Chapter 1. The History of Software Security

Before delving into actual offensive and defensive security techniques, it is important to have at least some understanding of software security's long and interesting history. A brief overview of major security events in the last one hundred years should be enough to give you an understanding of the foundational technology underlying today's web applications. Furthermore, it will show off the ongoing relationship between the development of security mechanisms and the improvisation of forward-thinking hackers looking for opportunities to break or bypass those mechanisms.

The Origins of Hacking

In the past two decades, hackers have gained more publicity and notoriety than ever before. As a result, it's easy for anyone without the appropriate background to assume that hacking is a concept closely tied to the internet and that most hackers emerged in the last 20 years.

But that's only a partial truth. While the number of hackers worldwide has definitely exploded with the rise of the World Wide Web, hackers have been around since the middle of the 20th century—potentially even earlier depending on what you define as "hacking." Many experts debate the decade that marks the true origin of modern hackers because a few significant events in the early 1900s showed significant resemblance to the hacking you see today.

For example, there were specific isolated incidents that would likely qualify as hacking in the 1910s and 1920s, most of which involved tampering with Morse code senders and receivers, or interfering with the transmis-

sion of radio waves. However, while these events did occur, they were not common, and it is difficult to pinpoint large-scale operations that were interrupted as a result of the abuse of these technologies.

It is also important to note that I am no historian. I am a security professional with a background in finding solutions to deep architectural and code-level security issues in enterprise software. Prior to this, I spent many years as a software engineer writing web applications in various languages and frameworks. I continue writing software today in the form of security automation in addition to contributing to various projects on my own time as a hobby. This means that I am not here to argue specifics or debate alternative origin stories. Instead, this section is compiled based on many years of independent research, with the emphasis being on the lessons we can extract from these events and apply today.

Because this chapter is not intended to be a comprehensive overview, but instead a reference for critical historical events, we begin our timeline in the early 1930s. Now, without further interruption, let's examine a number of historical events that helped shape the relationship between hackers and engineers today.

The Enigma Machine, Circa 1930

The *Enigma machine* used electric-powered mechanical rotors to both encrypt and decrypt text-based messages sent over radio waves (see Figure 1-1). The device had German origins and would become an important technological development during the Second World War.

