

## Chapter 1

# Law 1: Attackers Will Always Find Their Way

*The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair.*

ADAMS D., Mostly Harmless [5]

### 1.1 Examples

Attackers will always find their way. This statement is cruel and pessimistic! Unfortunately, it reflects the brutal bare reality of the world of security. No secure system is invulnerable. This is common knowledge and has been used well in the literature and the movie industry. One well-known Scottish saying is “Three can keep a secret if twa be awa.”<sup>1</sup> This saying highlights that secrets never remain hidden. They will leak out eventually 1 day.

The belief that invulnerability is a myth was already known in antiquity. In his *Iliad*, the Greek author Homer provided an illustration of this law with the Greek mythological hero Achilles [6]. At the birth of Achilles, his mother, the nymph Thetis, bathed him in the Styx, one of Hell’s rivers. Styx’s water had the magical power to make someone invulnerable. No weapon could hurt Achilles. Unfortunately, Thetis had to hold her son by his heel during the bath to immerse him in the Styx. Therefore, a tiny part of Achilles’s body, one heel, remained vulnerable. During the Trojan War, Achilles was the terror of the Trojans as he slaughtered many of them. His most famous victory was against Hector, son of the Trojan King Priam. As vengeance, Hector’s brother, Paris, killed Achilles with an arrow guided by the God Apollo to the unique, vulnerable spot of Achilles’s heel. Interestingly, the coward Trojan Paris defeated the mighty Greek warrior with the help of an insider: Apollo. Indeed, only a God could know the existence of this vulnerability. This classical scheme implies an insider is commonly encountered in the security arena. An insider, here the God

---

<sup>1</sup>Three can keep a secret if two are away (sometimes also found as “Three can keep a secret if two are dead”).

Apollo, will reveal the lethal secret or will partake in the plot of the attacker. Section 4.3.5 will explore this important scheme.

This myth of one single point of vulnerability appears in many other mythologies, such as with Sigurd in Norse mythology.<sup>2</sup>

Before the era of information technology (IT), many historical events demonstrated the truth of this law. The sinking of RMS Titanic is an iconic example. In 1912, the British company White Star Line built the largest cruiser of its time. Not only was RMS Titanic the largest ship, but she was also claimed to be the safest in naval history. White Star Line declared that Titanic was designed to be unsinkable with her sixteen watertight compartments. Her maiden journey from Southampton to New York began on April 10, 1912. During the night of April 14, 1912, the Titanic struck a colossal iceberg. The supposedly unsinkable ship sank in less than 3 h. Of the 2224 passengers and crewmembers, only 722 survived. The ship did indeed have sixteen watertight compartments. The purpose of watertight compartments is to isolate a section of the boat's hull from the rest of the vessel. If this part of the hull were to be ripped open, then water would fill only this compartment. The flooding would not reduce the boat's buoyancy significantly. Thus, at least in theory, sixteen watertight compartments should have prevented the ship from sinking. Unfortunately, the iceberg ripped off more compartments than foreseen by the naval designers. Therefore, the incoming quantity of water exceeded the buoyancy threshold, and the Titanic sank, like an ordinary boat.

### **Rule 1.1: Always Expect the Attacker to Push the Limits**

The attacker never obeys the rules; nor does she comply with the behavior expected by the designer of the system. A common method of attack is to take the system out of its nominal cases with the hope either of provoking a failure or of gaining some advantages. A buffer overflow attack is such an example, where the attacker inputs information longer than expected. Fault injection attacks are another example, where the attacker changes the operating environment (Sect. 1.2.2) so that the system cannot operate properly anymore.

In 2014, Joseph Hemanth designed a simple denial of service (DoS) attack on the Pebble, a smartwatch [7]. Each time the Pebble receives a notification from the email service, Twitter, or any social network, it vibrates and displays the corresponding notification message on its small screen. Unfortunately, the Pebble does not check the length of the message that it displays. If an attacker sends a few hundred emails or Facebook notifications in five seconds to the targeted user, then the display buffer overflows. The display of the targeted user's Pebble becomes dull with many white lines. After a while, the overflowed Pebble resets to factory defaults, deleting all configuration settings and stored messages. The attack does

---

<sup>2</sup>Sigurd killed the dragon Fafnir. Following the advice of the God Odin, he bathed in its blood to become invulnerable. Unfortunately, a leaf sticking on his shoulder created a weak point in this otherwise armored skin. Of course, his opponent, Gottrum, would defeat him through this unique vulnerable point.

not only DoS the device, but also wipes out the targeted device. Hemanth pushed the limits of the Pebble.

The number of examples that could illustrate this first law is vast. Every day, new exploits occur. The book will present many other illustrations.

## 1.2 Analysis

There is a strong asymmetry in this war between security defenders and attackers. The attacker has many advantages over the defender. First, the attacker needs to succeed only once, whereas the defender has to succeed every time. The attacker can afford many defeats. The defender cannot afford one. Furthermore, the attacker may reuse the winning strategy or technique against many different defenders. Second, the attacker benefits from all the security technologies and tools that the defender may use. She can twist these technologies to become weapons rather than armors. Furthermore, the attacker does not share the same constraints as the defender. She does not have to obey a bureaucracy, follow processes, and generally obey rules. She has the freedom to act outside of the established context and rules. Furthermore, nature favors the attacker. The second law of thermodynamics states that entropy tends not to decrease. It highlights that it is easier to break a system than to build it. Building a system increases the order and decreases chaos. Therefore, it reduces entropy. Breaking a system increases the chaos and thus increases entropy. The remainder of this chapter illustrates this sad reality.

### 1.2.1 *Should Vulnerabilities Be Published?*

These anecdotes may seem far from the IT world. Unfortunately, the current information era has a plethora of failure stories. The first example is from World War II (WWII): Enigma M4. For many centuries, nations have used some form of cryptography to protect their vital communication. Even Julius Caesar encrypted his messages.<sup>3</sup> During WWII, every country used its own proprietary encryption

---

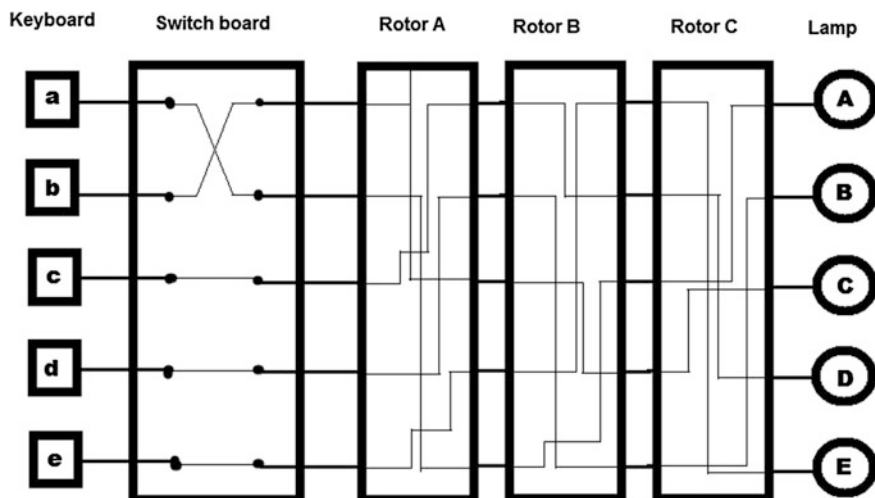
<sup>3</sup>Julius Caesar's substitution algorithm was extremely rudimentary. The encrypted character is the original character shifted by a fixed value within the alphabet. For instance, if the shift value is 3, A becomes D, B becomes E, ..., and X becomes A. With this key, "WHQ ODZV IRU VHFUXULWB" is the cipher text of "TEN LAWS FOR SECURITY." Obviously, this type of encryption can be easily broken. The easiest method is to make a statistical analysis of the frequency of occurrence of the encrypted characters in the cipher text and then try to match their distribution with the Gaussian distribution of the supposed language. For instance, in English, the three most frequent characters are E, T, and A, whereas in French they are E, S, and A. The analysis is even more efficient when using pairs of characters or groups of three characters. If the encrypted message is long enough, then the identification of these most frequent characters is easy. The correspondence reveals the "key" and thus the original message. Al Kindi introduced this statistical method for cryptanalysis in the ninth century [8].



**Fig. 1.1** Four-rotor Enigma machine

algorithm. Of course, each camp attempted, often successfully, to break the enemy's cipher [9]. They used cipher machines to encrypt and decrypt their confidential messages. Usually, these cipher machines were rotor-based. The Swiss Hagelin was the most renowned manufacturer of these cipher machines [10]. The best-known cipher machine is Enigma. Germans were inspired by the Hagelin machines to design their military, version Enigma G, by adding their own proprietary enhancements. Thus, they produced the Enigma M3 and later the Enigma M4 (with a fourth supplemental rotor), as illustrated in Fig. 1.1.

An Enigma machine looks like a typewriter with an extra set of rotors and switchboards. The rotors and switchboards serve for setting the secret key. The keyboard is used to enter the clear text to encrypt, or the ciphertext to decrypt. The entered text has no punctuation or spacing. Letters replace numbers. The nine letters



**Fig. 1.2** Simplified view of Enigma M3

at the top (Q, W, E, R, T, Z, U, I, and O) also represent nine digits. 0 is associated with letter P. Therefore, Q may mean the letter Q or the digit 1. Lightbulbs, one per letter, display the decrypted or encrypted result. Figure 1.2 depicts a simplified Enigma machine using only five letters and three rotors. The real devices use the full 26-character Latin alphabet, no punctuation, no digits, and no space. By convention, the input has uppercase characters, whereas the output has lowercase characters. Each rotor has 26 entry points physically wired to 26 exit points (one point per character). Rotors A, B, and C have different wiring. Of course, the rotors are identical in Enigma machines of the same model. The physical connection lights the corresponding letter's bulb. Rotor A moves one position each time the operator dials a new character. After a full rotation of rotor A, i.e., after 26 characters, rotor B moves one position. After a full rotation of rotor B, rotor C moves one position. The switchboard allows a fixed reshuffling of the keyboard. To decrypt a message correctly, the operator has to reproduce the initial conditions of the encrypting machine, i.e., the same configuration of the switchboard and the same initial position of the rotors. The secret key was the combination of the switchboard's configuration and the rotors' position.

During the war, Germans believed that no enemy could break their cipher. Most countries accepted this statement, and not many nations tried to challenge this assumption. However, in 1932, Polish cryptanalysts had the conviction that mathematics could be used to solve the problem. Through reverse engineering, they had already understood the particular wiring of the four rotors. They developed an electromechanical device, called the Bomba, which allowed accelerating the code breaking. With the help of these tools, they found two crucial design flaws in Enigma. Nevertheless, continuous improvements of Enigma kept it ahead in the

ensuing race. In 1939, when Germans invaded Poland, the Polish cryptanalysts sent all their work, with a few replicas of Enigma machine, to their British and French counterparts. This contribution would prove to be invaluable.

In 1940, the team established at Bletchley Park succeeded in breaking the cipher. Under the lead of Alan Turing and Gordon Welchman, the team partly automated the cracking process using their own electromechanical device: Colossus. The German forces continuously improved their Enigma machines, making each version harder to crack than the previous one. For instance, in 1942, the German naval forces deployed the four-rotor Enigma M4 on their submarines (called U-boats) in replacement of the three-rotor version. Nevertheless, during most of the WWII, Bletchley Park was able to decrypt German ciphertexts. This success seriously influenced the outcome of WWII. Some historians estimate that it reduced the duration of the war by 3 years and saved thousands of lives. For instance, the German Air Force lost the Battle of England in 1940 partly because it entrusted bomber-target information to the insecure Air Force Enigma [11]. Thanks to Bletchley Park, the British Royal Air Force knew in advance the targets that German bombardiers would bomb during the coming night raid. The German Navy lost a staggering number of submarines for the same reason because Bletchley Park could break German Navy Enigma-encrypted messages.

The way the Allied forces treated the decrypted messages is interesting. The decrypted messages were called Ultras. The handling of Ultras was restricted to a limited set of dedicated, trained, trusted officers. The German could never become aware that the Allies had cracked their Enigma codes. Thus, the exploitation of information collected from Ultras was extremely sensitive and tightly controlled. The Allied high command decided that military operations should never exploit Ultras unless there was another potential source of disclosure of the corresponding information. Because of this clever strategy, the German forces never discovered that the Allied forces could read their secret encrypted communications. Despite its losing many U-boats, the German naval high command always thought that its losses were due to the Allies' better radar and direction capabilities rather than a weak cipher. The Germans systematically incriminated the second source of information rather than questioning the secrecy of their communication.

