Technical white paper

2011 top cyber security risks report

Table of contents

Contributors

Producing the *2011 top cyber security risks report* is a collaborative effort among HP DVLabs and other HP teams, such as HP Fortify on Demand and the HP Web Security Research Group. We would like to sincerely thank the Open Source Vulnerability Database (OSVDB) for allowing print rights to its data in this report. For information on how you can help OSVDB:

https://osvdb.org/account/signup

http://osvdb.org/support

Contributor	Title
Linda Davis	Technical Writer, HP DVLabs
Will Gragido	Senior Manager, Advanced Cyber Threat Intelligence Solutions
Brian Hein	Technology Architect, HP Enterprise Security Products
Adam Hils	Senior Product Manager, HP Fortify
Dan Holden	Director, HP DVLabs
Prajakta Jagdale	Principal Security Researcher, HP Web Security Research Group
Jason Jones	Advanced Security Intelligence Engineer, HP DVLabs
Jennifer Lake	Product Marketing, HP DVLabs
Jason Lancaster	Technical Director, HP Enterprise Security Products
Mark Painter	Product Marketing Manager, HP Fortify
Young Park	Lead FOD Operator, HP Fortify on Demand
John Pirc	Director, Product Management, HP Enterprise Security Products
Erica Quinlan	TAM Manager, HP Fortify on Demand
Joe Sechman	Manager, HP Web Security Research Group
Nidhi Shah	Principal Security Researcher, HP Web Security Research Group
Craig Smith	TaaS Operations Manager, HP Fortify on Demand
Michael Thuma	Technical Account Manager
Jewel Timpe	Business Operations, HP DVLabs
Chaoting Xuan	Security Research Engineer, HP Web Security Research Group

Overview

Enterprise organizations have been under security attacks for the past decade, but the security events in 2011 have created a ripple effect that will be felt for years to come and will actually start to shift the way enterprise organizations view security. For example, 2011 saw a significant increase in activity from "hacktivist" groups Anonymous and Lulz Security (LulzSec). The motivation for these groups' organized, systematic attacks on businesses or individuals—retaliation for perceived wrongdoing—brings new visibility to a security threat that has been looming for years and highlights a new era of security risk that must be addressed. In addition, highly publicized attacks on major corporations such as Sony, RSA, and the United States Postal Service demonstrated the significant financial loss that can result from a vulnerable system.

Because unplugging the business from the Internet is not a viable security option, the question becomes: What is the best way to minimize risk to the most critical assets of the organization without interrupting or impeding business operations? Prioritization of assets and risk is essential, but so is prioritizing how and where to deploy security protection.

In the *2011 top cyber security risks report*, HP Enterprise Security provides a broad view of the vulnerability threat landscape, as well as in-depth research and analysis on security attacks and trends. The aim of this report is to highlight the biggest risks that enterprise organizations face today—and to help prioritize mitigation strategies. Key findings from this report include the following:

- **Continued decline of new, disclosed vulnerabilities in commercial applications**
  The report notes the decline in commercial vulnerability reporting, and it discusses the key trends in the vulnerability disclosure market that may be hiding a deeper issue. The report also highlights the growing market for private sharing of vulnerabilities, the increased expertise required to uncover complex vulnerabilities, and the price these can fetch in various markets. Data from HP Fortify will also highlight the increasing number of vulnerabilities that are being discovered in custom applications—vulnerabilities that can be devastating to the security posture of an organization.
- **Changes in attack motivation are increasing security risk**
  While security attackers have always sought glory and/or financial gain from their activities, the formation of hacktivist groups, like Anonymous, has added not only a purpose behind security attacks, but a level of organization as well. This shift in motivation and subsequent organization has given rise to newer and more severe security attacks. This report will highlight the motivations of today's security attack community—and the implications for security defense techniques.
- **Increase in the number of attacks against a "smaller" set of known vulnerabilities**
  Despite the shrinking number of known vulnerabilities in commercial applications, the report will use real data—pulled from the HP TippingPoint Intrusion Prevention System (IPS) and HP Fortify—to highlight an increase in severe attacks against both client/server and Web applications. The data is broken down by attacks, vulnerability category, source information, and severity to provide a snapshot of the attack landscape. This section also features an actual case study of the Web application risks at one large corporation.
- **Improved techniques for executing security attacks**
  While many targeted attacks leverage zero-day vulnerabilities, the average cyber criminal generally exploits existing vulnerabilities. Data from the report breaks down several techniques, including obfuscation, used to successfully exploit existing vulnerabilities. The report also includes an in-depth look at the Blackhole exploit toolkit, which uses many of the techniques highlighted.

Information in this report originates from various sources, allowing HP Enterprise Security to obtain a broad set of data from which to correlate meaningful findings. These sources include:

- Vulnerability information from the Open Source Vulnerability Database (OSVDB) and the HP DVLabs Zero Day Initiative (ZDI)
- Web application data from the HP Fortify Web Security Research Group and Fortify on Demand
- Attack information from a worldwide network of HP TippingPoint Intrusion Prevention Systems
- Exploit analysis from HP DVLabs

Vulnerability trends

Understanding security risk begins with knowing where weaknesses exist in an enterprise infrastructure. Vulnerabilities exist at every level of the enterprise infrastructure—at the perimeter, in the core, and even in some of the less traditional areas such as mobile and virtual environments. These vulnerabilities are the gateway that malicious cyber actors use to introduce attacks that can steal or alter data, shut down systems, or access confidential information.

