. Fingerprints, for example, have been compromised with casts taken from

fingers. One can easily imagine a method that uses a 3-D printer to convert a photograph of a fingerprint to create an ersatz finger to fool a fingerprint reader. If one print is compromised, the user can substitute a different finger, but there is a limit. Secret questions have been proven to be easily guessed by hackers who do a bit of prying into social media and other public records.

Passwords are a problem but they are also familiar, and techniques for programming password systems are well known. There are also methods built around the password system for sharing authentication.A site does not need to have its own authentication. Instead, it can use authentication provided by another site. Facebook, Google, and other sites provide authentication services.This reduces the number of passwords that a user must manage.

The best alternative to passwords on the horizon now is multi-factor authentication, which continues to use passwords but includes other factors such as a message to your phone containing a PIN you must enter to complete the authentication. Combining passwords with physical artifacts like fingerprints is also more secure.A weaker password combined with other factors can be more secure than a strong password alone.

Authorization

After a user is authenticated, they can be assigned rights to the computer and system resources. Rights could be permission to read or write certain files, or permission to execute programs or use services. All activities on a computer system can be controlled by offering or holding back authorization. See Figure 3-7.

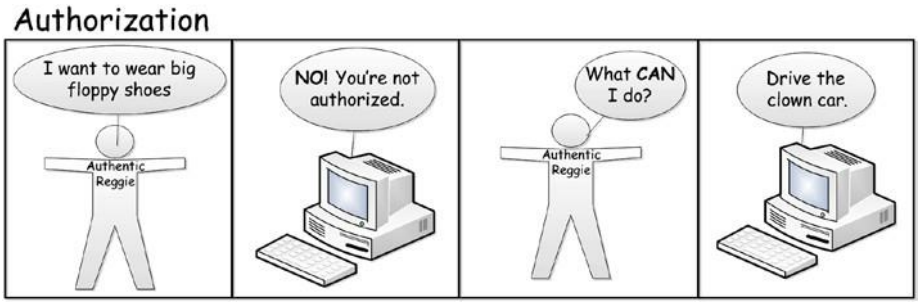

Figure 3-7. Authorization determines what an authenticated person or agent can do. Contrast authorization to authentication in Figure 3-4

Authorization follows authentication, although it may not be obvious that authorization is taking place. Unlike authentication, which is usually based on challenge and response, authorization takes place in the background and is only visible from its effects. Authentication and authorization are often confused or conflated because they seem to take place at the same time.

An enterprise begins by defining a general security policy. The security policy usually specifies what is valuable to the organization and who should be able to access the valuables. Security policies refer to groups and roles rather than individuals and may include non-computer–related valuables such as buildings and manufacturing equipment.

An authorization policy is more specific, indicating specific resources such as file names and processes. System administrators configure hardware and software to enforce the policy. The first step in authorization is to assign authenticated people or agents to groups such as "system administrators" or "human relations staff." When an authenticated member of the human relations staff attempts to read a personnel record, they will be permitted, but they would not be allowed to reboot a critical server.

In Windows, there are two pre-defined groups, "administrator" and "user." When Windows is installed, the first user is placed in the administrator group. This is practical because ordinary users are not authorized to install applications or make most system configuration changes and there are usually many installations and configurations that have to be made when the operating system is installed.

However, good security practice is to authorize users to do their job and nothing more. This practice prevents users from venturing into areas where they may do damage and it applies to a personal computer as much as it applies to a global corporation. The ordinary user group on Windows is adequate for most uses of a personal computer and most of the time, users should not have administrator authorization. The Windows default may not be the best choice.[9]

Personal computer users are not limited to the two default authorization groups. Custom groups can be created with access to specific files and programs. If you have several users that you want to restrict to certain areas, you could create different groups assigned to different account types and give each user a different account type. Logging in with different types can be a way of foiling hackers when they break in and find they are in a limited group without free reign over the system.

Isolation

Thinking back to the Target heist, one of Target's basic problems was that after the hackers got into their system, they had free reign. They entered as facilities subcontractors and made their way into critical financial records.

[9] A word of caution: you must be cautious when changing authorizations. If you end up with no way to log in with administrator privileges, you are in a horrible pickle. Check the documentation, understand it, and proceed carefully.

Secure systems are isolated from the outside world and they are segmented so that an intruder who breaches the outer wall will be limited to a single segment.