The design of a robust cryptographic system requires, at least, two different skill sets: cryptography and cryptanalysis. Cryptography, practiced by cryptologists, is the art of designing the actual algorithms, whereas cryptanalysis, practiced by cryptanalysts, is the art of breaking the designed algorithms. Cryptology encompasses both cryptography and cryptanalysis. The real robustness of the encryption depends on the quality of both teams. A skilled cryptographer may develop a suitable algorithm. Nevertheless, only skilled cryptanalysts can give reasonable assurance that this algorithm is robust. According to American military experts, the German cryptographic systems were brilliantly conceived [11]. German cryptologists were brilliant. Unfortunately, the Allied cryptanalysts were better than the German cryptanalysts. At the end of the war, the German forces had new encryption devices ready for rollout. The Allied cryptanalysts would not have been able to crack them. Fortunately, until the end of the war, the German forces were always

convinced that Enigma was robust, and they did not see the need to deploy the most robust version. Hence, the special treatment of Ultras was a winning strategy.

Unfortunately, this strategy is not a good one anymore. The designer of a secure system must request other skilled people to attack his system. Furthermore, it is essential to look continuously for evidence that the system can be eventually broken. This search of proof requires monitoring the Internet, dedicated forums such as Doom9 [12], security contests such as Pwn2Own, and hacking conferences such as Defcon or Black Hat. This continuous monitoring may reveal attacks applied to another system but also applicable to yours. Having researchers and ethical hackers publish exploits is essential to the process of designing more secure systems. This practice is called full disclosure policy. Some people claim that publishing vulnerabilities benefits attackers and is dangerous for consumers, and so disclosing vulnerabilities should not be allowed. According to them, software vulnerabilities should be kept secret. Indeed, once a vulnerability is publicly disclosed, the number of attacks based on this vulnerability may multiply by up to five orders of magnitude [13]. Jason Lanier, one of the fathers of virtual reality, called the full disclosure policy the ideology of violation [14]. According to him, the only purpose of the researchers was to seek glory. I disagree. I believe that refusing to disclose vulnerabilities is a poor strategy. Indeed, publishing exploits has three positive consequences.

- *Disclosure of the Vulnerability:* The designers of the vulnerable system can now implement a countermeasure that will close the hole. That a given vulnerability has not been publicly revealed does not mean that an attacker is not already exploiting it. Maybe an attacker already knows and exploits this vulnerability but keeps this knowledge to herself. In that case, this stealthy attacker may exploit the vulnerability without her victims being aware and with no chance of seeing this hole fixed.

An unknown vulnerability is called a 0-day vulnerability. 0-day vulnerabilities are the quintessence of vulnerability for attackers, as there is yet no available defense. Thus, the attack will be taking place off the radar. Therefore, there is a black market for trading 0-day vulnerabilities. Their price is ranging from \$5000 to \$250,000 depending on the criticality of the exploit [15, 16]. Furthermore, it is claimed that some private companies, such as ReVuln [17], Vupen Security, FinFisher, and the Hacking Team, purchase such 0-day vulnerabilities from hackers. Later, these enterprises sell these vulnerabilities to interested customers [18]. The sale of these vulnerabilities is secret to give a competitive edge to the customer. The potential clients of such 0-day vulnerabilities are private corporate teams looking for new attack vectors or attempting to set up an advanced persistent threat (APT), as well as government agencies. For instance, in 2012, the US National Security Agency (NSA) paid Vupen Security for a 1-year supply of 0-day vulnerabilities. Vupen Security provided both the vulnerabilities and the pieces of software needed to exploit those security flaws to attack electronic systems [19].

Interestingly, many companies may reward the private disclosure of vulnerabilities in their products or services [20]. Since November 2010, Google fostered responsible disclosure through its vulnerability reward program [21]. The reward ranges from \$100 for a new cross-site scripting (XSS) to \$20,000 for unknown remote code execution. As an extremely smart incentive, it also publishes the name of the contributors in its security hall of fame. In September 2012, Google had rewarded about 250 persons. On June 26, 2013, Microsoft launched a similar program, called “mitigation bypass and blue hat defense,” with bounties of up to \$100,000 [22]. In January 2014, Facebook paid a \$33,500 bounty to a Brazilian security researcher who found one remote code vulnerability [23]. These bounty programs have started to become popular, well attended, and well funded. In 2013, Facebook received 14,763 submissions. This number was an increase of about 250 % compared to the previous year [24]. Among them, only 687 submissions were valid vulnerabilities. About 40 of them were critical vulnerabilities. Google went even one step further. In 2015, Google launched the Vulnerability Research Grant program [25]. Researchers may apply for this grant program to look for potential vulnerabilities within sensitive Google applications. The grant ranges from \$500 to \$3133.70. The application clearly states that some researchers might not find any vulnerability. In other words, the grant is not contingent on the discovery of any actual vulnerability. Many forms of reward are in use. For example, United Airlines rewards the vulnerability discoverers with air miles [26].

Nevertheless, the bounty is not necessarily the primary driver of the disclosure, as many ethical hackers are motivated by fame rather than by monetary reward. Reputation is a powerful motivation. The Web site HackerOne.com is a hub for about 100 of such vulnerability discovering programs. The programs range from large companies or projects (Adobe, OpenSSL, Python, Twitter, or Yahoo) to small sites or projects. A little less than half of these projects offer pecuniary rewards. Furthermore, the site provides a federated hall of fame. Some security-oriented conferences also organize contests. For instance, the annual Pwn2Own contest is an organized challenge to crack the security of the four top browsers.<sup>4</sup>

Of course, responsible disclosure requires careful scheduling. Known, published, responsible disclosure policies exist, such as the Coordinated Vulnerability Disclosure [27] or the Zero Day Initiative [28]. Without such policies, the disclosure of vulnerabilities would subscribe to the ideology of violation. The ethical hacker must first disclose the vulnerability to the product owner/developer. The product owner should negotiate a reasonable delay with the ethical hacker or security researcher before the public disclosure. During this grace period, the product owner will fix the vulnerability and provide a security patch to its customers. Then, customers must apply the security patch, which

---

<sup>4</sup>In 2015, Jung Hoon Lee was rewarded \$225,000 for three successful exploits on Chrome, Safari, and Internet Explorer 11.



may take many months. A recent study evaluated the average duration of one 0-day vulnerability in the field to 312 days. In extreme cases, it lasted up to 30 months [13]. Later, the ethical hacker publishes the exploit. Unfortunately, as exploits are often described immediately after the delivery of the corresponding patch, and as people do not immediately update their systems, a flurry of exploits may be launched during this transitional period [29]. Late in 2014, with its Project Zero, Google unilaterally decided that its researchers should disclose within 7 days the critical vulnerabilities they discovered [30] and that less critical vulnerabilities should be disclosed within 90 days. The claimed objective was to accelerate the availability of a patch or, at least, the publication of a security advisory bulletin. After some hiccups, Google added a potential grace period of 14 days if the software editor needed it to complete the security patch [31].

- *Availability of the Patch for Vulnerability:* Public disclosure gives a strong incentive to the software editor to fix the vulnerability. Avoiding bad mouthing that may ruin the reputation of a firm can be a powerful inducement for developing a new security patch. The more secure the deployed pieces of software or the Web services will be, the more secure the digital world will become [32]. Having all systems patched correctly is of general public interest. See Rule 6.2: Patch, Patch, Patch.
- *An Excellent Educational Tool for Students and Especially for Practitioners:* A new vulnerability may use a new type of attack and technique. It is hard to defend against an attack that you are not aware of. Knowledge is paramount to security. The disclosed attack may apply to targets other than the original one. Designers of such potential targets can preventively address these vulnerabilities. They do not need to wait until the exploits are deployed on these targets. Furthermore, designers can avoid the same mistakes in new designs.

### **Rule 1.2: Responsibly Publish Vulnerabilities**

In the role of dissemination and education, the Computer Emergency Response Teams (CERTs) are key players. CERTs are primarily dedicated to helping enterprises and administrations rather than end users. Nevertheless, most of the time, the CERT information is available to everyone. Their primary missions are to analyze all reported threats, to maintain a database of all disclosed vulnerabilities, and to provide support and advice for implementing mitigation plans for these vulnerabilities. The first CERT was created in 1988 in Pittsburgh at Carnegie Mellon University under the lead of the US Defense Advanced Research Projects Agency (DARPA) following the appearance of the Morris worm. As CERT is a trademark of Carnegie Mellon University, the community should instead use the generic term Computer Security Incident Response Team (CSIRT). There are several hundreds of CSIRTs in the world. The CERT from Carnegie Mellon University remains the most famous CSIRT.

### 1.2.2 Jailbreaking and Secure Bootloaders

During World War II, the cryptanalysts at Bletchley Park succeeded in breaking the algorithms that the Enigma machines implemented. The next example describes another hack applied to a different type of device. This time, it is a consumer device rather than a military apparatus. Many modern devices execute only pieces of software approved by the device's manufacturer. This is the case, for instance, for game consoles, set-top boxes (STBs), and some smartphones. There are two main rationales for this design decision.

- **Security:** Controlling what software executes on a platform is a way to ensure that no malware contaminates the platform. This method is especially efficient for closed garden environments such as STBs and even smart cards. It ensures that the device behaves as expected or that no forged piece of software can extract secrets or misbehave. In the case of game consoles, it is a theoretical, copy protection method, as only signed, controlled pieces of software can execute on the platform. If usual ripping tools cannot make a pristine, complete copy of the genuine game, then it is not possible to produce illegal copies of genuine game. Usually, the corresponding physical media uses a logical format that is different from the format used by the media industry for movies, in order to thwart ripping tools. For instance, the logical format of a game DVD is different from the format of DVD-Video. Furthermore, it uses a proprietary format, whereas DVD-Video's one is public.
- **Business:** Enforcing what software executes on a platform is a way to control how the market distributes the corresponding software. The manufacturer is a gatekeeper who may charge a fee to the software editors for authorizing the execution of their applications on its platform. This is the case, for instance, with Apple with its Apple Store, or the manufacturers of game consoles. They create controlled vertical markets.

Whatever the reason to implement such restriction methods, these systems use the same enforcement mechanism: a secure bootloader. The secure bootloader is quasi the first piece of software executed by the device. It has two functions.

- The secure bootloader checks the integrity of the system. It typically verifies whether the operating system and the drivers have been tampered with. For that purpose, it compares the hash value<sup>5</sup> of the binary code of these elements with the corresponding stored reference values. Obviously, the stored reference hash values should not be rewritable by an attacker. Why use a hash function rather than a lightweight checksum, given the fact that the hash function requires much more calculation? Under the assumption that the stored reference hash values cannot be modified, if the attacker changes even one bit of the software's binary code, the corresponding hash value changes drastically. The attacker will not be

---

<sup>5</sup>Section 15.3 provides an explanation of the hash function.

able to forge a proper piece of code whose corresponding hash value will match the reference value. This feature is intrinsic to the hash function. With a checksum, it is easy to forge a piece of binary code that will present a corresponding checksum by appending a few dumb bytes. If the attacker modifies one bit of the software, she has just to adjust one of the dumb bytes to obtain an invariant checksum.

- This solution is applicable only to applications and drivers that the bootloader knows before runtime. For dynamically loaded applications, thus unknown to the bootloader beforehand, integrity checking needs another mechanism. Section 4.2.5 describes a potential solution to this problem.
- Once the integrity of the system is established, the bootloader checks the integrity and the origin of the piece of software to be executed. Usually, it verifies that the originator cryptographically signed this piece of software (Sect. 4.3.6). Of course, the bootloader has to store the public, signing, root key of the expected originator securely. This public root key has not to be secret. Nevertheless, it must not be rewritable by the attacker. If an attacker could change this public root key, then she could replace it with her own public root key and so execute arbitrary code signed by her corresponding private root key. It is beneficial to note that the signature empowers two verifications. The first verification is that nobody tampered with the software package. The second check is whether the authority who knows the private, signing, root key issued this software package. In other words, checking the signature verifies both the integrity and the origin of the piece of software before executing it.

Sometimes, the bootloader is more complex and has several stages. The two initial stages are the ones described above. The following stage loads a signed middleware. This middleware will load the dynamic application once it successfully checks that the signature of the application is valid. In this configuration, it is good practice to sign the dynamic applications with a different key hierarchy than the one used to sign the middleware. This differentiation allows distributing application-signing keys to external, trusted, software developers while keeping control and limiting the risk of forgery of its proprietary middleware. With such a key distribution mechanism, only in-house developers have access to the process that signs the middleware.