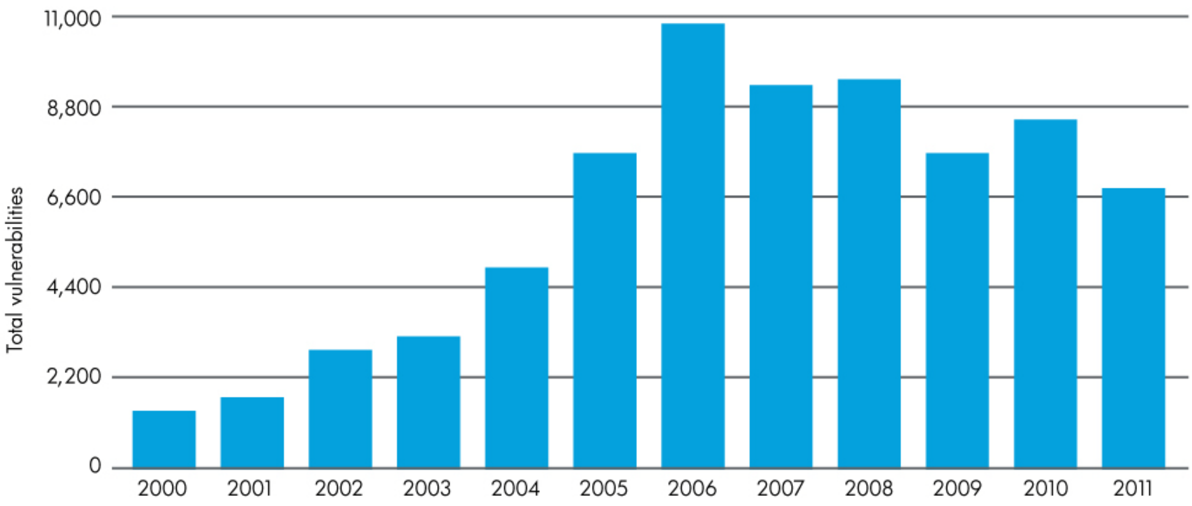
This section of the report uses data from the Open Source Vulnerability Database (OSVDB), HP DVLabs Zero Day Initiative (ZDI), and the HP Web Security Research Group to demonstrate the following vulnerability trends:

- **Decline in commercial vulnerabilities creates a false sense of security.**
  This section presents the numbers of new vulnerabilities being reported in commercial applications. It uses OSVDB data to provide a snapshot of what is happening in the world of vulnerabilities—notably a downward trend in numbers of new vulnerabilities being reported.
- **Changing vulnerability disclosure trends affect discovery and reporting through traditional channels.**
  This section delves further into the market for vulnerability disclosure to highlight why the numbers of new disclosed vulnerabilities are declining.
- **Web application vulnerabilities are still a significant risk to the enterprise.**
  This section provides another piece of the vulnerability landscape, highlighting vulnerabilities present in custom-developed Web applications. This section uses data from OSVDB and HP Fortify to demonstrate the type and frequency of vulnerabilities that exist in Web applications.
- **Vulnerabilities are rising in nontraditional enterprise infrastructures.**
  This final section provides a quick overview of the other areas of the vulnerability landscape. It touches on the vulnerabilities—and risks—present in virtual infrastructure as well as SCADA systems. It also includes data from the Web Security Research Group to provide a snapshot of the mobile vulnerabilities that are beginning to have an impact in the enterprise.

Decline in commercial vulnerabilities creates a false sense of security

Based on data from the OSVDB, the total number of new vulnerabilities reported during 2011 is about 19.5 percent lower than the number of new vulnerabilities reported in 2010. During 2011, OSVDB cataloged 6,843 reported vulnerabilities in Internet-based systems, applications, and other computing tools, compared to 8,502 cataloged in 2010 (see **Figure 1**).

Figure 1: Disclosed vulnerabilities measured by OSVDB, 2000–2011



In the year-by-year view of OSVDB data, it is clear that vulnerability reporting peaked in 2006 and has been in a slow decline ever since. This decline in disclosure of new vulnerabilities doesn’t mean that vulnerabilities don’t exist or that software is suddenly secure.

Rather it is indicative of a change in the vulnerability landscape—including how these vulnerabilities are being discovered, how they are disclosed, and where they exist. The following sections will use OSVDB, ZDI, and HP Web Security Research Group data to illustrate potential reasons behind this shift.

Changing vulnerability disclosure trends affect discovery and reporting through traditional channels

To understand why vulnerability reporting is down, it is important to look at the trends impacting discovery and disclosure of vulnerability information.

Vulnerability information can be sold through a number of different outlets, including public programs such as HP DVLabs ZDI, consulting organizations, or the black market.

The OSVDB data provides an excellent snapshot of the vulnerability landscape, particularly for common computing applications. However, due to the nature of vulnerability reporting, OSVDB can only count vulnerabilities being disclosed publicly or submitted directly to the database by individuals.

