CHAPTER 1

Web Application (In)security

There is no doubt that web application security is a current and newsworthy subject. For all concerned, the stakes are high: for businesses that derive increasing revenue from Internet commerce, for users who trust web applications with sensitive information, and for criminals who can make big money by stealing payment details or compromising bank accounts. Reputation plays a critical role. Few people want to do business with an insecure website, so few organizations want to disclose details about their own security vulnerabilities or breaches. Hence, it is not a trivial task to obtain reliable information about the state of web application security today.

This chapter takes a brief look at how web applications have evolved and the many benefits they provide. We present some metrics about vulnerabilities in current web applications, drawn from the authors' direct experience, demonstrating that the majority of applications are far from secure. We describe the core security problem facing web applications — that users can supply arbitrary input — and the various factors that contribute to their weak security posture. Finally, we describe the latest trends in web application security and how these may be expected to develop in the near future.

The Evolution of Web Applications

In the early days of the Internet, the World Wide Web consisted only of web *sites*. These were essentially information repositories containing static documents. Web browsers were invented as a means of retrieving and displaying those documents, as shown in Figure 1-1. The flow of interesting information was one-way, from server to browser. Most sites did not authenticate users, because there was no need to. Each user was treated in the same way and was presented with the same information. Any security threats arising from hosting a website were related largely to vulnerabilities in web server software (of which there were many). If an attacker compromised a web server, he usually would not gain access to any sensitive information, because the information held on the server was already open to public view. Rather, an attacker typically would modify the files on the server to deface the web site's contents or use the server's storage and bandwidth to distribute "warez."

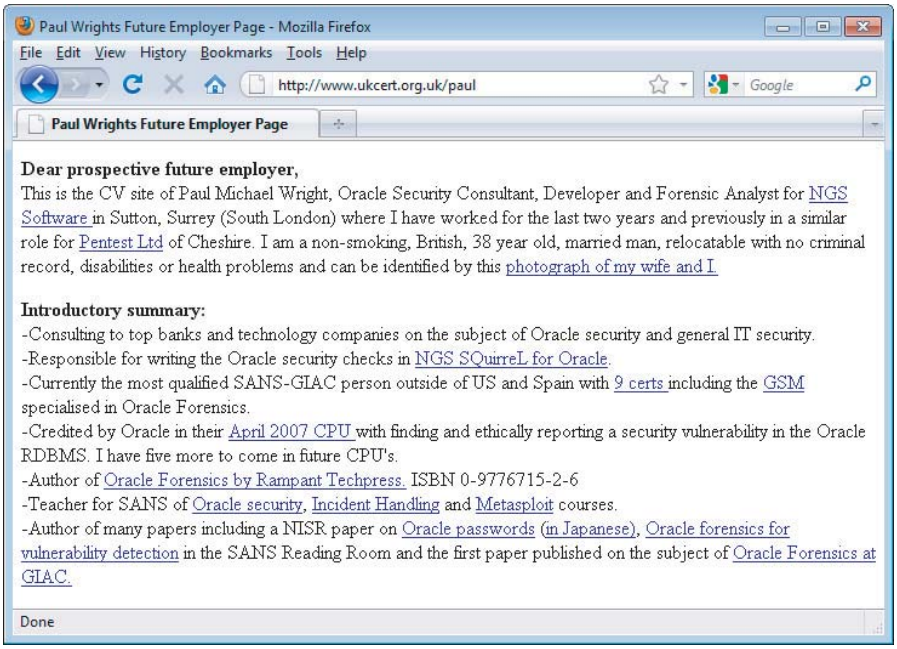

Figure 1-1: A traditional website containing static information

Today, the World Wide Web is almost unrecognizable from its earlier form. The majority of sites on the web are in fact applications (see Figure 1-2). They are highly functional and rely on two-way flow of information between the server and browser. They support registration and login, financial transactions,

search, and the authoring of content by users. The content presented to users is generated dynamically on the fly and is often tailored to each specific user. Much of the information processed is private and highly sensitive. Security, therefore, is a big issue. No one wants to use a web application if he believes his information will be disclosed to unauthorized parties.

Figure 1-2: A typical web application

Web applications bring with them new and significant security threats. Each application is different and may contain unique vulnerabilities. Most applications are developed in-house — many by developers who have only a partial understanding of the security problems that may arise in the code they are producing. To deliver their core functionality, web applications normally require connectivity to internal computer systems that contain highly sensitive data and that can perform powerful business functions. Fifteen years ago, if you wanted to make a funds transfer, you visited your bank, and the teller performed the transfer for you; today, you can visit a web application and perform the transfer yourself. An attacker who compromises a web application may be able to steal personal information, carry out financial fraud, and perform malicious actions against other users.