The secure bootloader should be a tamper-resistant, carefully crafted, piece of software. Ideally, to prevent any modification, the secure bootloader should be stored in a non-writable memory. With such a secure bootloader, it seems impossible for an attacker to execute an arbitrary piece of software. Unfortunately, Law 1: Attackers Will Always Find Their Way is always true. Attackers sometimes find a way to unlock such devices. This kind of attack is often called jailbreaking. The objective is to find a vulnerability that grants root access to the device. With these elevated privileges, the attacker can modify the secure loader so that it accepts any binary code, even if not signed by the manufacturer [33]. The next example examines such a jailbreaking exploit.

**The Devil Is in the Detail** A very sophisticated secure bootloader protects Microsoft's game console: Xbox 360 [34]. It uses a three-stage system with the initial bootloader being signed by RSA and being encrypted with RC4. The bootloader securely launches a hypervisor. The hypervisor starts the signed games once their signature is verified. Some early versions of this hypervisor had flaws that allowed executing unsigned code. Microsoft systematically issued a newly revised version of the hypervisor that was immune to the weakness. Microsoft forced the update of Xbox 360 to enforce the systematic use of the most current hypervisor that is not crippled with vulnerability. Every known hack of Xbox 360 uses one of the flawed hypervisors. The enforcement of the use of immune hypervisors should prevent the existing attacks. Thus, the goal of the attacker is to launch a defective hypervisor instead of a flawless new one. The bootloader checks whether the signature of the hypervisor is legitimate. The flawed hypervisors are genuine Microsoft pieces of software. Therefore, they are duly signed. If the bootloader would only verify the signature, it would not ban the defective hypervisors. Therefore, Microsoft modified the bootloader to check whether it attempts to execute one of these flawed hypervisors. In addition to validating the signature, it calculates the hash of the hypervisor and checks whether the calculated hash belongs to a blacklist of forbidden hashes. Hackers must bypass this additional control to install a flawed hypervisor.

Gligli and four hacking colleagues implemented a hardware attack based on hardware fault injection. Fault injection is a sophisticated but devastating class of attack [35]. The objective of fault injection is to make a piece of software act in a different way than expected by its designers. The source of the fault is external stimuli in the environment of the hardware component. Typical stimuli can be glitches in the external clock signal, variations in the voltage supply, tiny pulses in the reset signal, low temperature, and even X-rays and ion beams. These attacks are not intrusive, as they do not require modifying the piece of software or using a debugger or injecting signals inside the chip.<sup>6</sup> The stimulus should generate a fault in the processor, resulting in the program execution's derailing. It does not anymore follow the expected flow of execution.

The hackers found that a tiny pulse in the reset signal of the main processor while the C function `memcmp`<sup>7</sup> executes resulted in it always returning a true Boolean value, regardless of the compared values. Unfortunately, developers often use the standard C function `memcmp` to compare two hash values to check whether their signatures match. To trick the bootloader, the

---

<sup>6</sup>Some fault injection attacks are more intrusive as they require the depackaging of the component. Depackaging is the operation that removes the silicon die package and sometimes removes some physical layers. This is the case with white light and laser attacks.

<sup>7</sup>`memcmp` is a standard function of the `libc` library that compares two blocks of consecutive bytes. If the blocks are the same, the returned value is true; else the returned value is false.

attackers have to generate a glitch in the reset signal at the proper time. Even if the signature of the forged software is to differ from the expected one, the comparison will indicate that it matches the expected result. Consequently, the bootloader may authorize the execution of the banned, flawed hypervisor. The bootloader believes that the presented hash does not belong to the blacklist and thus permits execution of the defective hypervisor. The difficult trick is to find the exact timing of when to apply the glitch.

The Xbox 360 comes in two versions: flat and slim. On the flat version of Xbox 360, the attackers discovered that pulling up a pin of the processor reduces its speed by about 120 times. Thanks to this trick, they profiled the bootloader to find the precise moment of the occurrence of the `memcmp` function that compares the hash of the executing hypervisor with the reference values. After many trials, they designed a successful hack.

1. Wait for the beginning of the execution of the `memcmp` function that compares the two hash values.
2. Slow down the processor when it starts the hash comparison by pulling up the corresponding pin.
3. Wait for a precise duration and then apply a 100-nS pulse to the reset pin of the processor. The slowing down of the processor increases the reliability of the hack as it requires less accurate timings.
4. Let the processor return to nominal speed by releasing the pin's voltage.

For the slim version of the Xbox 360, the hackers did not find a similar pin that would slow down the clock of the processor. Nevertheless, they found that a programmable phase-locked loop (PLL) chip drove the clock using an I2C bus. The I2C bus is a cheap, two-line, standardized, serial bus widely used in hardware design for many decades. The I2C bus is not protected and is well documented. The previous hack evolved into:

1. Wait for the beginning of the execution of the `memcmp` function that compares the two hash values.
2. Through the I2C bus, command the PLL to slow down the clock of the processor.
3. Measure exact duration and then apply a 20-nS pulse to the reset pin of the processor.
4. Through the I2C bus, command the PLL to return the processor to its nominal speed.

Fault injection attacks are not easily reproducible. They often require many successive failed attempts until one is successful. Nevertheless, these attacks are devastating. Moreover, they are difficult to prevent. In the case of the described hack, the attackers claimed that their average success rate was about 25 %. For this reason, successfully booting an unsigned code may require a few minutes of automatic trials before the Xbox is successfully fooled. Furthermore, fast and accurate hardware is required to generate pulses

as small as 20-nS. In other words, this attack is not easy and not practicable by newbies. Nevertheless, the outcome is that the attackers succeed in running an arbitrary piece of software on Xbox 360. Once the flawed hypervisor is running, it is possible to trick it to accept any arbitrary piece of code. This exploit is a physical attack as it requires access to the device and some physical equipment. At the time of writing this book, there was no known purely software-based attack.

*Lesson:* Attackers are resourceful. Even garage hackers may use sophisticated attacks. All sophisticated attacks do not require expensive, sophisticated tools. All attacks are not easy to reproduce repeatedly. Thus, some attacks may not have an enormous business impact and affect reputation more than finances.

Fault injections are extremely powerful attacks. They are often wrongly categorized as a type of side-channel attack. Side-channel attacks collect information leaking from side channels, and through careful analysis, they retrieve secret information. For instance, the side channel can be the duration needed to perform encryption, measurement of power consumption which gives an idea of how many transistors commute, or measurement of local electromagnetic radiations. Since the publication in 1996 of the famous, first timing attack by Paul Kocher [36], side-channel attacks became more sophisticated than that seminal one. They have extensively proven to be devastating and have forced manufacturers to design and implement dedicated countermeasures (Sect. 6.2.2).

The previous insert describes a hack dedicated to the Microsoft Xbox 360. Microsoft is not the only hacked game console manufacturer. At the time of this writing, every game console has been broken. The Nintendo Wii [37] and Nintendo DS have been broken for each new generation of product.

There was no protection in the initial versions of the Sony Play Station (PS). Their design allowed running any piece of software. Soon the hobbyist community became enthusiastic and extremely active. With its PS3, Sony decided to restrict the execution to approved pieces of software. The latest versions of Sony PS3 have also been successfully jailbroken. At the December 2010 Chaos Computer Club (CCC)<sup>8</sup> conference, George Hotz, by the nickname of GeoHot, disclosed the digital signature algorithm (DSA) private key used to sign the firmware of all PS3 devices [39]. Normally, it is impossible to guess a private key while knowing only its public key. The usual security assumption is that this private key never leaks out. Usually, private root keys are hosted in Hardware Secure Modules (HSMs) and stored in a safe. Strictly enforced, security policies govern access to the HSM. The absolute secrecy of the private key is the cornerstone assumption of most trust models.

---

<sup>8</sup>For over 25 years, the CCC has been the largest European hacker's group. The activities of the club extend from technical research and dissemination to political engagement [38]. Each year, its December conference gathers many of the most influential hackers worldwide. Many new exploits are disclosed during this event.

GeoHot could not get access to the HSM. Then, how did he find this private key? The mistake was a poor implementation of the signing software. In a proper implementation of the DSA signature, the signing process uses a different random value for each new signature. Unfortunately, Sony used the same fixed value within its DSA implementation. Every signature employed the same “random” value. This mistake, well known by the cryptographic community, allows guessing the private key [40] with a few valid signatures. Indeed, GeoHot explored this option and discovered that he could retrieve the signing key. Nevertheless, this was not the ultimate key as it protected “only” the firmware. In October 2012, hackers by the nickname of “The Three Musketeers” went a step further [41]. They guessed and released the private key used for the bootloader LV0. With this key, it was possible to sign and install any arbitrary software on the console. Unfortunately, the Musketeers did not disclose how they discovered this private key.

The game industry is under continuous attack from the hacking community. Sadly, the game industry made many mistakes with its security systems. Copy protection systems regularly applied to games anger consumers. Nevertheless, in Sect. 2.3.1, we will come back to the world of game consoles, but this time with a positive example.

Jailbreaking applies to devices other than game consoles. Indeed, the phone industry initially coined the word jailbreaking. Historically, jailbreaking a mobile phone has meant circumventing the system that enforces that the mobile phone uses only a given operator’s network and subscription. Many operators subsidize the cost of the mobile phone. Therefore, they want to guarantee their return on investment by restricting the use of the subsidized phone to its own carrier network (and associated carrier networks when roaming). Jailbreaking removes this restriction. At the end of July 2010, the US Copyright Office and the Librarian of Congress announced six new exemptions to the Digital Millennium Copyright Act (DMCA)<sup>9</sup> [43, 44]. Exemptions authorize circumventing protection measures as defined by the DMCA under very specific conditions. One of these exemptions states that jailbreaking mobile phones is legal if the purpose is to enable the use of the phone on other carrier networks.

With the advent of smartphones, and especially the Apple iPhone, jailbreaking took on a second meaning: allowing the execution of arbitrarily downloaded applications on the smartphone. This operation is sometimes called “rooting.” For instance, Apple iPhone, Apple iPod, and Apple iPad can exclusively execute applications downloaded from the Apple App Store. Apple applies extremely tight control on these applications. As usual when restricting usage, this was a challenge for hackers. An arms race started between Apple and the hacking community. Until iOS 9.0.2,<sup>10</sup> hackers offered a solution to run arbitrary applications on these

---

<sup>9</sup>Since October 28, 1998, the DMCA [42] defines the US copyright laws. Normally, under the DMCA, it is illegal to circumvent any security measure. Nevertheless, there are some exemptions to this rule. Since its inception, five such amendments were issued in 2000, 2003, 2006, 2010, and 2014, defining new exemptions to the DMCA rules.

<sup>10</sup>iOS 9.0.2 was the latest version of iOS at the time of editing this chapter.

devices. They used pieces of software such as `redsn0w`, `Absinthe`, `JailBreakMe` [45], and `Pangu` [46]. Rooting enabled using an alternate application store: `Cydia` [47].

Other manufacturers also attempt to control their execution environment. Nevertheless, the advent of Android has tended to reduce this practice. Having access to a vast library of applications is an excellent selling argument.<sup>11</sup> Unfortunately, rooting introduces other security issues, such as opening the door to malware.

### 1.2.3 *Flawed Designs*

In the case of side-channel attacks and Xbox attacks, the success came from a new attack that was unknown to the designers. Unfortunately, often the attack succeeds because the design has serious security flaws [49]. Sometimes, poor design choices of the security designer are the cause of such defects. The following is a grotesque example. Wi-Fi routers currently use a secure protocol, WPA2-PSK, to protect the wireless communications against eavesdropping. This protocol uses a secret symmetric key that should be unique for each router. Usually, the router manufacturer stores on the device a random key generated by the factory. This key (or the passphrase that will generate this key) is often printed together with the network name (SSID<sup>12</sup>) on a label affixed to the router. The key is a long set of hexadecimal digits that the user might have to enter when joining a network for the first time.<sup>13</sup> Best security practice assumes that this WPA2-PSK key is random so that an eavesdropper cannot guess the key needed to join a network stealthily. On some of its models, the manufacturer Belkin did not respect this rule [51]. Rather than being truly random, the key was derived from the wide area network (WAN) medium access control (MAC) address of the router. By construction, the WAN MAC address is public information. The router broadcasts it. Thus, this address is available to any attacker. Unfortunately, the algorithm that derived the key from the WAN MAC address was far too simplistic. Each digit of the key was the result of a static substitution of one digit of the WAN MAC address. Once the substitution table was guessed, it was easy to calculate this pseudorandom key. Furthermore, Belkin's passphrase used only eight hexadecimal digits, i.e., 32 bits of entropy.

---

<sup>11</sup>Nevertheless, there are also "rooting" exploits available for Android. One example is `Towelroot`, designed by GeoHot [48].

<sup>12</sup>The Secure Set Identifier (SSID) is the alphanumeric string that is part of the header of the packets over wireless local area networks.