As consulting organizations—such as VUPEN—become more popular, significant numbers of vulnerabilities are being discovered but only disclosed to private clients. This practice leaves a significant quantity of vulnerabilities uncounted.

Improved software ups the ante for vulnerability reporting

Further complicating the vulnerability question is the complexity of software today. For the past several years, software vendors have been improving their products and adding features into their code to thwart the unauthorized discovery and exploitation of their products. These steps have not only helped close a number of holes in commercial products, but also made vulnerabilities more difficult to discover.

The increasing difficulty of discovering vulnerabilities, combined with the market value of bugs with a high CVSS rating (8 to 10), increases the market value for vulnerabilities across the board. This increase indicates two things:

- Security researchers need to have extensive expertise in a specific application set in order to find vulnerabilities.
- It is more worthwhile to spend extra time uncovering the severe vulnerabilities that fetch a higher market price.

The following graphs depict the way in which reporting severe vulnerabilities has changed over the years.

Figure 2: Vulnerability severity landscape using OSVDB data, 2000–2011

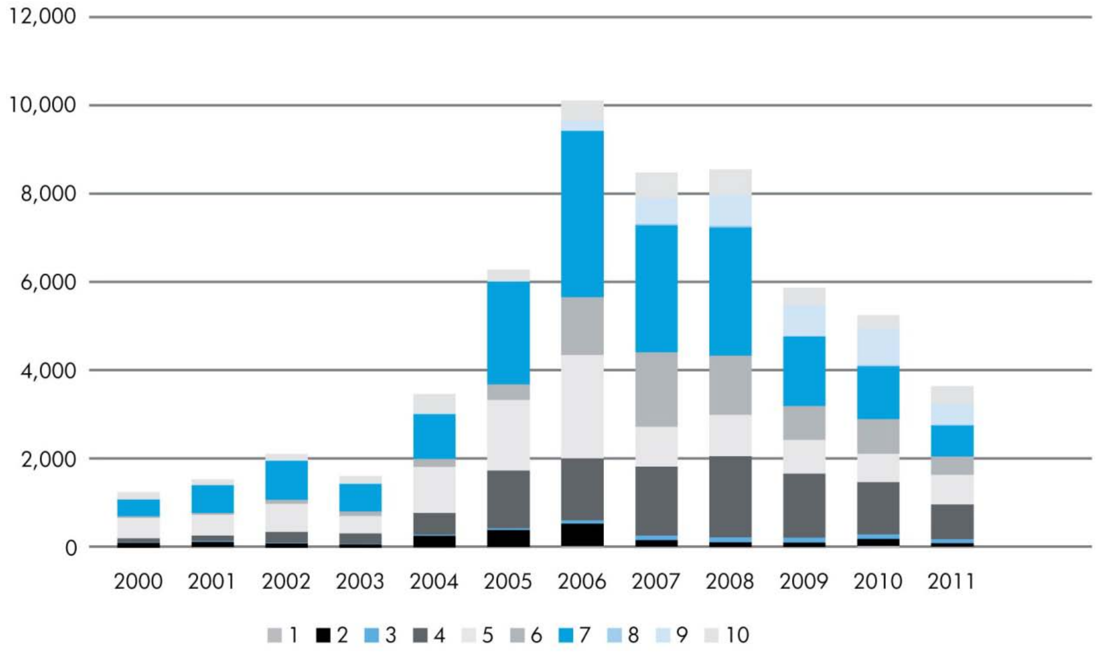
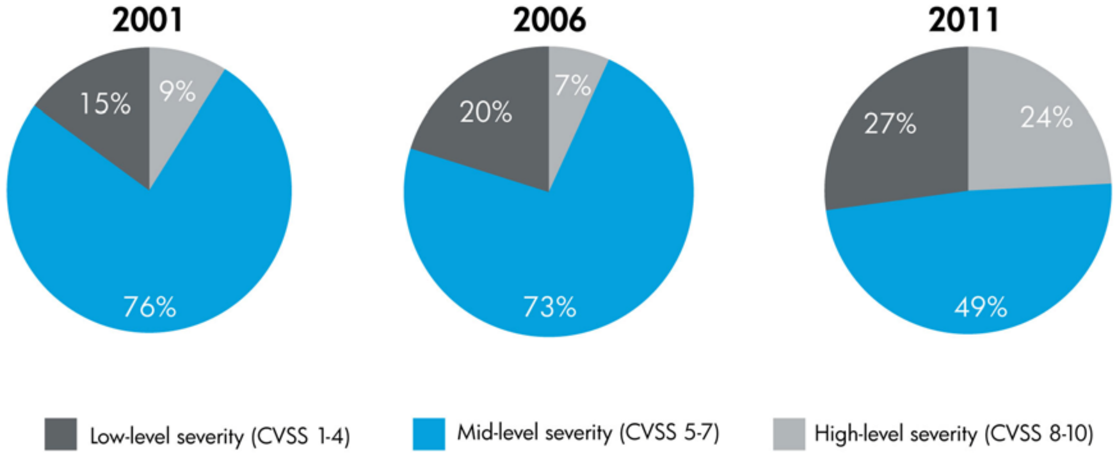


Figure 2 uses data from OSVDB to break down the vulnerabilities by year based on CVSS score. It should be noted that OSVDB does not require a CVSS score to report vulnerabilities, so the severity ratings only represent a subset of all reported vulnerabilities, but there is still some interesting information to take away from this.