Common Web Application Functions

Web applications have been created to perform practically every useful function you could possibly implement online. Here are some web application functions that have risen to prominence in recent years:

- Shopping (Amazon)
- Social networking (Facebook)
- Banking (Citibank)
- Web search (Google)
- Auctions (eBay)
- Gambling (Betfair)
- Web logs (Blogger)
- Web mail (Gmail)
- Interactive information (Wikipedia)

Applications that are accessed using a computer browser increasingly overlap with mobile applications that are accessed using a smartphone or tablet. Most mobile applications employ either a browser or a customized client that uses HTTP-based APIs to communicate with the server. Application functions and data typically are shared between the various interfaces that the application exposes to different user platforms.

In addition to the public Internet, web applications have been widely adopted inside organizations to support key business functions. Many of these provide access to highly sensitive data and functionality:

- HR applications allowing users to access payroll information, give and receive performance feedback, and manage recruitment and disciplinary procedures.
- Administrative interfaces to key infrastructure such as web and mail servers, user workstations, and virtual machine administration.
- Collaboration software used for sharing documents, managing workflow and projects, and tracking issues. These types of functionality often involve critical security and governance issues, and organizations often rely completely on the controls built into their web applications.
- Business applications such as enterprise resource planning (ERP) software, which previously were accessed using a proprietary thick-client application, can now be accessed using a web browser.

- Software services such as e-mail, which originally required a separate e-mail client, can now be accessed via web interfaces such as Outlook Web Access.
- Traditional desktop office applications such as word processors and spreadsheets have been migrated to web applications through services such as Google Apps and Microsoft Office Live.

In all these examples, what are perceived as "internal" applications are increasingly being hosted externally as organizations move to outside service providers to cut costs. In these so-called *cloud* solutions, business-critical functionality and data are opened to a wider range of potential attackers, and organizations are increasingly reliant on the integrity of security defenses that are outside of their control.

The time is fast approaching when the only client software that most computer users will need is a web browser. A diverse range of functions will have been implemented using a shared set of protocols and technologies, and in so doing will have inherited a distinctive range of common security vulnerabilities.

Benefits of Web Applications

It is not difficult to see why web applications have enjoyed such a dramatic rise to prominence. Several technical factors have worked alongside the obvious commercial incentives to drive the revolution that has occurred in how we use the Internet:

- HTTP, the core communications protocol used to access the World Wide Web, is lightweight and connectionless. This provides resilience in the event of communication errors and avoids the need for the server to hold open a network connection to every user, as was the case in many legacy client/server applications. HTTP can also be proxied and tunneled over other protocols, allowing for secure communication in any network configuration.
- Every web user already has a browser installed on his computer and mobile device. Web applications deploy their user interface dynamically to the browser, avoiding the need to distribute and manage separate client software, as was the case with pre-web applications. Changes to the interface need to be implemented only once, on the server, and take effect immediately.
- Today's browsers are highly functional, enabling rich and satisfying user interfaces to be built. Web interfaces use standard navigational and

input controls that are immediately familiar to users, avoiding the need to learn how each individual application functions. Client-side scripting enables applications to push part of their processing to the client side, and browsers' capabilities can be extended in arbitrary ways using browser extension technologies where necessary.

- The core technologies and languages used to develop web applications are relatively simple. A wide range of platforms and development tools are available to facilitate the development of powerful applications by relative beginners, and a large quantity of open source code and other resources is available for incorporation into custom-built applications.

Web Application Security

As with any new class of technology, web applications have brought with them a new range of security vulnerabilities. The set of most commonly encountered defects has evolved somewhat over time. New attacks have been conceived that were not considered when existing applications were developed. Some problems have become less prevalent as awareness of them has increased. New technologies have been developed that have introduced new possibilities for exploitation. Some categories of flaws have largely gone away as the result of changes made to web browser software.

The most serious attacks against web applications are those that expose sensitive data or gain unrestricted access to the back-end systems on which the application is running. High-profile compromises of this kind continue to occur frequently. For many organizations, however, any attack that causes system downtime is a critical event. Application-level denial-of-service attacks can be used to achieve the same results as traditional resource exhaustion attacks against infrastructure. However, they are often used with more subtle techniques and objectives. They may be used to disrupt a particular user or service to gain a competitive edge against peers in the realms of financial trading, gaming, online bidding, and ticket reservations.