Structuring a system like a boat with watertight chambers and a heavy puncture resistant steel hull is one way of limiting the effects of intrusion. If heavy outer hull is punctured and one of the chambers begins to leak, the boat will not sink because the intact chambers will keep the boat floating. Only a portion will be damaged. An enterprise can do the same thing. Each functional area can be sealed off and all interaction with other areas can be controlled and monitored.

In extreme cases, an *air gap* can be established. Two networks with an air gap have no electrical connection between them, neither wired nor wireless. To transfer data from one network to the other, the data has to be placed on a properly sanitized medium like a flash drive[10] and then physically carried to the other network. The storage on a device like a laptop can also be used like a flash drive to move data from one network to another. Perhaps this is not a good practice because a laptop is much harder to sanitize than a flash drive. Other measures include network diodes: hardware and software devices that allow data to move in one direction only.

These practices apply to personal computers as well. A PC owner thinking about segmentation could decide to place confidential documents on a flash drive or an external hard drive and disconnect them when they are not in use. Another example is to have a separate computer that is not connected to the Internet used for confidential work. Some people who are distracted by the Internet do this for other reasons. Thinking about how the system is used and how to segment it to prevent losing everything in an invasion can substantially increase the security of a personal system.

Firewalls

The fundamental purpose of a firewall is to create a perimeter that isolates the interior from attacks from the outside of the firewall.

Perimeters are a fundamental form of isolation and firewalls help enforce perimeters. Think about what a system without a perimeter is like. It would be as if the entire system were built in an open field without a fence or other border around it. Anyone could walk in and look at the system machinery and pull a lever here, turn a valve there, and bring the system to a halt. After a few nasty episodes, the management would probably put a perimeter wall around the system and allow no one in.

[10] Flash drives are sometimes used to spread malware. In circumstances like this, the flash drive should be reformatted to guarantee that malware is removed before it is used. There have been cases in which air gaps have been breached by this method.

Unfortunately, a system inside a wall with no openings may still have problems. Customers and suppliers are stuck outside the wall. The system might be fine, but the business might die. Management and the IT department must think again. What they need is a system that permits some transactions in and out, and blocks others. The rules for permitting and blocking transactions could be complex and difficult to implement, but the programmers claim that is their job.

A firewall is an implementation of rules permitting and blocking transactions. It selectively isolates a system from the outside world. The selectivity of an IT firewall differentiates it from a general firewall like the firewall between the engine and the passenger compartment of an automobile. An automobile firewall blocks all fires, but an IT firewall picks and chooses which fires it will stop.

Core mission of a personal computer firewall is to stop all unsolicited incoming transactions. The firewall will reject any message from a node outside the firewall that is not a response to a request from a computer inside the firewall.

This rule is not as restrictive as it might seem. For instance, a widget on a computer desktop shows the outside temperature. You might expect that whenever the temperature changes, some server outside the firewall sends a message with the new temperature. This is not the case. The widget on the desktop requests a new temperature periodically, which would pass through the firewall without an issue. Whenever possible, interactions with outside resources are designed to request information, not receive it unsolicited.

There are exceptions when a process running on a node outside the fire- wall must send an unsolicited message. These messages will pass through the firewall if the site is on the firewall's whitelist. Messages from a node on the whitelist are always accepted. The other side of the coin is the blacklist. Messages from the blacklist sites are always rejected, even when they are solicited. A blacklisted site could be a known malware source.

Firewalls are a critical element in both personal and enterprise site security. Personal firewalls are usually implemented as software. The Windows firewall is a prominent example, although most of the antivirus vendors also have their own software firewalls.

Enterprises usually rely on firewalls implemented as hardware. The job of a firewall becomes progressively more difficult as the volume of transactions rises. A software firewall can reduce the performance of the PC it protects when the transaction volume is high. When a firewall is protecting hundreds or thousands of users, maintaining satisfactory performance is more than software can provide. Enterprise firewalls are typically hardware appliances that use firmware and dedicated memory to filter incoming and outgoing messages.

Most home computer networks have what could be called a pseudo-firewall that is based on Network Address Translation (NAT). The usual home system has a small router attached to the Internet. All the computers in the house are

connected to the router, either by cables or wirelessly. The router distributes incoming messages to the connected computers. Every computer connected to the Internet has a unique address. The computers in a home network have a different kind of address that cannot be seen on the Internet. The router has its own address that is visible on the Internet and it sends and receives messages from other computers on the Internet. The router does a bit of magic. The home computers send all the messages to the router. The router transforms the messages so they look like they came from the router, and sends the messages on to the Internet. For all appearances, the messages came from the router. When replies arrive, the router sends them to the correct home computer. Only the router is visible to the other computers on the Internet and the home computers are invisible.