High-severity vulnerabilities (CVSS 8 to 10) make up 24 percent of the total vulnerabilities in 2011. Vulnerabilities with a CVSS score higher than 7 can, in theory, result in a remote code execution. In other words, an attacker can exploit that vulnerability and gain total control of the system.

Uncovering a remote code execution vulnerability requires specialized knowledge of the application and system, so it is no surprise that these comprise a small percentage of the vulnerabilities disclosed. However, what is interesting is that the number of severe vulnerabilities is growing at a faster rate than the other levels of vulnerabilities. **Figure 3** highlights how the percentage of high-severity vulnerabilities has increased in the past 10 years.

Figure 3: Severity of OSVDB vulnerabilities broken out over 10 years



In the peak year for vulnerability disclosure (2006), mid-level severity vulnerabilities (CVSS 5 to 7) made up a bulk of the disclosures. This was about the time that fuzzing tools became mainstream and researchers began finding some of the more common, easier-to-find vulnerabilities—and many of these fell into the mid-level in terms of exploitability. This number has since dropped sharply, in part because software vendors have made great strides in improving the security of their product code and eliminating old vulnerabilities.

The expertise and time required to uncover and prove the exploitability of very severe vulnerabilities demands a commensurately high payment. Because it is worth reseachers' effort to spend more time uncovering these very severe vulnerabilities, fewer bugs are being discovered overall. In addition, because very severe vulnerabilities often capture a higher price on the black or gray market, it is plausible that many of these may not be reported through the public channels measured by OSVDB.

ZDI information

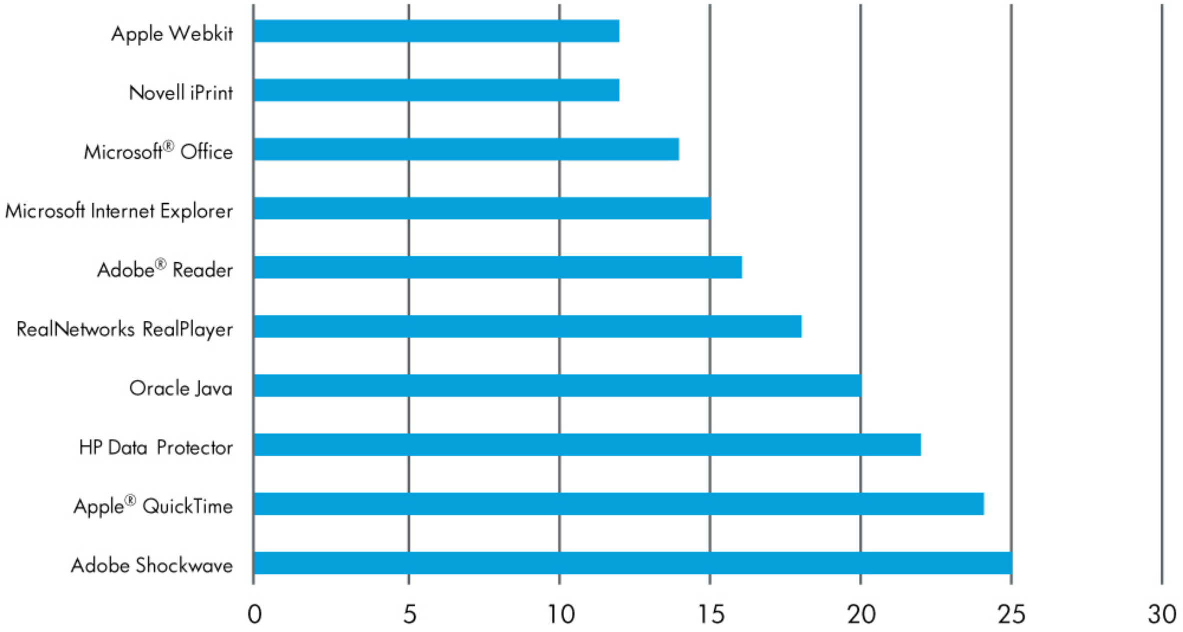
The Zero Day Initiative (ZDI) is a program managed by HP DVLabs that purchases vulnerability information from security researchers and provides it free of charge to the affected vendor. For the most part, ZDI purchases only very severe vulnerability information (CVSS greater than 7), so it provides a broader look at these vulnerabilities and how they affect the enterprise.

ZDI was founded by HP TippingPoint in 2005 to reward security researchers for responsibly disclosing vulnerabilities. The program is designed such that researchers provide HP TippingPoint with exclusive information about previously un-patched vulnerabilities they have discovered. HP DVLabs validates each vulnerability and works with the affected vendor until the vulnerability is patched. At the same time, HP DVLabs develops a security solution that provides pre-emptive protection for HP's customers even before the application vendor distributes a fix for the vulnerability.

Access to ZDI data not only adds to the overall view of how many vulnerabilities are reported, but also begins to paint a picture of where these vulnerabilities exist.