Throughout this evolution, compromises of prominent web applications have remained in the news. There is no sense that a corner has been turned and that these security problems are on the wane. By some measure, web application security is today the most significant battleground between attackers and those with computer resources and data to defend, and it is likely to remain so for the foreseeable future.

"This Site Is Secure"

There is a widespread awareness that security is an issue for web applications. Consult the FAQ page of a typical application, and you will be reassured that it is in fact secure.

Most applications state that they are secure because they use SSL. For example:

> *This site is absolutely secure. It has been designed to use 128-bit Secure Socket Layer (SSL) technology to prevent unauthorized users from viewing any of your information. You may use this site with peace of mind that your data is safe with us.*

Users are often urged to verify the site's certificate, admire the advanced cryptographic protocols in use, and, on this basis, trust it with their personal information.

Increasingly, organizations also cite their compliance with Payment Card Industry (PCI) standards to reassure users that they are secure. For example:

> *We take security very seriously. Our web site is scanned daily to ensure that we remain PCI compliant and safe from hackers. You can see the date of the latest scan on the logo below, and you are guaranteed that our web site is safe to use.*

In fact, the majority of web applications are insecure, despite the widespread usage of SSL technology and the adoption of regular PCI scanning. The authors of this book have tested hundreds of web applications in recent years. Figure 1-3 shows what percentage of applications tested during 2007 and 2011 were found to be affected by some common categories of vulnerability:

- **Broken authentication (62%)** — This category of vulnerability encompasses various defects within the application's login mechanism, which may enable an attacker to guess weak passwords, launch a brute-force attack, or bypass the login.
- **Broken access controls (71%)** — This involves cases where the application fails to properly protect access to its data and functionality, potentially enabling an attacker to view other users' sensitive data held on the server or carry out privileged actions.
- **SQL injection (32%)** — This vulnerability enables an attacker to submit crafted input to interfere with the application's interaction with back-end databases. An attacker may be able to retrieve arbitrary data from the application, interfere with its logic, or execute commands on the database server itself.

- **Cross-site scripting (94%)** — This vulnerability enables an attacker to target other users of the application, potentially gaining access to their data, performing unauthorized actions on their behalf, or carrying out other attacks against them.
- **Information leakage (78%)** — This involves cases where an application divulges sensitive information that is of use to an attacker in developing an assault against the application, through defective error handling or other behavior.
- **Cross-site request forgery (92%)** — This flaw means that application users can be induced to perform unintended actions on the application within their user context and privilege level. The vulnerability allows a malicious web site visited by the victim user to interact with the application to perform actions that the user did not intend.

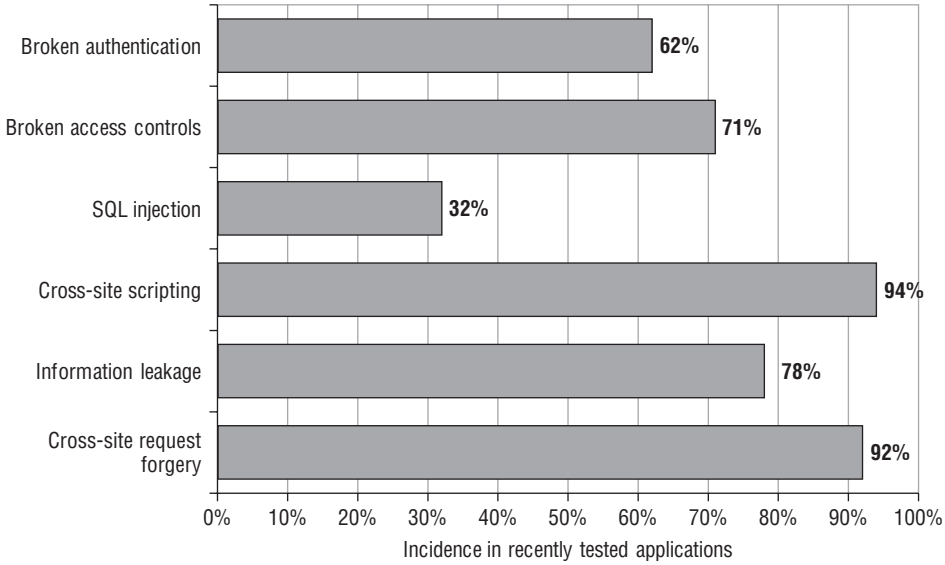

Figure 1-3: The incidence of some common web application vulnerabilities in applications recently tested by the authors (based on a sample of more than 100)