<sup>13</sup>A new protocol called Wi-fi Protected Setup (WPS) allows users to bypass this clumsy, complex phase of dialing long passwords. Unfortunately, once more, some weak implementations of this protocol undermined the security of some devices [50].



Exploring by brute force<sup>14</sup> 32 bits of entropy is currently just a matter of a few hours. For a symmetric key to be safe, this key should be at least 90 bits long [52]. This too short length was another significant design error.

It is interesting to consider why a manufacturer uses a deterministic algorithm rather than a true random number generator for creating the secret key. The reason is the ability to restore default values in the case of a problem. If the manufacturer were to use a random number generator, then it would have no way to provide the passphrase to Alice if ever the sticker of her router was destroyed or erased. The only solution would be to store all the generated passphrases in a database securely. The manufacturer or the operator has to offer through its hotline a sophisticated retrieval service. First, the operator must assess whether that the calling customer owns the device. Once this ownership is verified, the operator can deliver the value of the forgotten passphrase. This manual operation is expensive. Using a pseudo-random number generator (PRNG) with a known seed simplifies the logistics and reduces the cost. Nevertheless, this money-driven design choice requires a strong PRNG, which was not the case for Belkin (Chap. 3).

In the previous example, the motivation driving the error was the manufacturer's convenience and cost constraints. One of the leading causes of this flaw comes from a bad balance between user-friendly, marketable features and security constraints. For instance, for increased user-friendliness, car manufacturers introduced a new breed of car keys: the Passive Keyless Entry and Start (PKES) system. These car keys use radio frequency identification (RFID) wireless communication with the car over a secure protocol. The idea is that every automobile and its car key are paired. The vehicle automatically detects the presence of the right paired car key and acts correspondingly. For instance, when the car key is in the range of two meters of the car, the car will authorize the driver to open the doors with the handle (without his having to use the physical key). If the driver is inside the car, it authorizes his starting of the engine. As the car key remains in the driver's pocket or bag, the interaction with the car's access control is invisible to the driver. Obviously, this feature is extremely user-friendly.

Unfortunately, in 2011, three researchers from ETH Zurich, Aurélien Francillon, Boris Danev, and Srdjan Capkun, demonstrated a simple attack: a classical relay attack [53]. In a relay attack, the messages are retransmitted to make the communicating entities believe that they are closer than in reality. Thus, the weakness was not in the protocol itself, but in the initial concept. In PKES, the car initiates the challenge. The researchers exploit this specificity. They place a first antenna near the car to capture the emitted message of the car (as the antenna of the car key would do). Then, these messages are relayed to a second antenna close to the car key (8–10 m). Thus, the second antenna mimics the car emitter's behavior. This message relaying is independent of any logical protocol. A simple cable or RF

---

<sup>14</sup>Brute force attacks explore systematically every possible value of the key until one succeeds. Thus, in the case of 32 bits, it means at maximum 4,294,967,296 trials. Unfortunately, with current computers, exploring a 32-bit space is extremely fast. A brute force attack is the simplest attack. The defense is to increase the length of the key.

transmission links the two antennas. In other words, the system transmits messages between the car and the physical key a longer distance than intended. Consequently, if the attacker knows the location of the car, and if the attacker comes reasonably close to the car's owner, she may steal the signal of the car key. Near this targeted car, her accomplice can unlock the car and steal it. The researchers demonstrated this attack on ten car models from eight manufacturers. A few years later, high-tech thieves used this attack in the street [54].

The design is flawed because it assumes that the presence of the signal means that the emitting device is nearby. The researchers suggested one first simple countermeasure: Deactivate the car key with a physical switch. Unfortunately, this countermeasure has two issues. First, it does not take into account human factors. Without doubt, some people will forget to deactivate the key when leaving their car. Other users may not remember that they had disabled the car key and thus will struggle when trying to open the car, expecting it to automatically open despite their having deactivated the feature previously. The second issue is that adding a switch button would annul the perceived benefit of this system: being buttonless. This is most probably the heart of the problem. Unlocking the car is done without conscious action by the owner. Is it wise from a security point of view? The security designer should never neglect human factors (Chap. 7).

The researchers proposed a second countermeasure more complicated than the previous one. This solution requires accurately measuring the trip time, i.e., the duration needed for a message to reach the target point. The relay attack increases the trip time. Therefore, the car could detect its interception of the communication channel. Unfortunately, the trip time of wireless communication is not stable and precise. The dispersion of values is high. A class of secure protocols called distance bounding protocols attempts to prevent relay attacks [55, 56].

Sometimes, a feature of a system is an intrinsic vulnerability by design, as users may employ it in a way that its designers did not foresee. For example, Microsoft designed a proprietary file format, the Advanced Systems Format (ASF), for storing and playing digital media content. It is a container format used for Windows Media Audio (.wma) and Windows Media Video (.wmv). In addition to the audio and video streams, an ASF file may also contain text streams, Web pages, and script commands. One script command is extremely attractive to attackers: URLANDEXIT (code value 81). This command requires the player to open the Web page whose uniform resource locator (URL) is following the command code. The Web page is accessed automatically without any user interaction. The initial purpose was to be able to download transparently a new breed of codec that was not yet supported by the host. Unfortunately, in 2008, attackers used this documented feature to access sites that would distribute malwares [57]. For instance, the Trojan Win32.ASF-Hijacker.A searched the hard drive of the infected computer for ASF files [58]. Once the malware found them, it injected into these files the URLANDEXIT command pointing to a remote Web site controlled by its designers. The infected media file was ready to spread the infection. Many variants and malwares used this URLANDEXIT command.

Unfortunately, this dangerous feature is enabled by default. Microsoft released an update patch that allowed disabling this feature by manually updating some registry keys. Such operation requires some skill and is not user-friendly. Interestingly, the Trojan Win32.ASF-Hijacker.A disabled this feature for its infection to remain unnoticed by the computer that created the forged content. Not all players are vulnerable. For instance, the widely deployed, open source multimedia player VLC does not support the script commands of ASF.

As specified by Microsoft, this issue was not a security vulnerability but rather was a feature by design [59].

When a content owner creates an audio or a video stream, that content owner can add script commands (such as URL script commands and custom script commands) that are embedded in the stream. When the stream is played back, the script commands can trigger events in an embedded player program, or they can start your Web browser and then connect to a particular Web page. This behavior is by design.

Sometimes, an excellent feature may turn into an exploitable hole. The Thunderbolt port is an Apple proprietary port that combines both the DisplayPort for displays and PCIe for connecting devices using one unique physical connector. At the December CCC 2014 conference, Trammel Hudson disclosed the first known proof of concept of a bootkit for Mac OS X [60] using the Thunderbolt port. Bootkits are a particular category of rootkits that stealthily infect the master boot record or the volume boot record. The master boot record is the piece of software that the Basic Input/Output System (e 1st occ. expansion given) loads to initiate the boot. The volume boot record holds in the first partition the piece of software that loads the OS. In other words, a bootkit is a rootkit that installs itself in the boot system of the machine before the installation of the OS. Thus, the loaded OS is unaware of the presence of the bootkit.

Hudson's exploit uses several weaknesses in the boot system of Mac OS X.

- A CRC32, rather than a cryptographic signature, protects the integrity of the boot Read-Only Memory (ROM).<sup>15</sup> Unfortunately, the purpose of a cyclic redundancy code (CRC) is to check that the data is not corrupted (i.e., there is no mistake due to transmission). The goal of CRC is not to verify whether a principal modified a piece of data. Forging a CRC that matches an altered piece of software is extremely straightforward. The attacker can modify the boot process software and bypass the control of integrity by just calculating the new CRC and replacing the previous value with the calculated one.
- The firmware that is to be upgraded with the Extensible Firmware Interface (EFI) is signed with RSA 2048. However, the verification of the signature is done by the boot software. The previous vulnerability may alter this bootloader. The attacker may load her forged firmware at boot time using EFI if she can

---

<sup>15</sup>In the case of Mac OS, the ROM is indeed an electrically erasable programmable Read-Only Memory (EEPROM). This allows a potential upgrade of after its deployment.

deliver it to the machine to infect. Nevertheless, this attack requires physical access to the machine.

- Hudson used an attack that was disclosed in 2012 [61]. At boot time, EFI queries external devices that are connected through PCIe about whether they have any Option ROMs to execute. An Option ROM or expansion ROM is a piece of firmware that the BIOS launches at boot time. The Apple Thunderbolt port also supports the PCIe interface. The Thunderbolt port allows this function to load an arbitrary firmware from a connected device that would announce it has an Option ROM.
- Hudson fooled the boot firmware by replacing Apple's public key with his public key. Thus, the Apple software checks the signature of his malware and decides that it is legitimate, as the replaced key pair, used to verify the signature, and signed the malware. The legitimate Apple software then executes the malicious Option ROM. Later, the attacker's public key is written down in the ROM, preventing any Apple-authorized firmware upgrade from occurring. The compromised device will only accept firmware updates signed with the attacker's private key.

The potential attack is to design a forged Thunderbolt device with the expected malware as an Option ROM. The attacker needs physical access to the target computer. Then, she boots the device with the connected forged Thunderbolt device. In a few minutes, the attacker owns the machine. The entire attack is fast. Apple prepared fixes that prevent Option ROM execution during a firmware upgrade.

In 2014, Luyi Xing and his colleagues disclosed a new class of attack, which they coined *pileup* [62]. The attack uses an update of the OS to create new vulnerabilities. The researchers used Android to demonstrate the concept. Nevertheless, the attack may apply to other OSs. Google issues a new version of Android about every year. Each update usually features new system applications, new permissions, and new attributes. An increment of the application programming interface (API) level index signals this evolution. An attacker knows the history of this development. She designs a malicious application that uses selected permissions and attributes that are only available in the version of Android with the highest API level. The versions of the OS with a lower API level do not know these permissions. Thus, at installation time, they do not ask the user to grant this unknown permission. The forged application is installed on a device running a version of Android with a lower API level. The malicious application cannot use these newer unsupported features. Later, the device may be updated to the latest version of Android. When upgrading, Android automatically grants the previously claimed permissions, privileges, or attributes without asking for confirmation from the user again. It would not be user-friendly for the OS to request permission from the end user for each application he previously installed. This systematic request for each application would make the upgrade process tedious. The researchers gave some demonstrations of *pileup* attacks. For instance, Android 2.3 (so-called Gingerbread with API level 9) does not have permission to receive Google

Voice SMS.<sup>16</sup> This permission has been only present since Android 4.0 (so-called Ice Cream Sandwich with API level 14). Using such a pileup attack, the malicious application could read SMS messages of Google Voice without the user being aware once he upgraded to a version of Android greater than 4.0.

Another method is to claim some package name that the system will use in a future upgraded version. Once the OS is upgraded, the previous data will be kept. If malicious, this data may contaminate the system application. Pileup attacks may be dangerous in an extremely fragmented environment such as Android's.

Modern designs are extremely complex. Exploring all the use cases is difficult, if not impossible. It is even harder to foresee scenarios that will benefit from the evolution of the product. Flaws are ineluctable in elaborate designs. The attackers will use such flawed designs to create exploits.

### ***1.2.4 Advanced Persistent Threats***

There are two categories of attacks: opportunistic attacks and targeted attacks. In the case of opportunistic attacks, the attacker randomly selects her targets. For each random target, she checks whether she can successfully apply her exploit. If it is the case, then she perpetrates the attack, intrudes on the system, and attempts to dive deeper into it. If she fails, she looks for another victim. Usually, this type of attacker does not invest much effort into breaking into targets. If a target does not easily surrender, then the attacker moves to another random target. For example, this routinely occurs on Internet routers. Attackers scan IP addresses to find open ports, usually using automated tools, such as `nmap` or the Shodan search engine. Then, the reported open ports are analyzed to check whether their corresponding protocols are vulnerable to widely known exploits. Perimetric defense tools prevent attackers from breaching the perimeter defined by a company. They include firewalls, intrusion detection systems, and antiviruses. As long as they are up to date, typical perimetric defense tools are usually sufficient to deter opportunistic attacks. Opportunistic attackers always look for the weakest preys.

In the case of targeted attacks, the attacker focuses on a precise target or at least on a category of targets. The attacker's goal is to fulfill a particular task. For that purpose, the attacker uses a broad range of techniques. The attack may be sophisticated and encompasses many steps. The first step collects maximum amount of information about the target. This step uses all available resources, such as published information, stolen or leaked proprietary information, and information gathered through social engineering [63]. Business intelligence tools such as Maltego may help the attacker to cross-check the collected information. During the second step, the attacker attempts to breach the target using the collected information. Depending on the actual security of the target, this second step can be

---

<sup>16</sup>The permission is `com.google.googlevoice.RECEIVE_SMS`.