This invisibility is helpful, but hackers have found many ways to work around NAT. It does not provide an adequate perimeter for a home system.

Virtualization

Virtualization is another tool for isolation. Virtualization is the simulation of a computer in the memory of a host system. A virtual computer, usually called a *virtual machine* (VM), can be created and destroyed at will. There are many uses for virtual machines because they are easier to manage than a physical machine, especially when they are implemented in such a way that they can migrate from physical machine to physical machine.

Virtualization is the key to cloud implementations. A cloud consumer can request a VM without knowledge of the physical device where the VM is running and the cloud provider can migrate the VM to other devices at will. This permits great flexibility for both the consumer and provider.

A VM running on a computer, often called a *guest*, looks like a computer inside a computer. Install a new operating system on the guest and it is a freshly minted machine. Alternatively, you can install a *snapshot* that you have taken of a machine in a state you want to reproduce and have a duplicate of the machine.

Trying a new operating system can be a problem. If you install the operating system directly on a computer, you have to go to the trouble of restoring from a backup if you want to go back to your old system. A VM solves that problem. Create a VM, install the new operating system, and experiment with it on the VM. If you decide you don't like the operating system, shut down the VM; the operating system is gone, and your computer is unchanged.

A VM can be configured to use memory rather than a hard disk for storage. When the VM is shut down, everything that it has written disappears. Every local record of the guest's activity is gone. If it happened to be infected by malware, the infection is wiped out. If you are concerned about privacy, there is no record.

Malware researchers intentionally infect VMs to study virulent software safely. The researchers close all network ports or turn off the virtual network adapter completely. With no disk and no network connection, the infection can't spread and the researchers can poke and prod, examining the characteristics of the specimen and devising a detection and removal strategy.

Personal users can use a VM as a safe haven for critical tasks like bank transactions by launching a clean VM and performing the transaction from the VM. Malware may have infected your computer without you knowing it, but the chance of malware slipping into the new VM is very slight. When the transaction is complete, you can shut down the VM and it disappears.

VMs also can be used to test install applications that might be infested with malware or to visit questionable websites. You can also open email attachments in a VM when you suspect an attachment is legitimate but you are not sure. Perform the risky action on the VM and delete the VM when you are finished.

Attack Surface Reduction

The attack surface of a system is all the points where a system is vulnerable to attack. An armadillo rolled up into a tight ball showing only leathery plates has minimized its attack surface by hiding its soft and vulnerable belly. Computer users can minimize the attack surface of their computers, perhaps not as efficiently as a three-banded armadillo, but they can isolate themselves from some threats.

Any point where data can move in and out of a system is part of the attack surface. These include

- Network and USB ports
- CD or DVD drives
- SD card slots
- External SATA hard drive ports
- Wireless and Bluetooth radios

Older or newer devices may have other vulnerable connection types. Every one of these connections could be exploited to break into and take over or damage a system. The threats that come in through the network are well-known. Any external storage device, whether a flash drive, an SD card, or an external hard drive, can be a carrier for malware. CDs and DVDs also can contain malware. Wireless network connections are especially vulnerable to intrusion through unsecured public wireless services. Bluetooth may not often be thought of, but it also can be a route into a computer.

The first step to reducing the external connection attack surface is to remove, disconnect, unplug, or turn off any of external connection facility that is not used. The second is to be vigilant over what is plugged into ports. USB flash drives loaded with malware and left in parking lots to attract office workers are a popular method of attack.

Wireless networks and Bluetooth connections are useful and convenient, but turning them off when they are not in use reduces the danger that a miscreant will use them as an entry point.

Other attack surfaces that can be reduced are the programs installed on the computer. Most programs have exploitable defects that may never be known, but there is a risk that a criminal may discover them at any time. By removing all unused software, the attack surfaces of the unused software are also removed.

For example, manufacturers often preinstall utilities for maintaining their computers. These can be problematic because many users don't use them. They are frequently only slight improvements over similar and more familiar utilities that are part of the operating system. These utilities can have flaws that can compromise security.[11] Removing them reduces the attack surface.

Services and daemons are programs that run in the background without a user interface. Many of them are started automatically when the computer boots. These too are attack surfaces that are bound to have exploitable flaws. Quite often, there are unused services set to start automatically. Switching them to start manually or disabling them will reduce the attack surface and occasionally improve performance by freeing resources for more useful processes.