SSL is an excellent technology that protects the confidentiality and integrity of data in transit between the user's browser and the web server. It helps defend against eavesdroppers, and it can provide assurance to the user of the identity of the web server he is dealing with. But it does not stop attacks that directly target the server or client components of an application, as most successful attacks do. Specifically, it does not prevent any of the vulnerabilities just listed, or many others that can render an application critically exposed to attack. Regardless of whether they use SSL, most web applications still contain security flaws.

The Core Security Problem: Users Can Submit Arbitrary Input

As with most distributed applications, web applications face a fundamental problem they must address to be secure. Because the client is outside of the application's control, users can submit arbitrary input to the server-side application. The application must assume that all input is potentially malicious. Therefore, it must take steps to ensure that attackers cannot use crafted input to compromise the application by interfering with its logic and behavior, thus gaining unauthorized access to its data and functionality.

This core problem manifests itself in various ways:

- Users can interfere with any piece of data transmitted between the client and the server, including request parameters, cookies, and HTTP headers. Any security controls implemented on the client side, such as input validation checks, can be easily circumvented.
- Users can send requests in any sequence and can submit parameters at a different stage than the application expects, more than once, or not at all. Any assumption developers make about how users will interact with the application may be violated.
- Users are not restricted to using only a web browser to access the application. Numerous widely available tools operate alongside, or independently of, a browser to help attack web applications. These tools can make requests that no browser would ordinarily make and can generate huge numbers of requests quickly to find and exploit problems.

The majority of attacks against web applications involve sending input to the server that is crafted to cause some event that was not expected or desired by the application's designer. Here are some examples of submitting crafted input to achieve this objective:

- Changing the price of a product transmitted in a hidden HTML form field to fraudulently purchase the product for a cheaper amount
- Modifying a session token transmitted in an HTTP cookie to hijack the session of another authenticated user
- Removing certain parameters that normally are submitted to exploit a logic flaw in the application's processing
- Altering some input that will be processed by a back-end database to inject a malicious database query and access sensitive data

Needless to say, SSL does nothing to stop an attacker from submitting crafted input to the server. If the application uses SSL, this simply means that other users on the network cannot view or modify the attacker's data in transit. Because

the attacker controls her end of the SSL tunnel, she can send anything she likes to the server through this tunnel. If any of the previously mentioned attacks are successful, the application is emphatically vulnerable, regardless of what its FAQ may tell you.

Key Problem Factors

The core security problem faced by web applications arises in any situation where an application must accept and process untrusted data that may be malicious. However, in the case of web applications, several factors have combined to exacerbate the problem and explain why so many web applications on the Internet today do such a poor job of addressing it.

Underdeveloped Security Awareness

Although awareness of web application security issues has grown in recent years, it remains less well-developed than in longer-established areas such as networks and operating systems. Although most people working in IT security have a reasonable grasp of the essentials of securing networks and hardening hosts, widespread confusion and misconception still exist about many of the core concepts involved in web application security. A web application developer's work increasingly involves weaving together tens, or even hundreds, of third-party packages, all designed to abstract the developer away from the underlying technologies. It is common to meet experienced web application developers who make major assumptions about the security provided by their programming framework and to whom an explanation of many basic types of flaws comes as a revelation.

Custom Development

Most web applications are developed in-house by an organization's own staff or third-party contractors. Even where an application employs well-established components, these are typically customized or bolted together using new code. In this situation, every application is different and may contain its own unique defects. This stands in contrast to a typical infrastructure deployment, in which an organization can purchase a best-of-breed product and install it in line with industry-standard guidelines.

Deceptive Simplicity

With today's web application platforms and development tools, it is possible for a novice programmer to create a powerful application from scratch in a short period of time. But there is a huge difference between producing code that is

functional and code that is secure. Many web applications are created by well-meaning individuals who simply lack the knowledge and experience to identify where security problems may arise.

A prominent trend in recent years has been the use of application frameworks that provide ready-made code components to handle numerous common areas of functionality, such as authentication, page templates, message boards, and integration with common back-end infrastructure components. Examples of these frameworks include Liferay and Appfuse. These products make it quick and easy to create working applications without requiring a technical understanding of how the applications work or the potential risks they may contain. This also means many companies use the same frameworks. Thus, when a vulnerability is discovered, it affects many unrelated applications.