**Fig. 1.3** RSA SecurID

extremely complex and may involve multiple attacks spread over different systems. The new term advanced persistent threats (APTs) describes the most sophisticated attacks of this category. The most interesting recent ATP was the attack against RSA Ltd.

SecurID is a widely used two-factor authentication token. The system is usually used to authenticate a user, either to grant him access to a remote service or system or to establish a virtual private network (VPN) with his company's network. The token displays a six-digit number that changes every few seconds following a secret proprietary algorithm (Fig. 1.3). Each token is associated with one person. When receiving her token from her IT department, Alice is also attributed a four-digit personal identification number (PIN). To authenticate with SecurID, Alice provides her login identity (for instance, her email address), her four-digit PIN, and the six digits currently displayed by the SecurID token. The authenticating server checks whether these three parameters fit together. If they do, then the server authenticates Alice. The SecurID system is a two-factor authentication mechanism because it checks:

1. Something Alice knows, e.g., her PIN; this renders the theft of the SecurID token useless. The thief would not know her four-digit PIN.
2. Something that Alice owns, e.g., the token itself; the continuously changing value of the six digits proves that she currently holds it. This solution renders shoulder surfing useless because the validity of the code has an extremely short life (a few seconds).

On March 18, 2011, Art Coviello, the executive chair of RSA Ltd., announced to his customers that his company had been attacked, and some information leaked out [64]. The stolen information may have reduced the security of the SecurID tokens. RSA Ltd. did not disclose what information the intruders took. Nevertheless [65],

we may infer that the intruders have now a method to impersonate Alice's SecurID token or a way to fool the authentication server into believing that the login user is Alice.

RSA Ltd. properly managed this crisis. The company had a positive, responsible reaction regarding the hack. It provided some details and timely visibility on the issue. Therefore, this APT teaches us many interesting lessons.

**The Devil Is in the Detail** The attackers penetrated the company RSA Ltd. through a targeted phishing email using a malicious Excel file [66]. Over 2 days, two small groups of EMC<sup>17</sup> employees received two spear phishing emails [67]. Phishing emails are emails that attempt to impersonate a known user or entity. Attackers specifically craft spear phishing emails to target an organization or specific individuals within the targeted organization. On March 19, 2011, one employee of EMC opened one of these emails, which were copied to three other employees. This email was sent supposedly by a recruiting Web site: beyond.com. This site is a certified partner of RSA Ltd., making the phishing email less suspicious and even plausible. The subject of the email, the 2011 recruiting plan, had an attached file, `2011 recruiting plan.xls`. The attack used a typical social engineering ploy: exploiting human curiosity and greed. It always works. The inquisitive employee opened the attached object.

The malicious, poisonous Excel file embeds an Adobe Flash object that Excel executes automatically when the file is opened. Using the vulnerability CVE-2011-0609 [68], the Flash object installs a malware: PoisonIvy Backdoor. This vulnerability allows remote attackers to execute arbitrary code using a crafted Flash object. The vulnerability was first disclosed on March 15, 2011, i.e., 4 days before the attack. The malevolent Flash object installed the backdoor. PoisonIvy practically gives the attacker full control of the infected computer [69]. For example, PoisonIvy can rename, delete, execute, upload, or download files, read and edit the Windows registry, manage services, and access remote drives. In the RSA attack, the instance of PoisonIvy connected back to the domain `good.mincetur.com`. For many years, the domain `mincesur.com` was known to be associated with illegal activities [70]. Ideally, a firewall or proxy server should have blocked this domain. The compromised account was not one of a critical IT staff member. Nevertheless, the attacker succeeded in escalating her rights by looking for other more privileged accounts. Then, the attacker was able to gain access to critical information concerning SecurID tokens. She compressed the collected data into several password-protected rar files. Using encrypted files may allow passing under the radar of eventual data loss prevention (DLP) tools that analyze the semantics and structure of exported files. Finally, the attacker

---

<sup>17</sup>EMC is the company that owns RSA Ltd.

exfiltrated the stolen data using one File Transfer Protocol (FTP) server.

Was RSA Ltd. guilty of not being up to date? When the attack occurred, RSA had no means to patch this vulnerability. Adobe announced that the corresponding patch would be available only on March 21, 2011, i.e., 2 days after the infection [71]. The problem was that the targeted employee opened the infected file. We would expect that employees of a security-dedicated company would be extremely careful about spam and phishing emails.

*Lesson:* Increase the security awareness of all employees. As there will always be 0-day vulnerabilities in the field, the defense of antiviruses is not sufficient. Perimetric defenses are necessary but not anymore sufficient. Only vigilance may thwart 0-day email attacks.

This is not the end of the story. At the end of May 2011, Lockheed Martin, one of the largest US defense contractors, was under a “significant and tenacious” online attack. Some attackers attempted to penetrate Lockheed Martin’s network [72]. It seems they tried to enter through Lockheed Martin’s VPN which was protected by RSA SecurID. Apparently, the attackers possessed the seeds, serial numbers of valid tokens, and the underlying algorithm used to secure SecurID. Lockheed Martin immediately detected the attack and took drastic, preventive measures. The company disconnected all remote accesses to its corporate network. All employees were requested to work only from the office for 1 week. Telecommuting was banned. The VPN was disabled. All SecurID tokens were replaced. Every Lockheed Martin employee had to change his network password. The proximity in time to the RSA Ltd. hack, the intrusion vector (RSA SecurID), and the target itself (a defense contractor) tend to indicate a high correlation between the two events. The first stage of the attack would have been the theft of the secret information of the SecurID. The second stage used this information to try to penetrate Lockheed Martin’s systems stealthily. This attack is perhaps the first publicly widely known instance of an APT.

The term APT comes from the military defense domain. Initially, APT was defined as a cyberattack launched by nation-state-funded organizations to steal, over a prolonged period, critical nation information using sophisticated, stealthy techniques. Currently, APT transcends the specialized realm of warfare to extend into business intelligence [73]. APT is an advanced attack because it employs an extensive set of tools and techniques to perform attacks. The attackers are well informed, well equipped, highly skilled, and well funded. They may design custom tools to succeed in their attacks. APT is persistent because the threat is not opportunistic, but it is intended to fulfill a specifically defined objective. The attackers will use all the time necessary to succeed. The cost of the operation is not an issue. APT is a dangerous threat because human beings fully manage the attack rather than robots or botnets. Human attackers are more opportunistic and reactive than robots. Usually, the life cycle of an APT has eight steps.



1. Collecting the target company: Information concerns the employees as well as the IT infrastructure. The objective of this step is twofold.
  - A better understanding of the network topology, the materials used, and the different pieces of software employed by the targeted company. This cartography can be done using exploration tools such as SinFP [74] and Metasploit [75] and by gathering public information available on the Internet. Data stolen via social engineering may be another worthwhile source of information.
  - A better understanding of the organization of the company; it is valuable to identify people who may have the “nearest” possible access to the targeted data. Obviously, IT administrators are ideal targets. The social engineer uses all available sources. There are many potential sources of names. Social networks are excellent hunting places. Furthermore, professional social networks such as LinkedIn are even better locations to identify and collect information about potential, initial entry points [76]. It is easy to find the name of the chief information officer of an organization and the key members of the IT staff of a company via professional social networks.
2. The analysis of data collected during the previous step provides a list of potentially interesting targets. Through social engineering, the attacker targets the identified users. This step uses the full panoply of methods, such as spamming, spear phishing, and impersonating calls. See Chap. 7.
3. The attacker attempts to intrude the target. Once a gullible user found, the attacker uses vulnerabilities (often 0-day vulnerabilities) to install a Trojan horse on the victim’s computer. This backdoor will be the entry point of the hack. Of course, the installation and the operation of the infecting malware have to be stealthy. This is true for modern Trojans.

Spear phishing remains the favorite method for initial infection. New file-sharing sites, such as DropBox, have become fashionable in the hacking world [77]. The phishing email points to a document shared through such a service. Such cloud-based services can offer efficient anonymity to the attacker. Tracing back the owner of the account is hard for forensic investigators.

A waterhole attack is another example of a method employed to inject a backdoor [78]. Once the attacker has profiled the victim in step 2, she can identify a list of Web sites the victim will most likely visit. Then, the attacker analyzes these Web sites to find vulnerable ones. In the vulnerable ones, she attempts to implement some malware that will try to infect the victim’s computer. The attacker’s malware waits until the victim visits the Web site, like a lion waiting for its prey to come to the waterhole. During the victim’s visit to a compromised site, the trapped malware attempts to infect the visiting computer. Infected files are often powerful vectors of infection. Some sophisticated malwares identify the victim and affect only the targeted victim and not the visitors that the APT did not target. This trick reduces the risk of being detected as only

a limited number of computers will be infected. Thus, they reduce their own attack surface as malware detection tools do not see them.

4. Once inside the system, if the victim has no access to the servers that the attacker was targeting, she expands laterally to roam around the network. In other words, the attacker looks for other vulnerable computers on the internal network that would allow her to come nearer to her final target.
5. Escalate privileges (social engineering, access to administrators' encrypted passwords, etc.): As the targeted data is sensitive, we may expect them to be protected strongly. Only higher privileged principals should be granted access to them.
6. Compromise the system holding the targeted data.
7. Exfiltrate the data stealthily: Usually, the attacker encrypts the exfiltrated data to avoid detection by DLP tools, deep packet inspection, or any filtering proxies. Of course, the data transmission uses covert channels, i.e., hiding among legitimate communications. Sometimes, these covert channels are throttled down to avoid triggering detection of anomalies in the communication. The overall transfer takes longer, but it is less visible to monitoring than the open channel. APTs are persistent, and time is usually not an issue. The exfiltration occurs either by using the available communication channels or via the backdoor installed during step 3. Often, communication and control (C&C) servers are used in a similar way for botnet management.
8. Clean up to avoid being traced back: An APT should not be detectable. Stolen data has even more value when the victim is not aware of the leak. Therefore, the attacker clears the logfiles but also may remove the backdoors that she installed. Sometimes, she may keep under control the compromised computer that she used to infiltrate the system. This control allows her getting back to collect information later. For instance, the alleged Chinese APT group called APT1 or comment crew keeps control of the victims' system for an average of 200 days [79].

In summary, the attacker studies the system in steps 1 and 2. She gains access to it in step 3. She enhances this access in steps 4 and 5. She exploits this access in steps 6 and 7. Eventually, she closes this access in step 8. In the literature, these eight steps are sometimes condensed into five steps: scouting (steps 1–2), intrusion (step 3), discovery (steps 4–5), capture (step 6), and exfiltration (steps 7–8).

An APT may be more complex than described in the previous section. For instance, an APT may first need an APT against another target to acquire some critical information or find external access to the final target. This was probably the case with the Lockheed Martin attack, which required first stealing information from RSA Ltd. to penetrate Lockheed Martin's system. In 2012, Adobe announced that it discovered two pieces of malware that presented a valid Adobe signature [80]. Attackers compromised a build server of Adobe and through it requested Adobe to sign the pieces of malware. As one of the pieces of malware was not "publicly" available, the likelihood that this was part of a larger APT is extremely high.

Defending against APT is a complex and costly endeavor. The traditional perimeteric defense tools are insufficient. More sophisticated tools are mandatory, such as [81]:

- Automated sandboxing that analyzes every received file in a safe, isolated environment,
- Whitelisting applications that exclusively authorize execution of known trusted applications on a given device,
- Deep network traffic inspection that checks whether the communication is compliant with established protocols and acceptable patterns.

The ultimate defense is the human analysis by experts who carefully monitor activities and logfiles looking for anomalies and swiftly react. See Chap. 9.

APT should not be a concern for consumers. APT usually affects companies that handle strategic or critical infrastructures or organizations that have high-value intellectual properties. Third parties that work for such targets may become ancillary victims.

## 1.3 Takeaway

The previous sections highlight that attackers will always find a way to break the system. Once this fact is accepted, the security designer has to take this reality into account in all his designs. In a white paper, Kaspersky Lab identify that “assuming everything is OK” is one of the ten enablers of the IT department facilitating cybercrime [82]. Overconfidence is the worst weakness of the security designer. The security designer must never forget that the attackers will defeat him. The next sections present how the design must reflect this persistent threat.