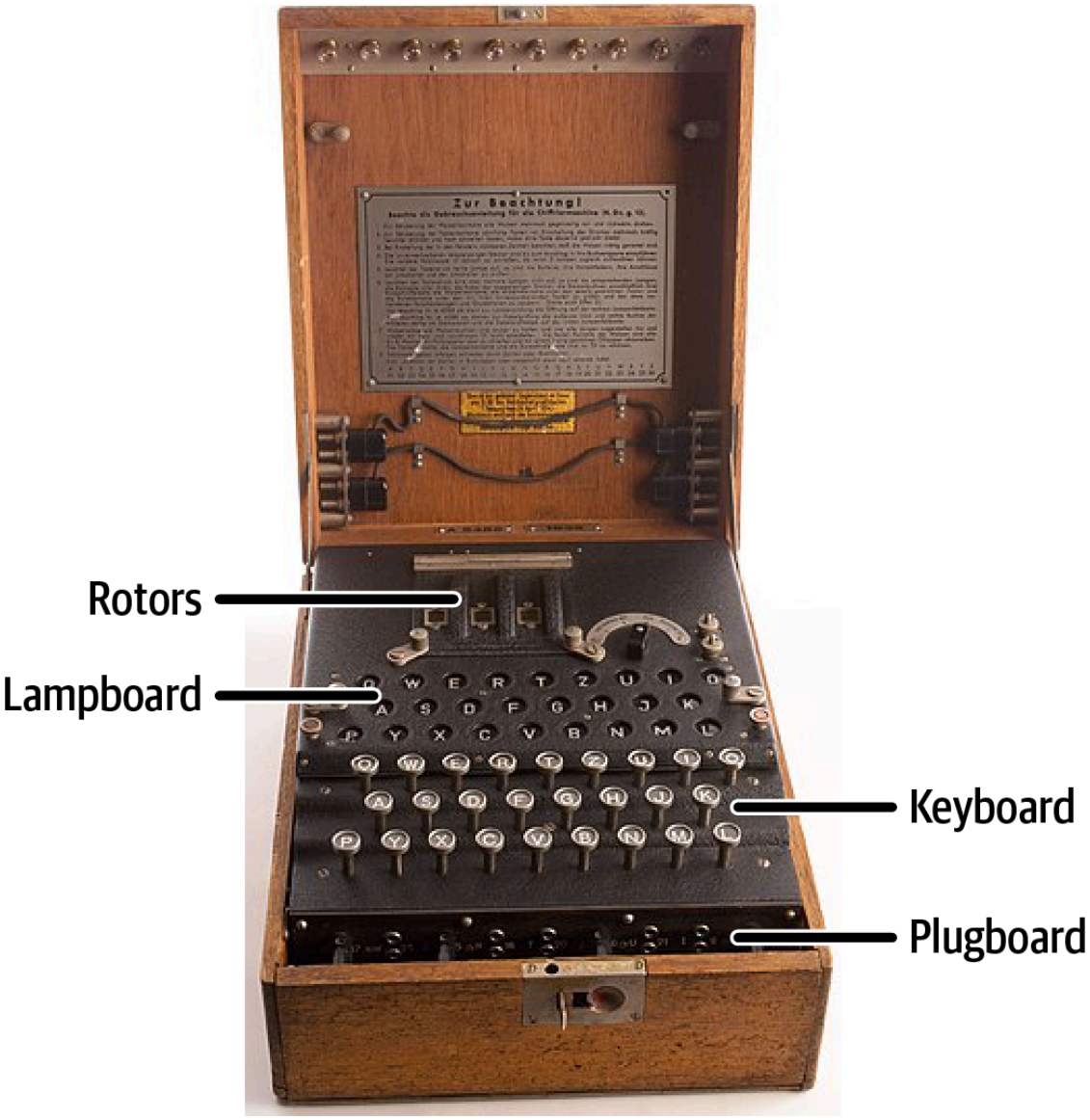

Figure 1-1. The Enigma machine

The device looked like a large square or rectangular mechanical typewriter. On each key press, the rotors would move and record a seemingly random character that would then be transmitted to all nearby Enigma machines. However, these characters were not random; they were defined by the rotation of the rotor and a number of configuration options that could be modified at any time on the device. Any Enigma machine with a specific configuration could read or “decrypt” messages sent from another machine with an identical configuration. This made the Enigma machine extremely valuable for sending crucial messages while avoiding interception.

While a sole inventor of the rotary encryption mechanism used by the machine is hard to pinpoint, the technology was popularized by a two-man company called Chiffriermaschinen AG based in Germany. In the

1920s, Chiffriermaschinen AG traveled throughout Germany demonstrating the technology, which led to the German military adopting it in 1928 to secure top-secret military messages in transit.

The ability to avoid the interception of long-distance messages was a radical development that had never before been possible. In the software world of today, the interception of messages is still a popular technique that hackers try to employ, often called a *man-in-the-middle* attack. Today's software uses similar (but much more powerful) techniques to those that the Enigma machine used a hundred years ago to protect against such attacks.

While the Enigma machine was an incredibly impressive technology for its time, it was not without flaws. Because the only criterion for interception and decryption was an Enigma machine with an identical configuration to the sender, a single compromised configuration log (or *private key*, in today's terms) could render an entire network of Enigma machines useless.

To combat this, any groups sending messages via the Enigma machine changed their configuration settings on a regular basis. Reconfiguring Enigma machines was a time-consuming process. First, the configuration logs had to be exchanged in person, as secure ways of sharing them remotely did not yet exist. Sharing configuration logs between a network of two machines and two operators might not be difficult. But a larger network, say 20 machines, required multiple messengers to deliver the configuration logs—each increasing the probability of a configuration log being intercepted and stolen, or potentially even leaked or sold.

The second problem with sharing configuration logs was that manual adjustments to the machine itself were required for the Enigma machine to be able to read, encrypt, and decrypt new messages sent from other Enigma machines. This meant that a specialized and trained staff member had to be present in case a configuration update was needed. This all occurred in an era prior to software, so these configuration adjustments required tampering with the hardware and adjusting the physical layout

and wiring of the plugboard. The adjuster needed a background in electronics, which was very rare in the early 1900s.

As a result of how difficult and time-consuming it was to update these machines, updates typically occurred on a monthly basis—daily for mission-critical communication lines. If a key was intercepted or leaked, all transmissions for the remainder of the month could be intercepted by a malicious actor—the equivalent of a hacker today.

The type of encryption these Enigma machines used is now known as a symmetric key algorithm, which is a special type of cipher that allows for the encryption and decryption of a message using a single cryptographic key. This family of encryption is still used today in software to secure data in transit (between sender and receiver), but with many improvements on the classic model that gained popularity with the Enigma machine.

In software, keys can be made much more complex. Modern key generation algorithms produce keys so complex that attempting every possible combination (*brute forcing* or *brute force attack*) with the fastest possible modern hardware could easily take more than a million years. Additionally, unlike the Enigma machines of the past, software keys can change rapidly.

Depending on the use case, keys can be regenerated at every user session (per login), at every network request, or at a scheduled interval. When this type of encryption is used in software, a leaked key might expose you for a single network request in the case of per-request regeneration, or worst-case scenario, a few hours in the case of per-login (per-session) regeneration.

If you trace the lineage of modern cryptography far back, you will eventually reach World War II in the 1930s. It's safe to say that the Enigma machine was a major milestone in securing remote communications. From this, we can conclude that the Enigma machine was an essential development in what would later become the field of software security.

The Enigma machine was also an important technological development for those who would be eventually known as "hackers." The adoption of

Enigma machines by the Axis Powers during World War II resulted in extreme pressure for the Allies to develop encryption-breaking techniques. General Dwight D. Eisenhower himself claimed that doing so would be essential for victory against the Nazis.

In September of 1932, a Polish mathematician named Marian Rejewski was provided a stolen Enigma machine. At the same time, a French spy named Hans-Thilo Schmidt was able to provide him with valid configurations for September and October of 1932. This allowed Marian to intercept messages from which he could begin to analyze the mystery of Enigma machine encryption.

Marian was attempting to determine how the machine worked, both mechanically and mathematically. He wanted to understand how a specific configuration of the machine's hardware could result in an entirely different encrypted message being output.

Marian's attempted decryption was based on a number of theories as to what machine configuration would lead to a particular output. By analyzing patterns in the encrypted messages and coming up with theories based on the mechanics of the machine, Marian and two coworkers, Jerzy Różycki and Henryk Zygalski, eventually reverse engineered the system. With the deep understanding of Enigma rotor mechanics and board configuration that the team developed, they were able to make educated guesses as to which configurations would result in which encryption patterns. They could then reconfigure a board with reasonable accuracy and, after several attempts, begin reading encrypted radio traffic. By 1933 the team was intercepting and decrypting Enigma machine traffic on a daily basis.