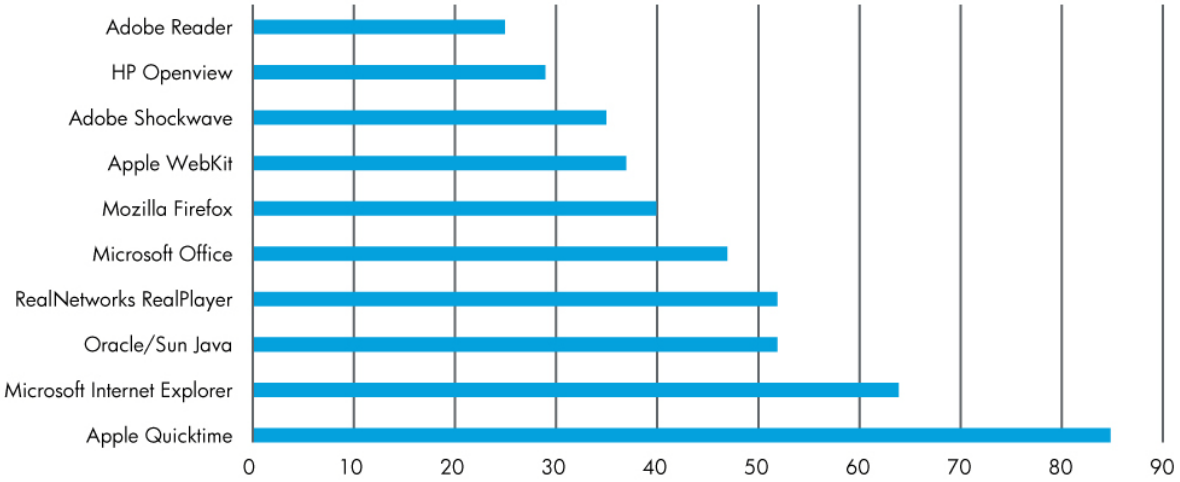
Figure 4 highlights the top 10 vulnerabilities reported through ZDI in 2011. Browsers and browser plug-ins like ShockWave and Java continue to lead in terms of the number of vulnerabilities disclosed. This supports the theory that while software may be getting more secure, the applications running on top of them are introducing vulnerabilities.

Figure 4: Top 10 vulnerabilities disclosed through ZDI in 2011



The same is also true historically. In **Figure 5**, which shows ZDI disclosed vulnerabilities from 2005 through 2011, the number of QuickTime vulnerabilities jumps significantly, as do the vulnerabilities in Web browsers such as Internet Explorer and Firefox.

Figure 5: Top 10 vulnerabilities disclosed through ZDI 2005–2011



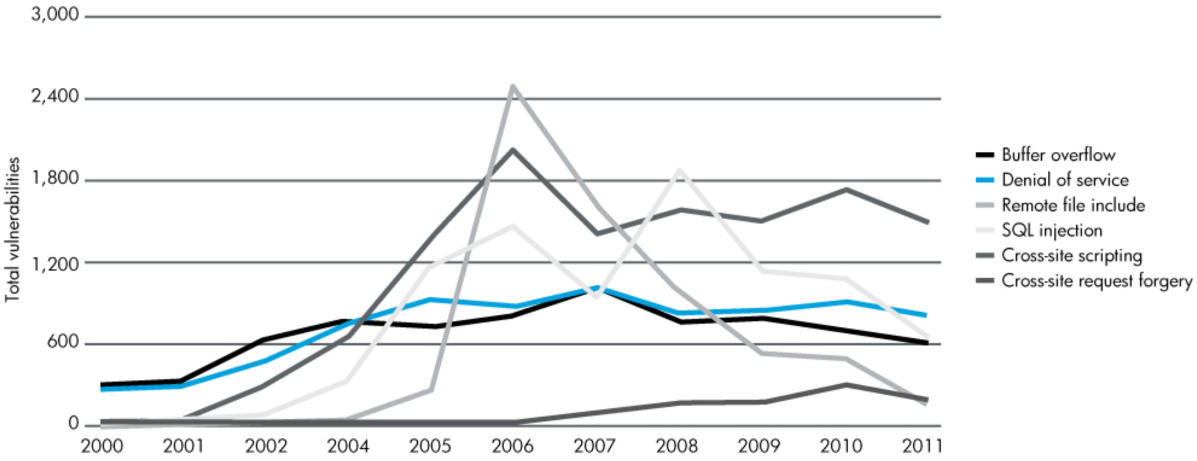
The next section takes a closer look at vulnerabilities in commercial applications broken down by category.

Web applications are still a significant risk to the enterprise

Vulnerabilities can exist anywhere in the application—in the underlying software code itself, in the connections to other systems, and even in the plug-ins or add-ons that users deploy. Given the number of applications in circulation—both commercially available products and custom-developed apps—the opportunity for vulnerabilities to exist is high.

Figure 6 highlights the six most common vulnerability categories reported in OSVDB; all are exploitable remotely.