Rapidly Evolving Threat Profile

Research into web application attacks and defenses continues to be a thriving area in which new concepts and threats are conceived at a faster rate than is now the case for older technologies. Particularly on the client side, it is common for the accepted defenses against a particular attack to be undermined by research that demonstrates a new attack technique. A development team that begins a project with a complete knowledge of current threats may have lost this status by the time the application is completed and deployed.

Resource and Time Constraints

Most web application development projects are subject to strict constraints on time and resources, arising from the economics of in-house, one-off development. In most organizations, it is often infeasible to employ dedicated security expertise in the design or development teams. And due to project slippage, security testing by specialists is often left until very late in the project's life cycle. In the balancing of competing priorities, the need to produce a stable and functional application by a deadline normally overrides less tangible security considerations. A typical small organization may be willing to pay for only a few man-days of consulting time to evaluate a new application. A quick penetration test will often find the low-hanging fruit, but it may miss more subtle vulnerabilities that require time and patience to identify.

Overextended Technologies

Many of the core technologies employed in web applications began life when the landscape of the World Wide Web was very different. They have since been pushed far beyond the purposes for which they were originally conceived, such as the use of JavaScript as a means of data transmission in many AJAX-based

applications. As the expectations placed on web application functionality have rapidly evolved, the technologies used to implement this functionality have lagged behind the curve, with old technologies stretched and adapted to meet new requirements. Unsurprisingly, this has led to security vulnerabilities as unforeseen side effects emerge.

Increasing Demands on Functionality

Applications are designed primarily with functionality and usability in mind. Once-static user profiles now contain social networking features, allowing uploading of pictures and wiki-style editing of pages. A few years ago an application designer may have been content with implementing a username and password challenge to create the login functionality. Modern sites may include password recovery, username recovery, password hints, and an option to remember the username and password on future visits. Such a site would undoubtedly be promoted as having numerous security features, yet each one is really a self-service feature adding to the site's attack surface.

The New Security Perimeter

Before the rise of web applications, organizations' efforts to secure themselves against external attack were largely focused on the network perimeter. Defending this perimeter entailed hardening and patching the services it needed to expose and firewalling access to others.

Web applications have changed all this. For an application to be accessible by its users, the perimeter firewall must allow inbound connections to the server over HTTP or HTTPS. And for the application to function, the server must be allowed to connect to supporting back-end systems, such as databases, mainframes, and financial and logistical systems. These systems often lie at the core of the organization's operations and reside behind several layers of network-level defenses.

If a vulnerability exists within a web application, an attacker on the public Internet may be able to compromise the organization's core back-end systems solely by submitting crafted data from his web browser. This data sails past all the organization's network defenses, in the same way as does ordinary, benign traffic to the web application.

The effect of widespread deployment of web applications is that the security perimeter of a typical organization has moved. Part of that perimeter is still embodied in firewalls and bastion hosts. But a significant part of it is now occupied by the organization's web applications. Because of the manifold ways in which web applications receive user input and pass this to sensitive back-end systems, they are the potential gateways for a wide range of attacks, and defenses against these attacks must be implemented within the applications themselves. A single

line of defective code in a single web application can render an organization's internal systems vulnerable. Furthermore, with the rise of mash-up applications, third-party widgets, and other techniques for cross-domain integration, the server-side security perimeter frequently extends well beyond the organization itself. Implicit trust is placed in the services of external applications and services. The statistics described previously, of the incidence of vulnerabilities within this new security perimeter, should give every organization pause for thought.

NOTE **For an attacker targeting an organization, gaining access to the network or executing arbitrary commands on servers may not be what he wants to achieve. Often, and perhaps typically, what an attacker really wants is to perform some application-level action such as stealing personal information, transferring funds, or making cheap purchases. And the relocation of the security perimeter to the application layer may greatly assist an attacker in achieving these objectives.**

For example, suppose that an attacker wants to "hack in" to a bank's systems and steal money from users' accounts. In the past, before the bank deployed a web application, the attacker might have needed to find a vulnerability in a publicly reachable service, exploit this to gain a toehold on the bank's DMZ, penetrate the firewall restricting access to its internal systems, map the network to find the mainframe computer, decipher the arcane protocol used to access it, and guess some credentials to log in. However, if the bank now deploys a vulnerable web application, the attacker may be able to achieve the same outcome simply by modifying an account number in a hidden field of an HTML form.