Much like the hackers of today, Marian and his team intercepted and reverse engineered encryption schemes to get access to valuable data generated by a source other than themselves. For these reasons, I would consider Marian Rejewski and the team assisting him as some of the world's earliest hackers.

In the following years, Germany would continually increase the complexity of its Enigma machine encryption. This was done by gradually increas-

ing the number of rotors required to encrypt a character. Eventually the complexity of reverse engineering a configuration would become too difficult for Marian's team to break in a reasonable time frame. This development was also important because it provided a look into the ever-evolving relationship between hackers and those who try to prevent hacking.

This relationship continues today, as creative hackers continually iterate and improve their techniques for breaking into software systems. And on the other side of the coin, smart engineers are continually developing new techniques for defending against the most innovative hackers.

Automated Enigma Code Cracking, Circa 1940

Alan Turing was an English mathematician who is best known for his development of a test known today as the "Turing test." The Turing test was developed to rate conversations generated by machines based on the difficulty in differentiating those conversations from the conversations of real human beings. This test is often considered to be one of the foundational philosophies in the field of AI.

While Alan Turing is best known for his work in AI, he was also a pioneer in cryptography and automation. In fact, prior to and during World War II, Alan's research focus was primarily on cryptography rather than AI. Starting in September of 1938, Alan worked part time at the Government Code and Cypher School (GC&CS). GC&CS was a research and intelligence agency funded by the British Army, located in Bletchley Park, England.

Alan's research primarily focused on the analysis of Enigma machines. At Bletchley Park, Alan researched Enigma machine cryptography alongside his then-mentor Dilly Knox, who at the time was an experienced cryptographer.

Much like the Polish mathematicians before them, Alan and Dilly wanted to find a way to break the (now significantly more powerful) encryption

of the German Enigma machines. Due to their partnership with the Polish Cipher Bureau, the two gained access to all of the research Marian's team had produced nearly a decade earlier. This meant they already had a deep understanding of the machine. They understood the relationship between the rotors and wiring, and knew about the relationship between the device configuration and the encryption that would be output (Figure 1-2).

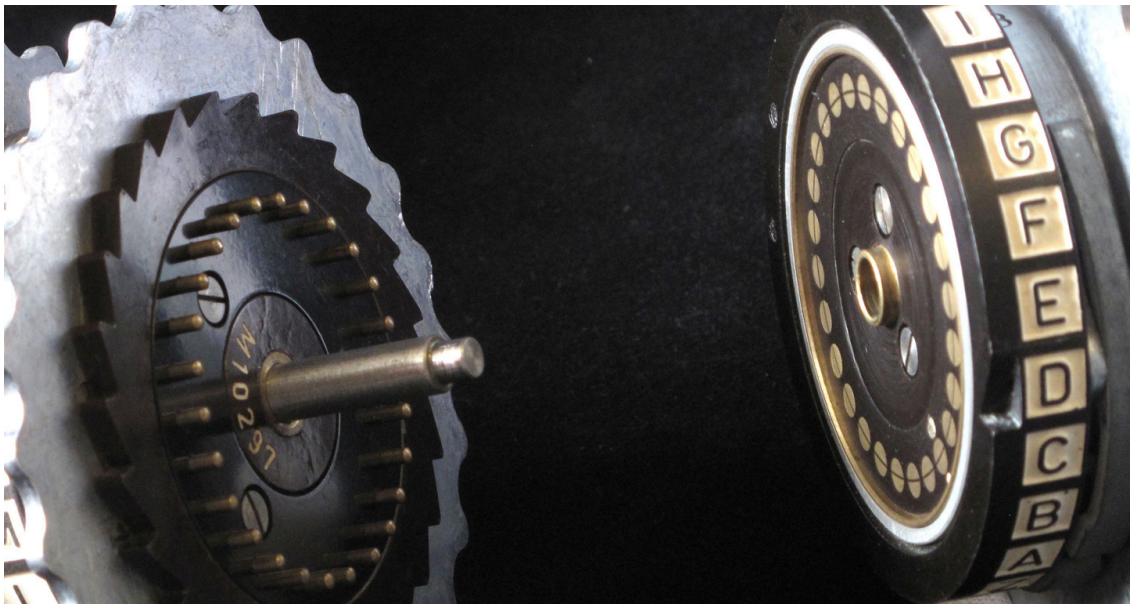

Figure 1-2. A pair of Enigma rotors used for calibrating the Enigma machine's transmission configuration, an analog equivalent of changing a digital cipher's primary key

Marian's team was able to find patterns in the encryption that allowed them to make educated guesses regarding a machine's configuration. But this was not scalable now that the number of rotors in the machine had increased as much as tenfold. In the amount of time required to try all of the potential combinations, a new configuration would have already been issued. Because of this, Alan and Dilly were looking for a different type of solution; a solution that would scale and that could be used to break new types of encryption. They wanted a general-purpose solution rather than a highly specialized one.

A bombe was an electric-powered mechanical device that attempted to automatically reverse engineer the position of mechanical rotors in an Enigma machine based on mechanical analysis of messages sent from such machines (see Figure 1-3).

The first bombes were built by the Polish in an attempt to automate Marian's work. Unfortunately, these devices were designed to determine the configuration of Enigma machines with very specific hardware. In particular, they were ineffective against machines with more than three rotors. Because the Polish bombe could not scale against the development of more complex Enigma machines, the Polish cryptographers eventually went back to using manual methods for attempting to decipher German wartime messages.