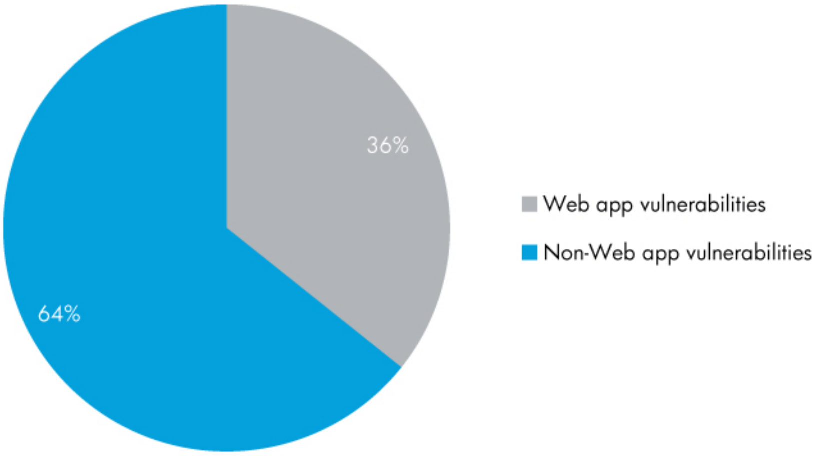
Figure 6: Most common vulnerabilities in OSVDB, broken down by category, 2000–2011



It is worth noting that four of these six vulnerabilities—SQL injection, cross-site scripting, cross-site request forgery, and remote file include—can be exclusively exploited via the Web. Other types of vulnerabilities are becoming more popular as well, including privilege escalation, memory corruption, and authentication bypass.

As shown in **Figure 7**, the four types of Web application vulnerabilities described above account for 36 percent of overall vulnerabilities by category, as reported through OSVDB.

Figure 7: Breakdown of OSVDB vulnerabilities by category for 2011



As shown in **Figure 8**, the number of Web application vulnerabilities peaked in 2006, and has since been slowly declining—at least in commercially available applications and components.

Figure 8: Year-by-year view of OSVDB vulnerabilities, broken down by type

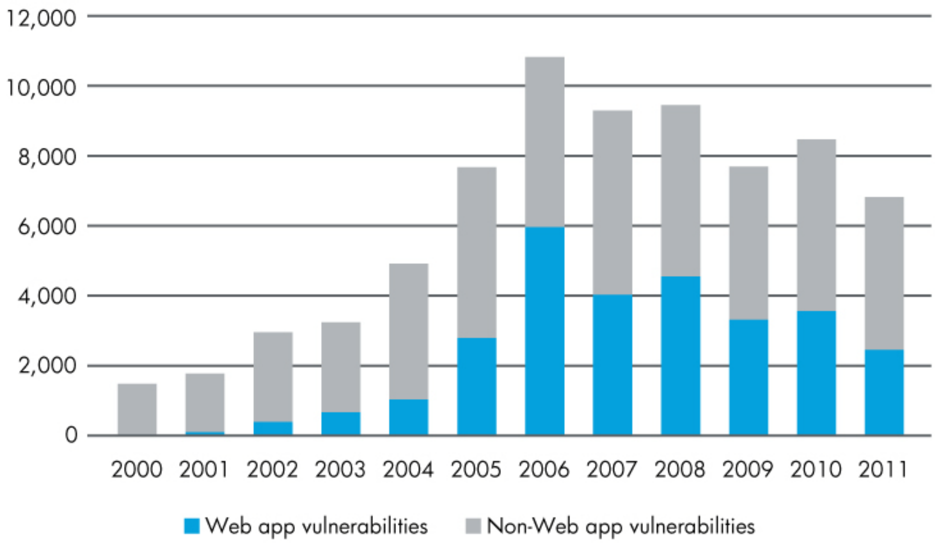
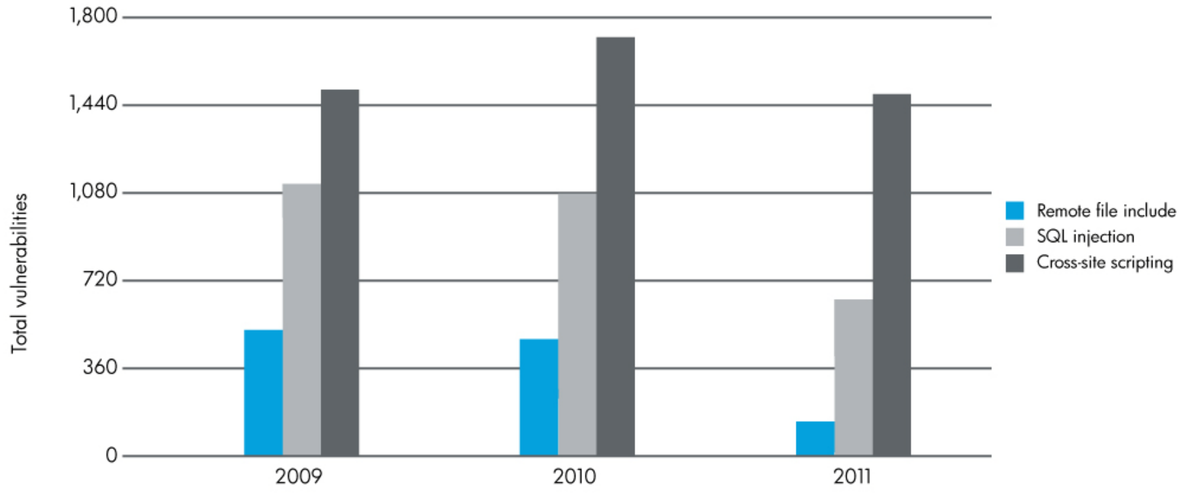


Figure 9 delves into this subject further, mapping out the individual vulnerabilities that make up the Web application category. It is clear that cross-site scripting vulnerabilities continue to have a strong presence in Web applications.