A second way in which web applications have moved the security perimeter arises from the threats that users themselves face when they access a vulnerable application. A malicious attacker can leverage a benign but vulnerable web application to attack any user who visits it. If that user is located on an internal corporate network, the attacker may harness the user's browser to launch an attack against the local network from the user's trusted position. Without any cooperation from the user, the attacker may be able to carry out any action that the user could perform if she were herself malicious. With the proliferation of browser extension technologies and plug-ins, the extent of the client-side attack surface has increased considerably.

Network administrators are familiar with the idea of preventing their users from visiting malicious web sites, and end users themselves are gradually becoming more aware of this threat. But the nature of web application vulnerabilities means that a vulnerable application may present no less of a threat to its users and their organization than a web site that is overtly malicious. Correspondingly, the new security perimeter imposes a duty of care on all application owners to protect their users from attacks against them delivered via the application.

A further way in which the security perimeter has partly moved to the client side is through the widespread use of e-mail as an extended authentication mechanism. A huge number of today's applications contain "forgotten password" functions that allow an attacker to generate an account recovery e-mail to any registered address, without requiring any other user-specific information. This allows an attacker who compromises a user's web mail account to easily escalate the attack and compromise the victim's accounts on most of the web applications for which the victim is registered.

The Future of Web Application Security

Over a decade after their widespread adoption, web applications on the Internet today are still rife with vulnerabilities. Understanding of the security threats facing web applications, and effective ways of addressing these, are still underdeveloped within the industry. There is currently little indication that the problem factors described in this chapter will disappear in the near future.

That said, the details of the web application security landscape are not static. Even though old and well-understood vulnerabilities such as SQL injection continue to appear, their prevalence is gradually diminishing. Furthermore, the instances that remain are becoming more difficult to find and exploit. New research in these areas is generally focused on developing advanced techniques for attacking more subtle manifestations of vulnerabilities that a few years ago could be easily detected and exploited using only a browser.

A second prominent trend has been a gradual shift in attention from attacks against the server side of the application to those that target application users. The latter kind of attack still leverages defects within the application itself, but it generally involves some kind of interaction with another user to compromise that user's dealings with the vulnerable application. This is a trend that has been replicated in other areas of software security. As awareness of security threats matures, flaws in the server side are the first to be well understood and addressed, leaving the client side as a key battleground as the learning process continues. Of all the attacks described in this book, those against other users are evolving the most quickly, and they have been the focus of most research in recent years.

Various recent trends in technology have somewhat altered the landscape of web applications. Popular consciousness about these trends exists by means of various rather misleading buzzwords, the most prominent of which are these:

- Web 2.0 — This term refers to the greater use of functionality that enables user-generated content and information sharing, and also the adoption of various technologies that broadly support this functionality, including asynchronous HTTP requests and cross-domain integration.

- Cloud computing — This term refers to greater use of external service providers for various parts of the technology stack, including application software, application platforms, web server software, databases, and hardware. It also refers to increased usage of virtualization technologies within hosting environments.

As with most changes in technology, these trends have brought with them some new attacks and variations on existing attacks. Notwithstanding the hype, the issues raised are not quite as revolutionary as they may initially appear. We will examine the security implications of these and other recent trends in the appropriate locations throughout this book.

Despite all the changes that have occurred within web applications, some categories of "classic" vulnerabilities show no sign of diminishing. They continue to arise in pretty much the same form as they did in the earliest days of the web. These include defects in business logic, failures to properly apply access controls, and other design issues. Even in a world of bolted-together application components and everything-as-a-service, these timeless issues are likely to remain widespread.

Summary

In a little over a decade, the World Wide Web has evolved from purely static information repositories into highly functional applications that process sensitive data and perform powerful actions with real-world consequences. During this development, several factors have combined to bring about the weak security posture demonstrated by the majority of today's web applications.

Most applications face the core security problem that users can submit arbitrary input. Every aspect of the user's interaction with the application may be malicious and should be regarded as such unless proven otherwise. Failure to properly address this problem can leave applications vulnerable to attack in numerous ways.

All the evidence about the current state of web application security indicates that although some aspects of security have indeed improved, entirely new threats have evolved to replace them. The overall problem has not been resolved on any significant scale. Attacks against web applications still present a serious threat to both the organizations that deploy them and the users who access them.