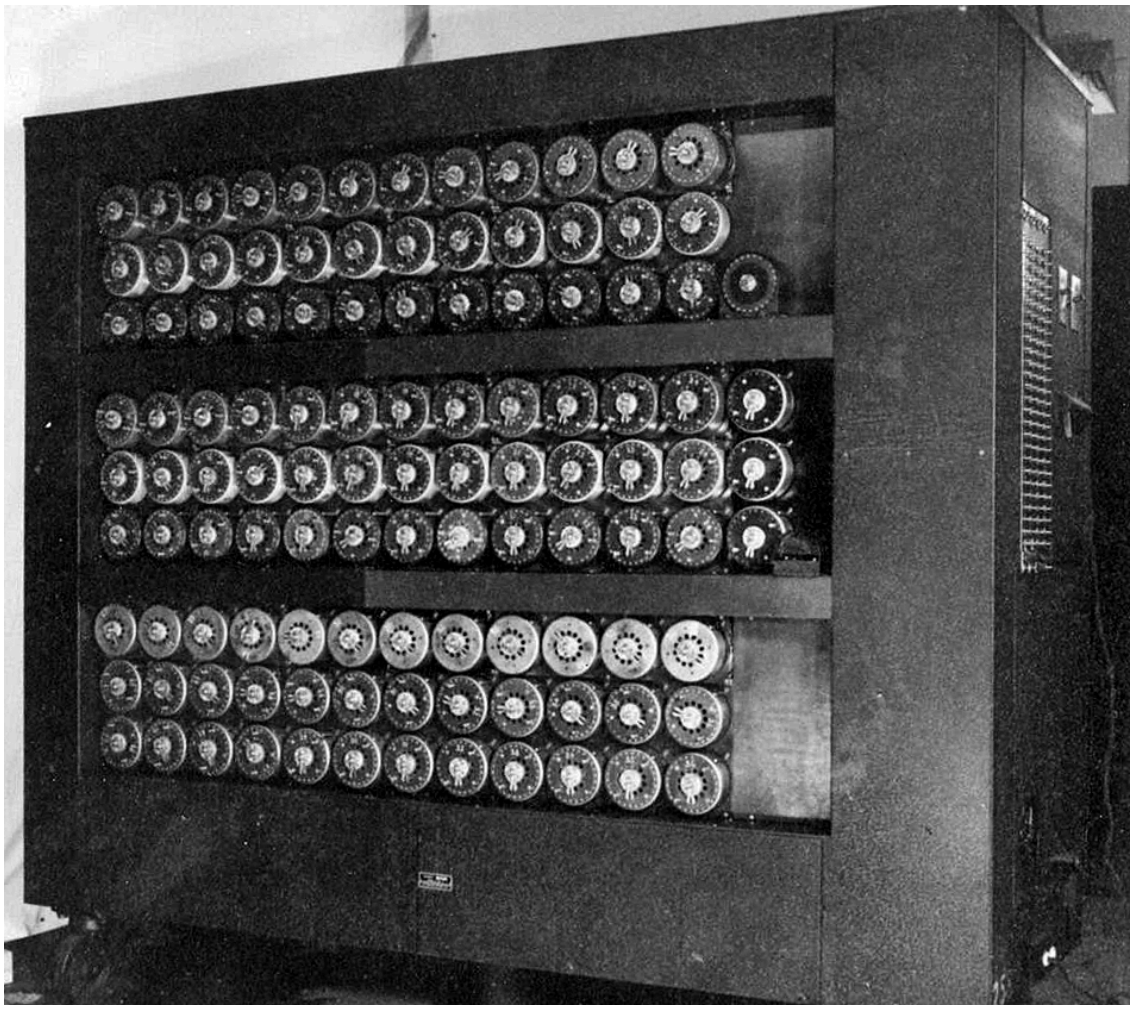

Figure 1-3. An early Bletchley Park bombe used during World War II (note the many rows of rotors used for rapidly performing Enigma configuration decryption)

Alan Turing believed that the original machines failed because they were not written in a general-purpose manner. To develop a machine that could decipher any Enigma configuration (regardless of the number of rotors), he began with a simple assumption: in order to properly design an algorithm to decrypt an encrypted message, you must first know a word or phrase that exists within that message and its position.

Fortunately for Alan, the German military had very strict communication standards. Each day, a message was sent over encrypted Enigma radio waves containing a detailed regional weather report. This is how the German military ensured that all units knew the weather conditions without sharing them publicly to anyone listening on the radio. The Germans did not know that Alan's team would be able to reverse engineer the purpose and position of these reports.

Knowing the inputs (weather data) being sent through a properly configured Enigma machine made algorithmically determining the outputs much easier. Alan used this newfound knowledge to determine a bombe configuration that could work independently of the number of rotors that the Enigma machine it was attempting to crack relied on.

Alan requested a budget to build a bombe that would accurately detect the configuration requirements needed to intercept and read encrypted messages from German Enigma machines. Once the budget was approved, Alan constructed a bombe composed of 108 drums that could rotate as fast as 120 RPM. This machine could run through nearly 20,000 possible Enigma machine configurations in just 20 minutes. This meant that any new configuration could be rapidly compromised. Enigma encryption was no longer a secure means of communication.

Today Alan's reverse-engineering strategy is known as a *known plaintext attack* or KPA. It's an algorithm that is made much more efficient by being provided with prior input/output data. Similar techniques are used by modern hackers to break encryption on data stored or used in software. The machine Alan built marked an important point in history, as it was one of the first automated hacking tools ever built.