There are also processes or tasks that are started for no understandable reason. These processes may have been left behind by an application that started them but neglected to shut them down. These processes simply wait for input or a termination signal. Hackers are always on the prowl for idle processes running with administrative privileges. Sometimes, a hacker can feed data to the program that will cause it to crash and confer its privileges on the hacker, who is then able to wreak any havoc they care to.

The tasks and services lists reveal the beating heart of the computer more accurately that a cardiologist's echocardiogram. Weeding services and processes from the task list is tough. There is no easy way to spot the rogues. You must become familiar with which processes are legitimate and which are not and a feeling for the CPU time and memory activity that each process should have. If you get it wrong and kill a legitimate task or service, you have programs crashing or losing stability. The entire operating system may crash.

---

[11] For an example, see Kif Leswing, "Another Huge Security Hole Has Been Discovered on Lenovo Computers" http://fortune.com/2015/12/08/lenovo-solution-center-hack/. Accessed February 2016.

On the other hand, a user with a good feel for what should be running and how a normal system behaves on detailed level can reduce their attack surface and keep their system stripped of extraneous performance-sapping activity. By watching the task and services list, Googling process names, and judiciously stopping tasks, a user can become an expert on their own computer. Such a user may be able detect and remediate zero-day attacks before other instrumentation raises an issue.

Damage Reduction and Prevention

Computer users can minimize or prevent damage from successful exploits.

Best practice for enterprises is to maintain a comprehensive information security plan. The plan identifies the elements of the system that are critical to the enterprise. For instance, account documents are critical to most businesses and must be given more attention than office supply inventories. The plan documents

- Who is responsible for the information asset;
- Where and how the critical assets are maintained;
- The enterprise policies, laws, and regulations that may apply;
- How they are secured;
- The steps to take when the asset breached. These steps include who is to be informed and how the system is to be restored.

For large enterprises, the information security plan is a thick document that is revised continually to keep up with changing computing personnel and business practices.

Enterprise security experts often say that an information security plan is the most important tool for keeping IT safe. Personal computer users do not need a formal document like a business should have, but taking an informal inventory of information assets and thinking through security and what to do if a hacker succeeds in getting into the system is a powerful tool.

You may want to answer questions like:

- Are financial records stored on my personal computer? How are they protected from unauthorized access?
- How is my email protected? Where is it stored?
- Do I have data that belongs to my employer on my personal computer? Do I have my employer's permission? Have I secured it according to my employer's rules?

- Do I have valuable personal documents or other files stored on my computer? Are they securely backed up?
- Do I have documents that may be of little monetary value, but are of great value to me? Photographs and videos often fall in this category. Are they properly backed up?

Having answered these questions, you may want to consider investing in some USB drives and storing some of this sensitive information offline. You also might consider investing in a backup system. Also, you might consider whether you are getting worthwhile benefits that balance the risks from storing your employer's data on your computer.

Encryption

Encryption can prevent many types of data theft and intrusion. For example, a hacker may break into a personal computer, but if the computer's files are encrypted, the hacker may not find anything to steal. Encryption has become increasingly important in enterprise computing because workplaces are changing and successful exploits have become more frequent. Employees work from home or places like coffee shops. Encryption has become an important tool for protecting information assets that are outside controlled premises or vulnerable to exploits. These benefits apply to personal computers as well as enterprise systems.

Unfortunately, encryption is surrounded by controversy on exactly how secure it is. Successful encryption requires an entire implementation, not just a strong encryption algorithm. For instance, the Microsoft BitLocker file encryption system, which has been a part of Windows since Vista, combines an encryption algorithm with hardware key generation.[12] In theory, BitLocker is very secure because its algorithm is strong. However, some believe that Microsoft built a backdoor into BitLocker for government agencies. They may have, or they may not. There are vehement partisans on both sides. Some say backdoors for proper law enforcement are good, others say backdoors are always bad. The point is that it is never good to assume that encryption, or any other technology, is infallible. For securing data on a personal computer, perhaps the only thing to say is that even flimsy encryption will defeat an inept hacker, and which is an improvement over no encryption.

[12] See Niels Ferguson, "AES-CBC + Elephant diffuser: A Disk Encryption Algorithm for Windows Vista," http://css.csail.mit.edu/6.858/2012/readings/bitlocker.pdf. Accessed February 2016. This is a paper by the developer of BitLocker from Microsoft, providing a rather technical discussion of the encryption algorithm and hardware used in BitLocker.