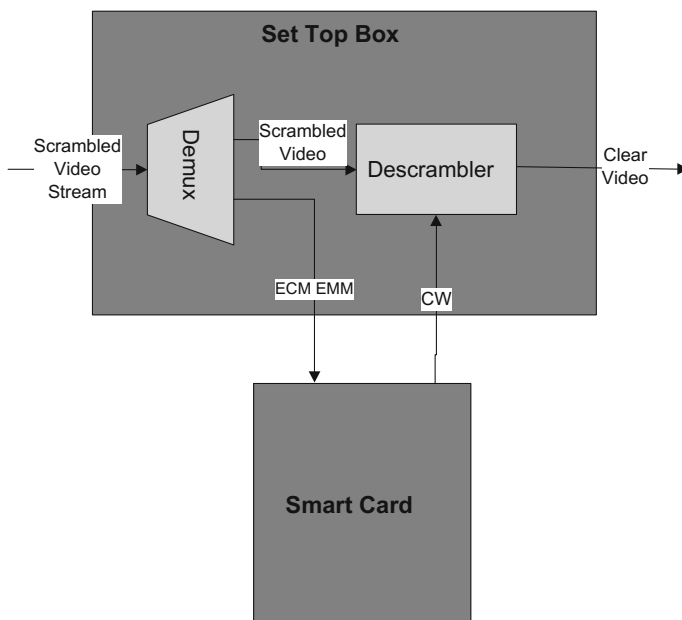
### 1.3.1 *Design Your System for Renewability*

*“Even if the Allies lost this battle, we should not have lost the war.”* The British Minister of Information, Duff Cooper, noted this sentence on May 28, 1940, after the defeat of the Battle of France. Every security designer should endorse this sentence. The system must be able to recover after any successful attack. In other words, the system must be able to renew itself. This section starts by examining a class of systems that were designed with renewability as a core feature.

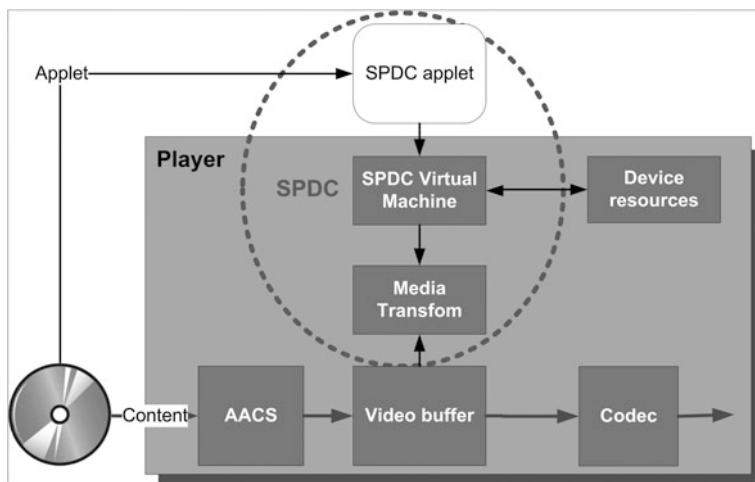
In 1984, the French broadcaster Canal+ launched its first subscription-based Pay TV channel. A few months later, the first schematics of pirate decoders were published. In 1985, the US operator HBO used Videocipher II to protect its satellite programs. Once more, pirate decoders were available soon on the market. The piracy market was organized as quasi-industrial organizations [83]. These first

commercial deployments of Pay TV highlighted the need for longer keys and renewability. Two European Conditional Access Systems (CASs), Eurocrypt and Videocrypt, built the foundations of modern content protection. The Union Européenne de Radiodiffusion (UER) established Eurocrypt as an international standard. The Eurocrypt specifications created the vocabulary that is still in use today in Pay TV systems [84]. News DataCom and Thomson codesigned Videocrypt [85]. These systems created a new generic scheme for content protection [86] that is still in use today.

The main significant improvement over previous Pay TV systems came from the use of smart cards. Smart cards are removable secure processors. Figure 1.4 provides an overview of the utilization of the smart card within a Pay TV system. The set-top box receives a scrambled video stream multiplexed with encrypted information: entitlement control messages (ECMs) and entitlement management messages (EMMs). An ECM contains the decryption key, the so-called control word (CW), used to scramble the video, whereas an EMM carries the credentials that manage user rights. The STB extracts the ECM and EMM from the stream and transmits them to the attached smart card. The smart card holds in its secure memory both the secret keys and the algorithms used by the Pay TV system. It decrypts the ECM and checks whether the holder of the smart card is entitled to watch the current program. If this is the case, then the smart card returns the control word to the STB. The STB descrambles the scrambled video using the returned CW.



**Fig. 1.4** Simplified view of a card-based Pay TV system



**Fig. 1.5** Basic synopsis of SPDC

The initial assumption behind the trust model of modern Pay TV was that crackers would ultimately break the system and, in particular, the key management component. The assumption was that nobody would crack the scrambling method. In other words, the design assumption was that it was impossible to build a pirate decoder without first breaking the key management. Thus, the smart card implemented the entire key management and the management of entitlement rights. Changing the smart cards was a powerful way to address potential future hacks of the key management. This approach has proven to be successful.

Eurocrypt and Videocrypt survived many years. They were phased out by obsolescence of the video technology rather than by crackers. In 1995, the Digital Video Broadcasting (DVB) group standardized a way to protect MPEG2 transport streams. It employed the same smart card-based architecture. For more than 30 years, despite several successful hacks, most smart card-based CASs recovered.

The Digital Video Disc (DVD) was hacked in less than 2 years [87] after its launch. Unfortunately, its protection mechanism was not renewable. The hole could never be closed. Therefore, when the entertainment industry decided to design the Blu-ray Disc, renewability of the security was one of the main requirements of the future systems. In addition to the encryption mechanism [called Advanced Access Control System (AACS)], the Blu-ray Disc Association specified an additional layer of security, called BD+. The purpose of this layer is to enable recovery from potential security breaches occurring in the AACS. BD+ disks contain a title-specific piece of software, called the applet, which executes on a BD+ player before and during the movie playback. The program is written in self-protecting digital content (SPDC) language, a dedicated interpreted language designed by

Cryptography Research Incorporated<sup>18</sup> [88]. Figure 1.5 describes its main elements. SPDC is mainly a virtual machine (VM) and a media transform block. The VM is far simpler than Java's, another well-known VM. The media transform block can apply simple modifications to the compressed video. When reading a BD+ disk, the player loads the applet from the disk. Then, the player executes the applet. Usually, this applet first checks whether the host player is compliant and not revoked. Then, the applet provides configuration parameters to the media transform block. This block deterministically modifies, using the supplied parameters, the compressed content, for instance, by swapping some bytes in the stream.

In the BD+ environment, the applet is stored on the disk. This applet modifies the AACs-decrypted content before video decoding. This downloadable applet ensures the renewability of the security of BD+; if ever AACs or one of its implementations were to be hacked, the use of an applet would allow the broken AACs to heal. At the authoring stage, the essence is modified with a function  $f$  before being AACs-encrypted. Thus, on the disk, there is  $AACs(f(essence))$ . The applet that defines the reverse transformation, i.e.,  $f^{-1}$ , is delivered within the Blu-ray Disc. When the disk is played back, the applet will apply  $f^{-1}$  to the AACs-decrypted essence, i.e.,  $f(essence)$ . It is assumed that non-compliant players will not properly execute this applet or that the applet will detect non-compliant players. Thus, if the applet does not execute, the non-compliant player will try to render  $f(essence)$  rather than the essence itself; hence, the rendering will fail. The applet may also enforce a given manufacturer's model to perform some operations that will patch a security hole.

### Rule 1.3: Design Renewable Systems

Renewing security is possible in three ways.

1. Renewing the piece of software that handles the security: This is useful if the algorithm or its implementation has been broken. This either replaces the compromised algorithm by a secure one or replaces the weak implementation of the algorithm by a more secure one.
2. Renewing the cryptographic material used by the system: This is mandatory when a secret key leaked out. Of course, the transfer to the host, as well as the installation of the new cryptographic material, has to be secure. The secret data may be changed for the entire installed base in the case of a class attack or only for a set of principals if the attack is more localized.
3. Renewing the complete system, i.e., both the algorithms and the secrets: This is in the case of extremely severe attack.

In the example of Pay TV, as a smart card encompasses the execution environment, the software, and the secrets, all three kinds of renewability are extremely

---

<sup>18</sup>Cryptography Research Incorporated (CRI) is the company founded by Paul Kocher who designed the first side-channel attacks: timing analysis and power analysis. In 2013, Rambus acquired CRI.

simple to activate. In the extreme case, renewability just requires the replacement of the physical element (although this replacement may imply serious challenges of logistics). Of course, the secure download of software and data may also be used for partially renewing security in smart cards. In the case of embedded devices, such as consumer products, methods 1 and 2 are possible but require a root of trust. Usually, the secure loader builds this Root of Trust (Sect. 1.2.2). If the secure loader is compromised, then it is not possible anymore to renew the system. In the case of computers, often the third method is preferred. Replacing an entire system is rather easy in this configuration. Only the new version of the piece of software needs to be downloaded. This ease of replacement has, of course, a price: the lack of hardware Root of Trust (Sect. 4.2.5). There is no way to guarantee execution in a trusted environment. Malware may compromise the newly downloaded software. In current general computers, no trusted software or trusted hardware checks the signature of a piece of software.<sup>19</sup> The verification of the signature may not operate in a trusted environment and can itself be the victim of an attack.

Some people wrongly associate revocation to a mechanism of renewability. The aim of revocation is to disable a principal whose security has been compromised or who is not anymore trusted. No compliant system should interact with a revoked principal. Revocation is a major weapon in the battle against attacks, but it does not replace renewability. Indeed, once a secret revoked, there is a need to replace the revoked secret by a new valid one. This is the role of renewability. Existing standards handle revocation, such as X509 [89]. Unfortunately, there is no standard for renewability mechanisms.

### 1.3.2 *Design for Secure Failure*

The previous framework for in-depth defense recommends addressing any consequences of an attack. A smart way to cure is to prevent the failure from impairing the system. In other words, the system should fail securely in the case of a successful hack. When the system is hacked, the attacker should not gain undue advantages. Her benefits while gaining access should be annulled or at least minimized. Safe failure is a concept widely used in the safety of systems. Critical systems are built so that if a failure occurs, the system may still operate correctly, or, at least, its critical, vital functions should be maintained. Nuclear power units, aircraft, and automotive systems are all designed with a strong requirement in mind to fail safely, i.e., avoiding putting users in danger. There is a clear distinction between failing safely and failing securely. For instance, a fail-safe lock will be unlocked in the case of power shutdown for people to leave the room safely. Under

---

<sup>19</sup>To be precise, many modern computers have the possibility to implement such a Root of Trust because they are equipped with Trusted Platform Modules (TPMs). Unfortunately, the major operating systems do not take advantage of this feature.

the same conditions, a fail-secure lock will remain closed to prevent attackers from entering the room. Safety and security are sometimes opposites.

HSMs present such secure failing defense. HSMs are highly secure devices dedicated to cryptographic operations. They have two key features.

- Dedicated, embedded cryptographic coprocessors, which perform exponentiation, drastically accelerate public key cryptography.
- Highly secure components that, like smart cards, they include many tamper-resistant features. These devices prevent the disclosure of stored secrets.

Particular care is given to protecting the most valuable assets of the HSM, i.e., the private or secret key(s) stored in its secure memory. A typical security measure is the safe erasure of these keys whenever the HSM suspects that an attack is occurring. The range of monitored attacks includes power attacks, as when glitching the power supply to attempt fault injection; timing attacks, as when slowing down the CPU clock to better analyze the datagrams; chemical, as when depassivating the chip, for instance, with nitrofluoric acid; and physical attacks, through microprobing. Thus, even if the attacker succeeds in her attack, she will only access random data rather than the actual keys.

Securely failing is important also when designing secure protocols. One of the difficult challenges relies on the so-called atomicity of the protocol. If ever a protocol is interrupted before its normal completion, it should fail safely. An attacker should not benefit from such an interruption. In 2015, a black-box device was available to brute-force the PIN of iPhones without triggering the restriction of ten failed attempts [90]. The hack used an optical sensor that checked the screen, a relay to command the power supply, and a USB cable. The box tries a PIN number and sends it to the iPhone using the USB port. If the dialed PIN is wrong, the optical sensor detects the error message, and the box turns off immediately the power supply of the iPhone and reboots it. The box tries another potential PIN. As the phone has to reboot after each trial, each failed attempt took about 40 s. Brute forcing a four-digit PIN took about 4 days.

Figure 1.6a represents the mistake. The iPhone likely checks first the validity of the PIN. If the dialed PIN is wrong, it displays the result. Then, it updates the failed attempt counter and checks whether the counter reached the maximum number of authorized failed attempts. The black box switches off the power supply before the iPhone can update the failed attempt counter in its nonvolatile memory. When rebooting, the failed attempt has not been registered in the nonvolatile memory. Fault injections often exploit this classical known mistake in the design. Figure 1.6b represents a potential countermeasure. The initial value of the failed attempt counter is set to the maximum number of trials. The failed attempt counter should be decremented before the verification of the PIN occurs. If the PIN is correct, the failed attempt counter is set back to the top limit. In this order of operations, the protocol fails safely. If the hacker switches off the power supply after the verification of the PIN, then the counter is already decremented. She gains nothing.