Telephone "Phreaking," Circa 1950

After the rise of the Enigma machine in the 1930s and the cryptographic battle that occurred between major world powers, the introduction of the telephone is the next major event in our timeline. The telephone allowed everyday people to communicate with each other over large distances

and at rapid speed. As telephone networks grew, they required automation in order to function at scale.

In the late 1950s, telecoms like AT&T began implementing new phones that could be automatically routed to a destination number based on audio signals emitted from the phone unit. Pressing a key on the phone pad emitted a specific audio frequency that was transmitted over the line and interpreted by a machine in a switching center. A switching machine translated these sounds into numbers and routed the call to the appropriate receiver.

This system was known as *tone dialing*, and it was an essential development that telephone networks at scale could not function without. Tone dialing dramatically reduced the overhead of running a telephone network because the network no longer needed an operator to manually connect every call. Instead, one operator overseeing a network for issues could manage hundreds of calls in the same time as one call previously took.

Within a short period of time, small groups of people began to realize that any systems built on top of the interpretation of audio tones could be easily manipulated. Simply learning how to reproduce identical audio frequencies next to the telephone receiver could interfere with the intended functionality of the device. Hobbyists who experimented with manipulating this technology eventually became known as *phreakers*—an early type of hacker specializing in breaking or manipulating telephone networks. The true origin of the term *phreaking* is not known, though it has several generally accepted possible origins. It is most often thought to be derived from two words, “freaking” and “phone.”

There is an alternative suggested derivation that I believe makes more sense. I believe that the term phreaking originated from “audio frequency” in response to the audio signaling languages that phones of the time used. I believe this explanation makes more sense since the origin of the term is very close chronologically to the release of AT&T’s original tone dialing system. Prior to tone dialing, telephone calls were much

more difficult to tamper with because each call required an operator to connect the two lines.

We can trace phreaking back to several events, but the most notorious case of early phreaking was the discovery and utilization of the 2600 Hz tone. A 2600 Hz audio frequency was used internally by AT&T to signal that a call had ended. It was essentially an "admin command" built into the original tone dialing system. Emitting a 2600 Hz tone stopped a telecom switching system from realizing that a call was still open (logged the call as ended, although it was still ongoing). This allowed expensive international calls to be placed without a bill being recorded or sent to the caller.

The discovery of the 2600 Hz tone is often attributed to two events. First, a young boy named Joe Engressia was known to have a whistling pitch of 2600 Hz and would reportedly show off to his friends by whistling a tone that could prevent phones from dialing. Some consider Joe to be one of the original phone phreakers, although his discovery came by accident.

Later on, a friend of Joe Engressia's named John Draper discovered that toy whistles included in Cap'n Crunch cereal boxes mimicked a 2600 Hz tone. Careful usage of the whistle could also generate free long-distance phone calls using the same technique. Knowledge of these techniques spread throughout the Western world, eventually leading to the generation of hardware that could match specific audio frequencies with the press of a button.

The first of these hardware devices was known as a *blue box*. Blue boxes played a nearly perfect 2600 Hz signal, allowing anyone who owned one to take advantage of the free calling bug inherent in telecom switching systems. Blue boxes were only the beginning of automated phreaking hardware, as later generations of phreakers would go on to tamper with pay phones, prevent billing cycles from starting without using a 2600 Hz signal, emulate military communication signals, and even fake caller ID.

From this we can see that architects of early telephone networks only considered normal people and their communication goals. In the software world of today, this is known as "best-case scenario" design.

Designing based off of this was a fatal flaw, but it would become an important lesson that is still relevant today: always consider the worst-case scenario first when designing complex systems.

Eventually, knowledge of weaknesses inherent in tone dialing systems became more widely known, which led to budgets being allocated to develop countermeasures to protect telecom profits and call integrity against phreakers.

Anti-Phreaking Technology, Circa 1960

In the 1960s, phones were equipped with a new technology known as dual-tone multifrequency (DTMF) signaling. DTMF was an audio-based signaling language developed by Bell Systems and patented under the more commonly known trademark, “Touch Tones.” DTMF was intrinsically tied to the phone dial layout we know today that consists of three columns and four rows of numbers. Each key on a DTMF phone emitted two very specific audio frequencies, versus a single frequency like the original tone dialing systems.

This table represents the “Touch Tones,” or sounds, (in hertz) that older telephones made on keypress:

1	2	3	**(697 Hz)**
4	5	6	**(770 Hz)**
7	8	9	**(852 Hz)**
*	0	#	**(941 Hz)**
(1209 Hz)	**(1336 Hz)**	**(1477 Hz)**	

The development of DTMF was due largely to the fact that phreakers were taking advantage of tone dialing systems because of how easy those systems were to reverse engineer. Bell Systems believed that because the DTMF system used two very different tones at the same time, it would be much more difficult for a malicious actor to take advantage of it.

DTMF tones could not be easily replicated by a human voice or a whistle, which meant the technology was significantly more secure than its predecessor. DTMF was a prime example of a successful security development introduced to combat phreakers, the hackers of that era.

The mechanics of how DTMF tones are generated are pretty simple. Behind each key is a switch that signals to an internal speaker to emit two frequencies: one frequency based on the row of the key and one frequency based on the column. Hence the use of the term *dual-tone*.

DTMF was adopted as a standard by the International Telecommunication Union (ITU) and would later go on to be used in cable TV (to specify commercial break times), in addition to phones.