Figure 9: Categories of vulnerabilities present in Web applications based on OSVDB data, 2009–2011



Expanding the vulnerability risk surface—custom Web apps

Thus far, this report has primarily focused on vulnerability disclosure, which may or may not reflect the complete picture of vulnerability trends unfolding on the Internet.

The HP Web Security Research Group, in conjunction with the HP Fortify on Demand professional services organization, has gathered and analyzed the results from thousands of assessments to paint a more complete picture of the current state of Web application security.

Web application vulnerabilities

This set of data includes results from both dynamic and static testing. This analysis comes from many different companies, includes multiple industries, and should serve as a representative cross-sample of Web application security practices.

Static analysis: Static (or code) analysis finds security vulnerabilities by examining software without executing it.

For this sample set, the static results cover 359 unique applications. The results show that Web applications are vulnerable on multiple levels:

- 54 percent were vulnerable to reflected cross-site scripting. 40.66 percent were susceptible to persistent cross-site scripting. Persistent cross-site scripting occurs when the attack is stored on the target server in some form, such as in a database or on a submission to a bulletin board or visitor log. The victim will retrieve and execute the attack code in a browser when a request is made for the stored information.
- 86.35 percent were vulnerable to injection flaws. This category includes SQL injection and any other occurrences of an application accepting un-trusted data as part of a command or query.
- 77.16 percent were susceptible to insecure direct object reference vulnerabilities. This happens when a developer inadvertently reveals a reference to an internal implementation object, like a file or database key. Because there is no access control, these vulnerabilities can be often be leveraged by malicious users to gain unintended access to data.
- 93.87 percent were vulnerable to information leakage and improper error handling. While a specific piece of information might not seem important, it might be the one that allows an attacker to escalate a methodology and conduct a more devastating attack. At the end of the day, successful hacking is 99 percent information gathering and reconnaissance.
- 60.72 percent were vulnerable to broken authentication and session management issues. These issues can potentially be leveraged by malicious users to gain unintended application access.
- 87.74 percent were susceptible to insecure cryptographic storage. The large majority of applications in this sample set did not properly encrypt data, potentially allowing a breach in the confidentiality of the information.

Web application risk: a case study

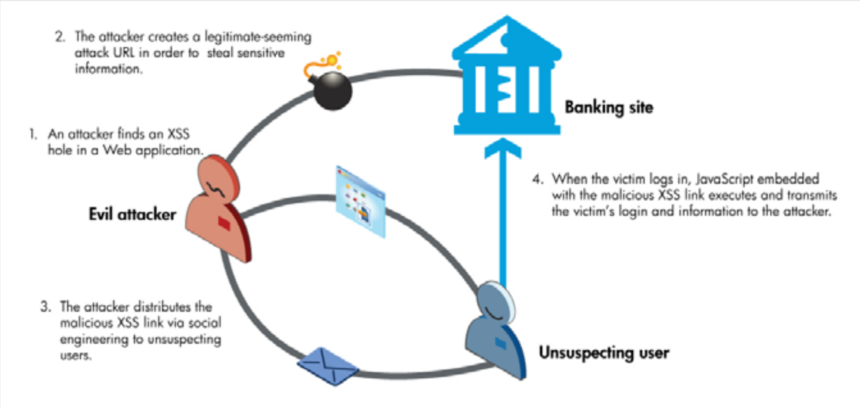
Company profile:

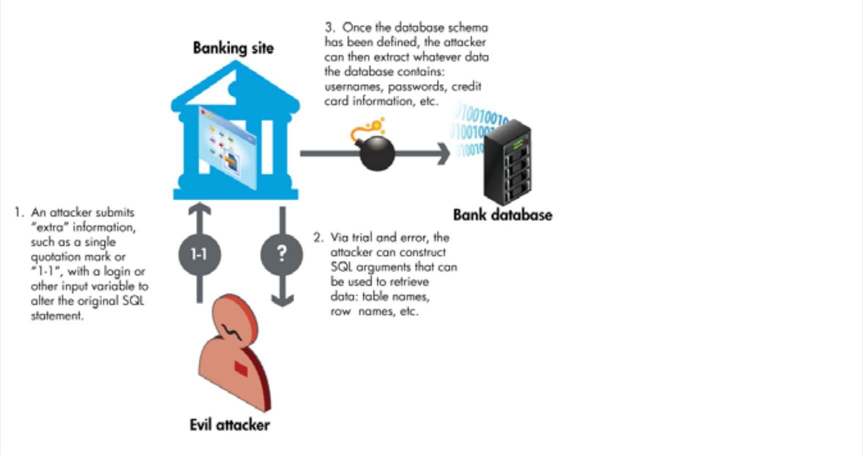
Large (more than 100,000 employees) multinational organization with revenues exceeding US$80 billion for FY2011

This company has an active security program in place. The findings indicate how much diligence it takes to fully manage application security risks. The following shows the results of a comprehensive scan of one of the organization's Web application—meaning all of the sites and pages associated with that application.