**Fig. 1.6** Implementing a PIN verification

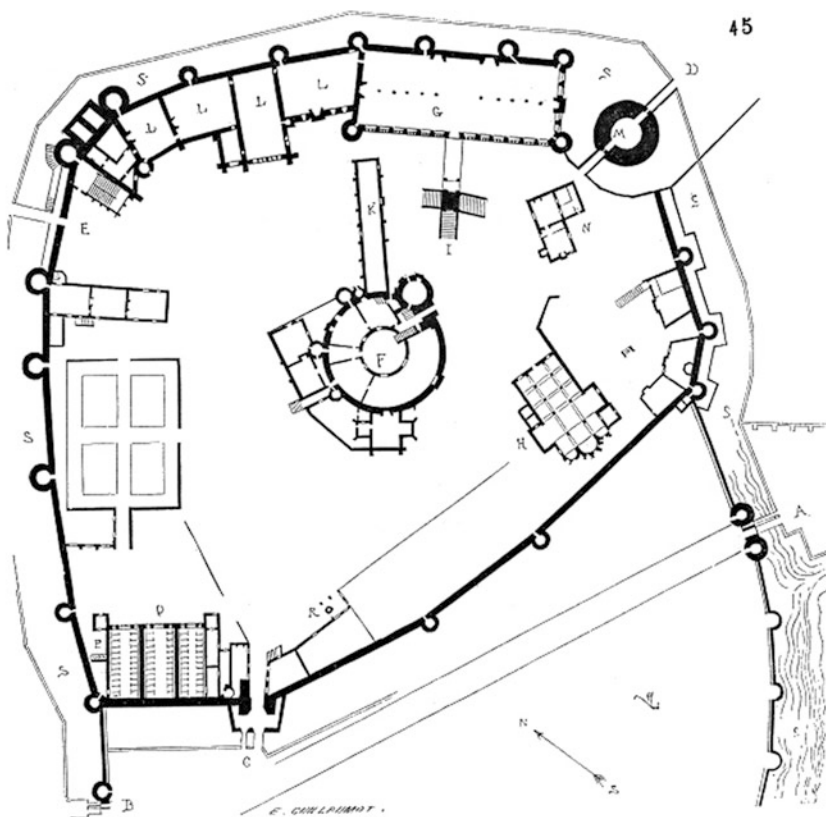
The designer should think how the attacker might derail his implementation for her advantage by injecting fault. Several techniques are possible for making the execution flow more robust against fault injection. Usually, the objective of a fault injection is to counter a Boolean test. One such technique is to split one Boolean test into a set of successive Boolean tests. For instance, rather than performing a comparison on the full data, the designer may use successive comparisons on subparts of the complete data. The attacker has to identify the occurrence of each comparison rather than one unique occurrence. The needed effort increases proportionally, and the likelihood decreases proportionally.

Securely failing is also important when writing security policies. There is no possible sound security in an organization if there are no well-documented security policies. The policy defines, at a high level, what is authorized and what is forbidden. It is essential that there be as little ambiguity as possible. Gray areas are always potential weak points in a security policy. The likelihood is extremely small that the editors have explored thoroughly and defined all the possible situations exhaustively. This likelihood may even be zero. Therefore, it is necessary to state that if a situation is not expressly permitted, it is prohibited by default [91]. This is a secure-fail condition. Good security policies ban an unknown legitimate situation

rather than authorize an unknown dangerous situation [92]. Afterward, it is always possible to update the policy to cope with unforeseen legitimate situations without having to heal after the wreckage of an attack.

### 1.3.3 Defense in Depth

Medieval castle builders had substantive notions of security. Figure 1.7 illustrates a typical castle from the Middle Ages. Most of the fortified castles followed the same architectural core principles. A first, outer rampart protected the castle. Often a deep ditch surrounded this wall, or, even better, the castle was on top of a cliff. A portcullis protected the access to the inside of the castle. Defenders lifted the portcullis in the case of an attack. Usually, the houses and common infrastructures,



<http://commons.wikimedia.org/wiki/File:Plan.chateau.Montargis.png#mediaviewer/Fichier:Plan.chateau.Montargis.png>

**Fig. 1.7** Medieval castle

such as the blacksmith's workshop or shops, were located behind this first outer wall. A second concentric wall, which was higher than the previous rampart, isolated from the common infrastructures the most critical infrastructures, such as the armory, the warehouses, and soldiers' barracks. The ultimate protection was the keep (location F in Fig. 1.7): a highly fortified inner tower. The noblemen and their families lived in the keep. In the case of a siege, the attackers had to defeat each successive protection before being able to reach the commander. Each successive wall was expected to be easier to defend and harder to breach than the previous one.

What are the lessons from this medieval architecture? The first lesson is that the medieval builders of castles knew about Law 1. They knew that they would inevitably lose the outer walls in the case of a siege. Therefore, they used several successive walls. Furthermore, they also knew which asset was the most precious one. In their context, this asset was the life of their Lord and his family (Chap. 2). They put this asset within the ultimate wall, i.e., the keep. Therefore, the lordship benefitted from all the successive defensive layers. This principle of successive in-depth defenses is still valid in our modern IT world. An attacker should have to breach many protections before reaching the most valuable assets.

#### **Rule 1.4: Design In-depth Defense**

The second example is about how to protect data centers, secure rooms, or vaults against physical intrusions. Physical security obeys to the rule of "security in depth." Physical security must create a set of successive zones isolated by physical barriers and gates [93]. Only the users who are the most privileged should access successive zones. The following example illustrates this rule. A facility receives visitors and handles sensitive assets in dedicated areas. Of course, most visitors should never approach these sensitive assets. Physical security may suggest a layout delimiting different zones.

- Zone 0: This zone is the external world where the security of the site does not apply. Everybody can freely access this area. The tenants of the facility have no control over this zone.
- Zone 1: Usually, the site is isolated from the external world (Zone 0) by a fence. Ideally, a clear area should be around the fence. This clear zone creates an unobstructed view to detect illegal intrusion. Furthermore, the fence drives traffic to one or a set of controlled entrances. Physical access controls the entrances to the site. Guards or receptionists monitor this access. Employees have company access badges. Visitors receive dedicated guest access badges.
- Zone 2: In this area, the visitors and employees can interact. Automatic gates control its access from Zone 1. Demonstration rooms and meeting rooms are in this area.
- Zone 3: This area is limited to employees. Visitors should not enter into this zone. Physical access controls admission to this area. Visitors' badges do not grant access to Zone 3. It is only accessible from Zone 2 area.
- Zone 4 (and above): This area is restricted to a limited set of employees. Only employees who need to access it should be granted corresponding access rights.

The access control may be more sophisticated than a badge reader. For instance, it may use two-factor authentication with biometrics. It is only accessible from Zone 3 area. Usually, vaults and data centers are located in such zones. Few people should be granted access to them.

The defense and control mechanisms between the successive layers should not be identical. The defense mechanisms of each zone should be more restrictive than the one protecting the previous zone. The mechanisms act as a sieve. Ideally, the consecutive defenses should feature increasing robustness. Indeed, the goal of the “inner” defense is to stop attackers who succeeded in defeating the “outer” defense. An attacker who manages to reach the “inner” defense already has proven that she is stronger than the “outer” defense. Were the “inner” defense to be weaker or equal to than the “outer” one, then the attacker would also easily defeat the inner defense if defeating the “outer” one. Indeed, in such unrestrained configuration, the only adequate defense would be the outer one. The builders of castles knew this design principle. The inner walls were larger and higher than the outer walls and thus stronger. Each successive defense slowed down the attackers. In network security, the use of firewalls and DMZs is a perfect illustration of in-depth defense (Sect. 8.2.1).

In-depth defense should not be limited to successive layers of increasing similar protection. It should also use a set of complementary defenses. Then, the security widens the defense perimeter. Diversity is a good security attribute.

In 2014, the Council on Cyber Security published the fifth version of its critical security controls [94]. It defines 20 security controls. Any organization should mandatorily establish these twenty control points. The variety of control points ensures that attacks are detected and remedied, guarantees that already compromised systems are identified, and prevents disrupting attacks. They constitute a comprehensive list of in-depth defenses. The ordered twenty critical security controls are as follows:

1. Inventory of authorized and unauthorized devices,
2. Inventory of authorized and unauthorized software,
3. Secure configurations for hardware and software on mobile devices, laptops, workstations, and servers,
4. Continuous vulnerability assessment and remediation,
5. Malware defenses,
6. Application software security,
7. Wireless access control,
8. Data recovery capability,
9. Security skills assessment and appropriate training to fill gaps,
10. Secure configurations for network devices such as firewalls, routers, and switches,
11. Limitation and control of network ports, protocols, and services,
12. Controlled use of administrative privileges,
13. Boundary defense,
14. Maintenance, monitoring, and analysis of audit logs,

15. Controlled access based on the need to know,
16. Account monitoring and control,
17. Data protection,
18. Incident response and management,
19. Secure network engineering,
20. Penetration tests and red team exercises.

An organization that implements all these twenty controls has both an in-depth defense and a wide defense.

Content protection within an audio/video professional postproduction environment is another example of in-depth complementary defenses. In the professional environment, there should ideally be four different types of protection—each fulfilling a different goal. The goals are as follows:

1. Control access to the asset; in this case, the video in preparation,
2. Protect the asset itself,
3. Trace the asset,
4. Limit illegal use of the asset.

*Controlling Access:* The first defense involves controlling access to a digital asset. This control was already in place during the analog era. It uses the principles of physical security described in the previous example, such as guards, physical access controls, and vaults. In the digital world, the type of access control is IT security. Typically, the IT department defines a perimeter that it defends against intruders using firewalls, DMZs, or virtual private networks (VPNs). Within this perimeter, IT segregates the access of corporate users to data using tools such as access control lists and role-based policies. Only authorized corporate users should be able to access the assets legitimately.

*Protecting the Asset:* The second defense targets direct attacks on the asset, such as theft, alteration, or replacement. The deployed tools widely use encryption and cryptographic signature. Encryption enforces the confidentiality of the asset, whereas cryptographic signature enforces its integrity.

*Trace the Asset:* The third barrier complements the previous one. Ultimately, any digital content has to be rendered in the analog world where the protection provided by encryption does not work anymore. Video content has to be displayed on a screen. Audio content has to be played on loudspeakers or earphones. It is prone to being captured by recording devices. Forensic marking appends information about the source that rendered the content. The protection technology used is digital watermarking [95]. A digital watermark is a message embedded into digital content that can be detected or extracted later. Watermarks are imperceptible or perceptible. Imperceptible watermarks are harder to defeat than perceptible ones.

Typical watermark information includes copyright details and the identifier of the expected recipient of the watermarked piece of content. In the case of a leak, watermark detectors can extract this information and thus trace the source of the non-legitimate disclosure. Forensic marking is useful in numerous contexts: spotting leakage in the postproduction environment, “protecting” screeners used for

award selection or reviewing, or detecting the source of illegal rebroadcasting of content by identifying the infringing STB.

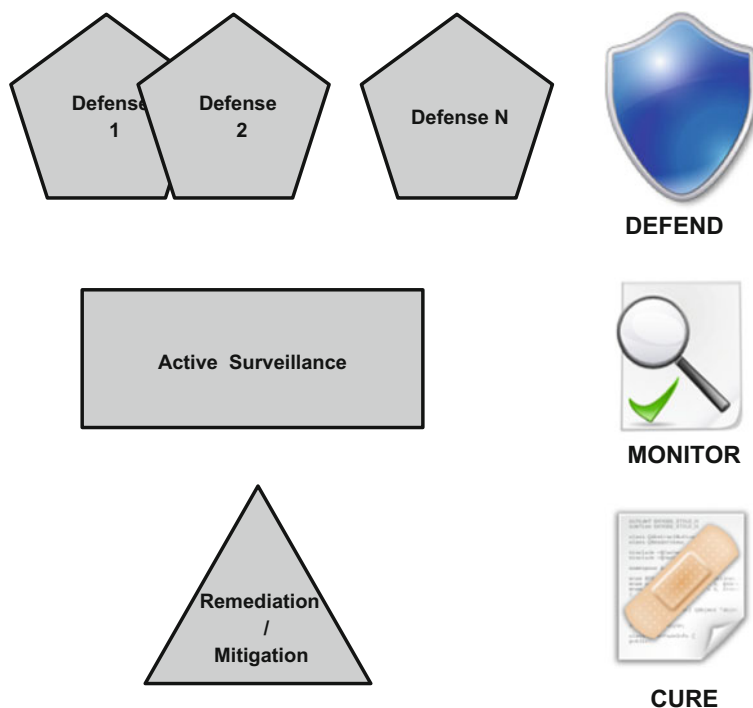
*Thwarting Illegal Distribution:* Unfortunately, as stated by Law 1, content will always leak. So, this fourth barrier attempts to limit the incurred losses. The first step is to detect illegal content. Fingerprinting is the most efficient technology for this operation. A reference database includes fingerprints that contain unique characteristics of every referenced content [96]. These characteristics may use visual hashes, distributions of color, time stamps, or points of interests. Fingerprinting may be audio-based or video-based or employ both media. Then, the system extracts the relevant fingerprints of suspicious content and compares them with those in the reference database of copyrighted content.