DTMF is an important technological development because it shows that systems can be engineered to be more difficult to abuse if proper planning is taken. Note that these DTMF tones would eventually be duplicated as well, but the effort required would be significantly greater. Eventually switching centers would move to digital (versus analog) inputs, which eliminated nearly all phreaking.

The Origins of Computer Hacking, Circa 1980

In 1976, Apple released the Apple 1 personal computer. This computer was not configured out of the box and required the buyer to provide a number of components and connect them to the motherboard. Only a few hundred of these devices were built and sold.

In 1982, Commodore International released its competitor device. This was the Commodore 64, a personal computer that was completely configured right out of the box. It came with its own keyboard, could support audio, and could even be used with multicolor displays.

The Commodore 64 would go on to sell nearly 500,000 units per month until the early 1990s. From this point forward, the sales trend for per-

sonal computers would continually increase year over year for several decades to come. Computers soon became a common tool in households as well as businesses and took over common repetitive tasks, such as managing finances, human resources, accounting, and sales.

In 1983, Fred Cohen, an American computer scientist, created the very first computer virus. The virus he wrote was capable of making copies of itself and was easily spread from one personal computer to another via floppy disk. He was able to store the virus inside a legitimate program, masking it from anyone who did not have source code access. Fred Cohen later became known as a pioneer in software security, demonstrating that detecting viruses from valid software with algorithms was almost impossible.

A few years later, in 1988, another American computer scientist named Robert Morris was the first person to ever deploy a virus that infected computers outside of a research lab. The virus became known as the *Morris Worm*, with "worm" being a new phrase used to describe a self-replicating computer virus. The Morris Worm spread to about 15,000 network-attached computers within the first day of its release.

For the first time in history, the US government stepped in to consider official regulations against hacking. The US Government Accountability Office (GAO) estimated the damage caused by this virus at $10,000,000. Robert received three years of probation, four hundred hours of community service, and a fine of $10,050. This would make him the first convicted hacker in the United States.

These days, most hackers do not build viruses that infect operating systems, but instead target web browsers. Modern browsers provide extremely robust sandboxing that makes it difficult for a website to run executable code outside of the browser (against the host operating system) without explicit user permission.

Although hackers today are primarily targeting users and data that can be accessed via web browser, there are many similarities to hackers that targeted the OS. Scalability (jumping from one user to another) and cam-

ouflaging (hiding malicious code inside of a legitimate program) are techniques employed by attacks against web browsers.

Today, attacks often scale by distribution through email, social media, or instant messaging. Some hackers even build up legitimate networks of real websites to promote a single malicious website.

Oftentimes, malicious code is hidden behind a legitimate-looking interface. Phishing (credential stealing) attacks occur on websites that look and feel identical to social media or banking sites. Browser plug-ins are frequently caught stealing data, and sometimes hackers even find ways to run their own code on websites they do not own.

The Rise of the World Wide Web, Circa 2000

The World Wide Web (WWW) sprang up in the 1990s, but its popularity began to explode at the end of the 1990s and in the early 2000s.

In the 1990s, the web was almost exclusively used as a way of sharing documents written in HTML. Websites did not pay attention to user experience, and very few allowed the user to send any inputs back to the server in order to modify the flow of the website. Figure 1-4 shows an Apple.com website from 1997 with purely informational data.

The early 2000s marked a new era for the internet because websites began to store user-submitted data and modify the functionality of the site based on user input. This was a key development, later known as *Web 2.0*. Web 2.0 websites allowed users to collaborate with each other by submitting their inputs over Hypertext Transport Protocol (HTTP) to a web server, which would then store the inputs and share them with fellow users upon request.

This new ideology in building websites gave birth to social media as we know it today. Web 2.0 enabled blogs, wikis, media-sharing sites, and more.

Figure 1-4. Apple.com website, July 1997; the data presented is purely informational and a user cannot sign up, sign in, comment, or persist any data from one session to another

This radical change in web ideology caused the web to change from a document-sharing platform to an application distribution platform. Figure 1-5 shows an Apple.com storefront from 2007 where you can buy things. Note the account link in the upper right-hand corner, suggesting that the website had support for user accounts and data persistence. The account link existed in previous iterations of the Apple website in the 2000s, but in 2007 it was promoted to the top right of the UX instead of a link at the bottom. It may have been experimental or underutilized beforehand.