Antivirus

Antivirus, or antimalware, tools find and remove malware. They do not prevent viruses and other malware from infecting computers; they detect and remove the malware after the computer has been infected. Antivirus products began to appear in the 1980s, not long after viruses began to appear. A virus is code that reproduces itself and travels to other computers. Other varieties of malware do not reproduce like viruses. Early antivirus products only detected and removed viruses. As other malwares appeared, the antivirus tools expanded to combat them.

The fundamental mechanism used to detect malware is the signature. The signature can be as simple as a file name and size, but they are often more sophisticated, capable of detecting malware that is disguised and hidden. More sophisticated signatures match patterns in compiled code and scan settings found in registries and other configuration files.

Signatures are both the strength and weakness of antivirus tools because the proper signature has to be devised and installed in the tool before the tool can detect and remove a virus. An antivirus tool is helpless against a virus that has no signature because no antivirus team has seen the virus. Malware developers are always busy devising new viruses, which they test against the major antivirus tools. It is a cat-and-mouse game that guarantees that some viruses will go undetected. Nevertheless, the antivirus tools protect against thousands of viruses.

Signatures are not the only way that antivirus tools work. Some use *heuristics*, rules of thumb that identify suspicious processes by their activity rather than a signature. Other methods examine the layout of binary files to identify patterns that are characteristic of viruses. Instead of scanning the computer periodically, some tools monitor the system continually, watching for anomalies as they occur. This is good, but can affect performance.

Antivirus tools have become more effective as automatic updates over the Internet have shortened the time between discovering a virus in the wild and the tool being prepared to detect and remove it. However, hackers have dealt with antivirus tools for a long time and they have become adept at thwarting them. Therefore, antivirus tools are only one line of defense against cybercrime.

Backing Up

Users may not think of backups, saving copies of the data stored on a computer, as a security tool. However, a good backup is the final protection for a compromised computer. With a current backup, a computer can be taken down to bare metal, where every bit and byte of software is removed and the system is cleaned to nothing but the hardware. After the system is thoroughly cleaned, the system can be built up again to its state at the time of the backup.

Although a backup is the ultimate defense, they are not infallible. Some malware buries itself into the firmware in non-volatile memory and crawls out again after the computer is restored. In that case, even the firmware has to be replaced. Also, the unfortunate fact is that backups are only as reliable as the person administering them. Good backup procedure relies on several backups. One should be off-site in a location where it will not be destroyed if a disaster strikes the computer. Backups should not be accessible to ransomware, as they may be on a hard drive. Finally, backups have to be tested regularly. Malfunctioning equipment, such as hard drives, can write unreadable backups, and have a habit of doing so at the moment they are needed most.

The Tools We Will Never Have

Two things are always true about security in general and personal cybersecurity in particular. Criminals will always be eager to break into our computers to steal and do damage. Second, there will always be flaws in our systems, which give the criminals the opportunity to do their malicious deeds.

Computer engineers can build systems that are more resistant to attack. Security has been improving steadily for more than a decade. In 2000, password hashing algorithms were weaker, recommendations for strong passwords were rare, hacking techniques that are now commonly stopped were waiting to be put to use. Home computer firewalls began to appear and have improved steadily. Antivirus tools have improved and are almost universally installed. Security updates are automated and occur, for the most part, silently in the background. Engineers are now aware of coding habits that are likely to leave flaws that hackers can use to break in or cause damage. Quality assurance testing now actively and intensely goes after security holes. These are all real advances that have made computers much more resistant to hacking and are likely to continue to improve security.

Security is much better, but more criminals are striking and plotting to strike every day. And make no mistake, some of the criminals are as skilled and dedicated as the engineers building the computer systems. For every new technique for thwarting attacks that is developed, inspired hackers are searching for ways to break or go around it. And eventually they do succeed, although success is getting harder and harder for them.

Nevertheless, the perfect toolset that will prevent every breach is not on the horizon and is not likely ever to be there. The computing environment itself, offering more services every year and gaining more users daily, continu- ally adds new challenges. The growth in the power and compactness of the computer is always predicted to be close to the end, but each year, they get smaller, cheaper, and more powerful. The global attack surface grows as these tiny powerhouses are used for more purposes that invite exploitation and their complexity increases.

We cannot expect a perfectly secure environment because the dangers and the potential to exploit our devices increases to match the effectiveness of our tools. Nevertheless, the security of our computers has increased. The challenge is to grow our safety faster than the criminals assault us. Fortunately, we have many tools today that we can learn to use more effectively instead of hoping for perfect tools.