Once illegal content is identified, the corrective action taken depends on the context. Currently, some user-generated content (UGC) sites filter the submitted candidates. In the case of peer-to-peer (P2P) networks and file-sharing sites, a takedown notification may be sent to the sharers. In this context, fingerprinting is a superior solution than identification using cryptographic hash values because fingerprinting is robust to geometrical modifications, mashups, and camcording.

The second step is to slow down the dissemination of the stolen content. The first objective is to deter the downloaders, for instance, by delivering a bad piece of content instead of the expected one. Typical poisoning techniques spread decoys, fake content, or even encrypted content that requires payment. The second objective is to inhibit access either by slowing down the bandwidth or by routing the requester to controlled peers.

Using this second example, we can define an in-depth defensive framework as illustrated in Fig. 1.8. The proposed framework uses three complementary actions.

1. *Defend:* The assets have to be protected by any means. The employed defensive tactics should be numerous, varied, and complementary. One unique class of protection is not sufficient because it will be defeated eventually. This defeat is the inevitable consequence of Law 1. The defenses should encompass physical security, network security (such as firewalls, intrusion detection systems (IDSs), and DLPs), antivirus and anti-malware software, encryption, authentication, and reputation-based protection.
2. *Monitor:* As ultimately the attacker will win, it is crucial to know as soon as possible when she succeeded and what benefits she gained from her exploit. Acquiring this knowledge requires carefully monitoring the system (Law 9: *Quis Custodiet Ipsos Custodes?*). The purpose is to know which asset was affected and how (theft, alteration, destruction, or substitution). As speed is of essence, practitioners should not wait until a breach happens to trigger the monitoring; rather, monitoring should be active and continuously enabled.
3. *Cure:* This action is, unfortunately, sometimes forgotten. Limiting the impact of the exploit is paramount. Defining strategies of mitigation for every critical asset should be part of any strategy of defense. Furthermore, once the exploit is analyzed, a new set of protections should be designed and deployed, or the existing ones should be enhanced to defeat this class of attack in the future.



**Fig. 1.8** Framework for active, in-depth defense

### 1.3.4 Backup

Around 2005, a new type of malware, coined ransomware, appeared. The noun ransomware is from the combination of two nouns: ransom and malware. Ransomware is a kind of malware that disables some functionality of the infected system [97]. Then, the ransomware requests payment to reactivate the blocked feature. The first generation of ransomware encrypted the files of the infected computer. Since 2009, a new generation of ransomware has also blocked the user's display<sup>20</sup> rather than just encrypting files [99]. More recently, a new generation of ransomware blocked the functionality of some applications such as browsers. In 2014, ransomware extended its target field from computers to specialized hardware. For instance, the ransomware Synlocker attacked Synology's Network as Storage solutions [100], encrypting remotely stored files. In 2015, ransomware extended its reach to Web sites. A new type of ransomware started progressively to encrypt the

<sup>20</sup>In some reported cases, ransomware blocked the screen by displaying pornographic pictures or even pedophilia [98] to shame the blackmailed person. The authors of such ransomware expected that the hacked person would not dare to call for help. It seems that this tactic was rather efficient as the ratio of paid ransoms was rather high. This is a very nice, dirty piece of social engineering.

database used by Web sites. During this encryption period, the ransomware decrypted the data so that the administrators of the infected Web sites were not aware of the ongoing attack. This lasted several weeks. Before the full backup, blackmailers removed the decryption key and asked for the ransom. Unfortunately, the incremental backup contained encrypted data [101], and the decryption was inactive. The ransomed Web site lost all data produced since the last full backup.

Typically, ransomware uses three stages:

- Seek a target: This is the usual step to finding a victim. Usually, the attacker randomly looks for an easy target to penetrate. The targets are often individuals or small companies rather than larger organizations, which are expected to be better protected.
- Enable blackmailing by disabling some functionality such as access to the files.
- Claim for the ransom to reactivate the blocked feature.

The operating mode for claiming the ransom is interesting. The first phase explains to the attacked user why he does not have access anymore to his computer, smartphone, or files. The initial strategy was to announce openly that the attacker had taken control of victim's device. Recently, ransomware used another technique by "impersonating" a legal authority [for instance, the Federal Bureau of Investigation (FBI)]. The ransomware claimed that the computer had been blocked due to illegal activities conducted from the machine [102, 103] (Fig. 1.9). In the second phase, the blackmailer proposes a method of anonymous payment. The blackmailer does not want the payment to trace back to her to avoid retaliation and potential prosecution. Anonymous payment is the trickiest part of her attack. Obviously, traditional electronic payment means such as credit cards, electronic transfers, or PayPal cannot be used. Investigators could find who received the money. The recent advent of prepaid electronic payment such as MoneyPak and WebMoney and digital currency such as BitCoin eliminates this risk. The recipient of the payment remains entirely anonymous.<sup>21</sup> Therefore, many instances of ransomware use these new types of payments. Once the victim has paid, he waits (often uselessly) for the blackmailer to enable again the blocked functionality. In 2014, a UK online survey demonstrated that about 40 % of the victims of CryptoLocker<sup>22</sup> paid the ransom [105]. In 2015, an annual report evaluated the return on investment (ROI) of ransomware at 1425 % [106]. This is an excellent lucrative business.

**The Devil Is in the Detail** On June 8, 2008, Kaspersky Lab, the Russian antivirus editor, detected a variant of the virus GPcode. GPcode is a ransomware that has many variants. The initial versions mainly renamed some files. They replaced the original names by random-looking names. Later,

<sup>21</sup>Indeed, cryptocurrencies such as Pecunix, AlertPay, PPcoin, Litecoin, Feathercoin, or Zerocoin are the payment methods used by the black market of the Darknet [104].

<sup>22</sup>CryptoLocker is the most well-known ransomware.





Fig. 1.9 An example of a ransomware screen

variants became more sophisticated and started to employ cryptography, often not very smartly. This newly discovered variant encrypts some data files on the victim's hard drive, renames them with extension `._CRYPT`, and adds the file `!_README!.txt` in their folder. Then, it displays a ransom message announcing the encryption of the data and giving a contact email. The ransomware claims to use RSA-1024. Thus, it is out of the reach of brute force attacks against the private key. The pirated individual must contact the pirate, pay the ransom, and may receive in return a decryption tool.

Through reverse engineering, Kaspersky Lab extracted the public keys used by the ransomware to encrypt the files. The ransomware uses two public keys depending on the version of the OS. On June 10, 2008, Kaspersky Lab called for the help of the cryptographic community to try to crack the private key. This attempt was illusory. It would have required too much calculation power (or it would mean that RSA-1024 is not safe anymore). Moreover, there were two keys to crack!

One week later, Kaspersky Lab withdrew their challenge. Nevertheless, thanks to one “common” mistake of the ransomware's authors, there may be some hope for careless users who did not back up their data. When encrypting a file, the virus creates a new file that it renames with the expected extension and then deletes the original file. The method used was a standard deletion rather than a secure deletion. Indeed, it is common knowledge (at least in the security community) that a simple deletion does not erase the file [107]. It mainly clears the data fields of the file system's indexing tables. Consequently, the deleted files are still present on the hard drive as long as a new file has not overwritten them. Therefore, if there has not been too much activity on the hard drive since the infection, usual commercial recovery tools may be able to retrieve the “deleted” files. Kaspersky Lab proposed such a tool issued by the open source community. If the ransomware would have used secure deletion, then this fix would have been impossible. Usually,

secure deletion requires overwriting several times the physical location on the hard drive of the deleted sections with random data. The US National Institute of Standards and Technology (NIST) published a guideline for media sanitization (still in draft version) [108].

*Lesson:* Even the bad guys make mistakes. Law 1 also applies to viruses and malware. Furthermore, deleting data is not a straightforward operation (Sect. 6.3.3).

In the previous example, it was possible to retrieve the clear files. Sometimes, it is also possible to retrieve the keys used by the ransomware. In June 2014, the FBI brought down the infrastructure of the Gameover Zeus botnet and the initial CryptoLocker. The investigation teams seized the databases used by the malware. Among them was the database of RSA private keys that encrypt the AES encryption keys used by CryptoLocker. Using this information, security companies FireEye and Foxit provided an online tool that potentially may decrypt CryptoLocker-encrypted files [109]. Unfortunately, access to the decryption key is not available for most ransoms. The best and unique response to ransomware is still periodic, frequent backup.

### **Rule 1.5: Back up Your Files**

This book will never repeat this recommendation enough: Back up your data. Usually, people believe that the purpose of a backup is merely to recover from a crashed system. A backup is also the ultimate tool for recovering from a successful attack. Of course, there will be disenchantment of the affected people and loss of productivity, but at least the entire past work will not be lost. The recovery is only possible if the backup is regularly performed and with a sufficient periodicity. Furthermore, as some recent instances of ransomware can reach networked directories [110], it is safer to use both non-local physical storage and cloud-based storage. In other words, the best defense is to use an air-gapped backup. Malware cannot step over air gaps.

On June 17, 2014, the company Code Spaces suffered from a well-orchestrated distributed denial of service (DDoS) attack [111]. The blackmailer(s) requested a ransom to stop the attack. The problem was not limited to the denial of service. The attackers succeeded in getting access to the company's Amazon EC2 control panel. Thus, they owned the cloud infrastructure.<sup>23</sup> As the management team of Code Spaces refused to pay the ransom, the attackers started to delete large chunks of random file directories and backups. Only after Code Spaces succeeded in retrieving access to its control panel, could it assess that it had lost most of the software repositories of its customers, as well as all the backups irremediably. With this disaster, Code Spaces could never get back into business. This story highlights

---

<sup>23</sup>Amazon Elastic Compute Cloud (Amazon EC2) is a Web service that provides resizable computational capacity in a cloud.

the need for air-gapped backup. If an attacker who penetrated a system also gains access to the backups, then these backups only play their safety role but not their security role.

After a severe attack, the best solution is to rebuild the hacked system from scratch. First, install a new genuine OS, updated with the latest security patches. Then, install the usual security tools such as an antivirus and a personal firewall. After setting up a genuine, protected environment, install the applications from genuine sources and apply all applicable security patches. Once the applications are properly running, restore the data from the backup. It may be wise to sanitize the backup data with a scan by an up-to-date antivirus. Some backed up data may have been infected.

## 1.4 Summary

### **Law 1: Attackers Will Always Find Their Way**

No secure system is infallible. Any secure system is doomed to fail. Attackers will always find a way to defeat it. Security designers must not deny this fact, but rather put this heuristic at the heart of their design.

**Rule 1.1: Always Expect the Attacker to Push the Limits** Any design operates within a set of limits defined by its initial requirements. The system should work correctly within these boundaries. An attacker may attempt to operate outside these limits to get unexpected behavior. The security designer should ensure either that these limits are out of reach or at least that the system should detect the violation of these boundaries to react accordingly.

**Rule 1.2: Responsibly Publish Vulnerabilities** Publishing vulnerabilities is one of the best methods to reach a safer cyber world. Not only will the solution provider close the holes, but the publication of the vulnerability will also educate the designers. Obscurity is dangerous for security (Chap. 3). Nevertheless, implementers must have a reasonable amount of time to fix the issue before the public disclosure of the vulnerability.

**Rule 1.3: Design Renewable Systems** As any system will be broken, the designed system must be ready to survive by the updating of its defense mechanisms. Without renewability, the system will be definitively dead.

**Rule 1.4: Design In-depth Defense** As any defense will fail, a secure system should implement many defenses. It should construct successive obstacles that the attacker has to cross successfully. Diversity in protection makes the exploit harder to perform.

**Rule 1.5: Back up Your Files** As any system will be broken one day, data may be corrupted or lost. Regular, frequent air-gapped backup of all non-constructible data is the ultimate defense.

Ten Laws for Security

Diehl, E.

2016, XIX, 281 p. 42 illus., 3 illus. in color., Hardcover

ISBN: 978-3-319-42639-6