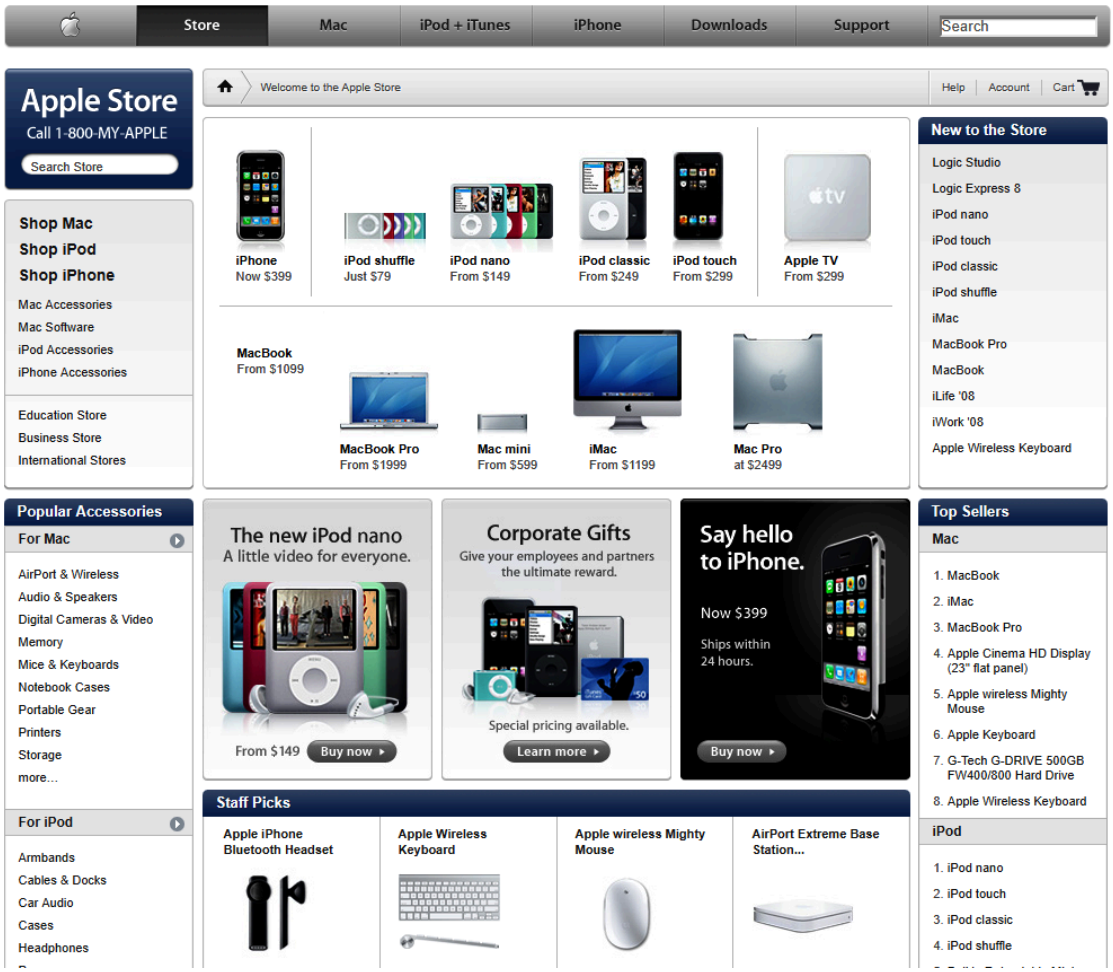

Figure 1-5. Apple.com, October 2007, showing a storefront with items that can be purchased online

This huge shift in architecture design direction for websites also changed the way hackers targeted web applications. By then, serious efforts had been taken to secure servers and networks—the two leading attack vectors for hackers of the past decade. With the rise of application-like websites, the user became a perfect target for hackers.

It was a perfect setup. Users would soon have access to mission-critical functionality over the web. Military communications, bank transfers, and more would all eventually be done through web applications (a website that operates like a desktop application). Unfortunately, very few security controls were in place at the time to protect users against attacks that targeted them. Furthermore, education regarding hacking or the mechanisms that the internet ran on was scarce. Few early internet users in the 2000s could even begin to grasp the underlying technology that worked for them.

In the early 2000s, the first largely publicized denial of service (DoS) attacks shut down Yahoo!, Amazon, eBay, and other popular sites. In 2002,

Microsoft's ActiveX plug-in for browsers ended up with a vulnerability that allowed remote file uploads and downloads to be invoked by a website with malicious intentions. By the mid-2000s, hackers were regularly utilizing "phishing" websites to steal credentials. No controls were in place at the time to protect users against these websites.

Cross-Site Scripting (XSS) vulnerabilities that allowed a hacker's code to run in a user's browser session inside of a legitimate website ran rampant throughout the web during this time, as browser vendors had not yet built defenses for such attacks. Many of the hacking attempts of the 2000s came as a result of the technology driving the web being designed for a single user (the website owner). These technologies would topple when used to build a system that allowed the sharing of data between many users.

Hackers in the Modern Era, Circa 2015+

The point in discussing hacking in previous eras was to build a foundation from which we can begin our journey in this book. From analyzing the development and cryptoanalysis of Enigma machines in the 1930s, we gained insight into the importance of security and the lengths that others will go to in order to break that security.

In the 1940s, we saw an early use case for security automation. This particular case was driven by the ongoing battle between attackers and defenders. In this case, the Enigma machine technology had improved so much it could no longer be reliably broken by manual cryptoanalysis techniques. Alan Turing turned to automation to beat the security improvements.

The 1950s and 1960s showed us that hackers and tinkerers have a lot in common. We also learned that technology designed without considering users with malicious intent will lead to that technology eventually being broken into. We must always consider the worst-case scenario when designing technology to be deployed at scale and across a wide user base.

In the 1980s, the personal computer started to become popular. Around this time, we began to see the hackers we recognize today emerge. These hackers took advantage of the powers that software enabled, camouflaging viruses inside of legitimate applications, and using networks to spread their viruses rapidly.

Finally, the introduction and rapid adoption of the WWW led to the development of Web 2.0, which changed the way we think about the internet. Instead of the internet being a medium for sharing documents, it became a medium for sharing applications. As a result, new types of exploits emerged that take advantage of the user rather than the network or server. This is a fundamental change that is still true today, as most hackers today have moved to targeting web applications via browsers instead of desktop software and operating systems.

Let's jump ahead to 2019, the year I started writing the first edition of this book. At the time of writing, there were thousands of websites on the web that were backed by million- and billion-dollar companies. In fact, many companies made all of their revenue from their websites (e.g., Google, Facebook, Yahoo!, Reddit, etc.)

YouTube allows users to interact with each other and with the application itself (see Figure 1-6). Comments, video uploads, and image uploads are all supported. All of these uploads have variable permissions that allow the uploader to determine who the content should be visible to. Much of the hosted data persists permanently and across sessions, and several features have changes reflected between users in near-real time (via notifications). Also, a significant number of critical features are offloaded to the client (browser) rather than residing on the server.

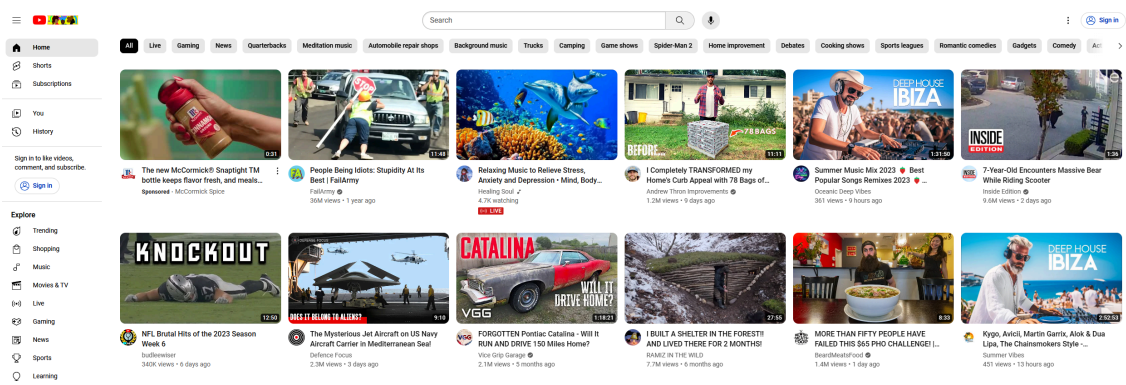

Figure 1-6. YouTube.com, now owned by Google, is a fantastic example of a Web 2.0 website

Some traditional desktop software companies are now trying to move their product lineup to the web, to what is known today as *the cloud*, which is simply a complex network of servers. Examples of this include Adobe with Creative Cloud, a subscription offering that provides Photoshop and other Adobe tools via the web, and Microsoft Office, which provides Word and Excel, but now as a web application.

Because of how much money is parked in web applications, the stakes are the highest they have ever been. This means applications today on the web are ripe for exploitation, and the rewards for exploiting them are sky high.

This is truly one of the best eras to be in for both hackers and engineers who emphasize security. Work for both is in high demand and on both sides of the law.

Browsers have become significantly more advanced than they were 10 years ago. Alongside this advancement has come a host of new security features. The networking protocols we use to access the internet have advanced as well.

Today's browsers offer very robust isolation between websites with different origins, following a security specification known as *Same Origin Policy* (SOP). This means that website A cannot be accessed by website B even if both are open at once or one is embedded as an iframe inside the other.

Browsers also accept a new security configuration known as *Content Security Policy* (CSP). CSP allows the developer of a website to specify var-

ious levels of security, such as whether scripts should be able to execute inline (in the HTML). This allows web developers to further protect their applications against common threats.

HTTP, the main protocol for sending web traffic, has also improved from a security perspective. HTTP has adopted protocols like SSL and TLS that enforce strict encryption for any data traveling over the network. This makes man-in-the-middle attacks very difficult to pull off successfully.

As a result of these advancements in browser security, many of the most successful hackers today are actually targeting the logic written by developers that runs in their web applications. Instead of targeting the browser itself, it is much easier to successfully breach a website by taking advantage of bugs in the application's code. Fortunately for hackers, web applications today are many times larger and more complex than web applications of the past.

Often today, a well-known web application can have hundreds of open source dependencies, integrations with other websites, and multiple databases of various types, and can be served from more than one web server in more than one location. These are the types of web applications you will find the most success in exploiting, and the types of web applications we will be focusing on throughout this book.

To summarize, today's web applications are much larger and more complex than their predecessors. As a hacker, you can now focus on breaking into web applications by exploiting logic bugs in the application code. Often these bugs result as a side effect of advanced user interaction featured within the web application.

The hackers of the past decade focused much of their time on breaking into servers, networks, and browsers. The modern hacker spends most of their time breaking into web applications by exploiting vulnerabilities present in code.

Summary

The origins of software security and the origins of hackers attempting to bypass that security go back at least around a hundred years. Today's software builds on top of lessons learned from the technology of the past, as does the security of that software.

Hackers of the past targeted applications differently than they do today. As one part of the application stack becomes increasingly more secure, hackers move on to target new emerging technologies. These new technologies often do not have the same level of security controls built in, and only through trial and error are engineers able to design and implement the proper security controls.

Similarly to how simple websites of the past were riddled with security holes (in particular, on the server and network levels), modern web applications bring attackers new surface area, which is being actively exploited. This brief historical context is important because it highlights that today's security concerns regarding web applications are just one stage in a cyclical process. Web applications of the future will be more secure, and hackers will likely move on to a new attack surface (maybe real-time communication or WebSockets, for example).

TIP

Each new technology comes with its own unique attack surface and vulnerabilities. One way to become an excellent hacker is to always stay up to date on the latest new technologies—these will often have security holes not yet published or found on the web.

In the meantime, this book will show you how to break into and secure modern web applications. But modern offensive and defensive security techniques are just one facet of learning you should derive from this book. Ultimately, being able to find your own solutions to security problems is the most valuable skill you can have as a security professional. If you can derive security-related critical thinking and problem-solving

skills from the coming chapters, then you will be able to stand above your peers when new or unusual exploits are found—or previously unseen security mechanisms stand in your way.