The results:

- 74 percent of the sites within the application contained cross-site scripting vulnerabilities. There were more than 56,000 instances across 1,419 unique applications.
- 38 percent of the pages on the site contained a "mixed scheme" unencrypted login form where information from an HTTP page was posted to an HTTPS page, or vice versa.
- Another 29.8 percent were vulnerable to logins sent over an unencrypted connection. This means that the form did not utilize SSL.
- 15.1 percent were vulnerable to some form of SQL injection. SQL injection can be particularly nasty, and in some circumstances it can be used to compromise a system.
- 6.1 percent of the sites contained a blind SQL injection vulnerability.
- 10.7 percent of the sites revealed information beneficial to an attacker via database server error messages.
- 17.17 percent included backup files. At a minimum, an attacker retrieving such a file would have all the information it contained, whether that be database calls, the format of parameters accepted by the application, or simply information regarding the architectural structure of the site. Often, old files are renamed with an extension such as .bak to distinguish them from production files. The source code for old files that have been renamed in this manner can often be retrieved.


2. The attacker creates a legitimate-seeming attack URL in order to steal sensitive information.
Banking site
1. An attacker finds an XSS hole in a Web application.
Evil attacker
4. When the victim logs in, JavaScript embedded with the malicious XSS link executes and transmits the victim's login and information to the attacker.
3. The attacker distributes the malicious XSS link via social engineering to unsuspecting users.
Unsuspecting user


Banking site
3. Once the database schema has been defined, the attacker can then extract whatever data the database contains: usernames, passwords, credit card information, etc.
Bank database
1. An attacker submits "extra" information, such as a single quotation mark or "1-1", with a login or other input variable to alter the original SQL statement.
1-1
?
2. Via trial and error, the attacker can construct SQL arguments that can be used to retrieve data: table names, row names, etc.
Evil attacker

Dynamic analysis: Dynamic analysis works by attacking the application under test using techniques like those a hacker might employ. It tries many attack scenarios and monitors the application's response in order to diagnose vulnerabilities. For this sample set, 2,714 unique applications were tested. The following is a high-level overview of the types of vulnerabilities discovered:

- 52.35 percent were vulnerable to reflected cross-site scripting. Insecure communications vulnerabilities affected 66.87 percent of the applications. Network traffic encryption is fundamental to protecting sensitive communications.
- 15 percent were vulnerable to information leakage and improper error handling. Information leakage covers attacks designed to acquire system specific information about a website. As stated earlier, 99 percent of hacking is information gathering, so this is not insignificant.
- Only 12.3 percent were vulnerable to injection flaws such as SQL injection. However, one SQL injection vulnerability can potentially be exploited to compromise an application, so the numbers are not reassuring.

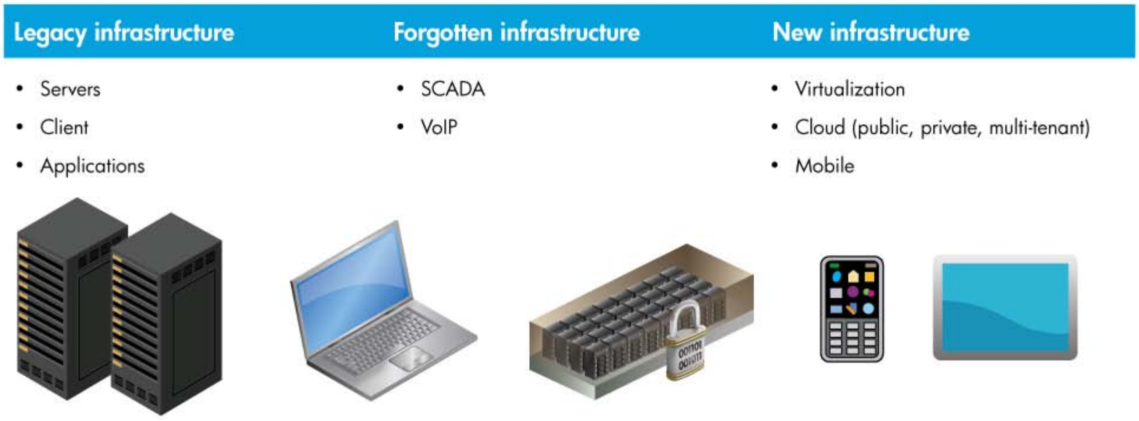
Conclusions: Overall, these sets of data show that coding mistakes that lead to security attacks are widespread. The number of security attacks on Web applications varies from year to year, but as reported here—and on the news—successful attacks are becoming much more dangerous to the business. Simple site defacement is no longer the norm. Instead, today's attacks are about stealing data for as long a period of time as possible without being detected or causing as much business interruption and brand damage as possible.

Multiple data sets confirm that cross-site scripting is a widespread and prevalent issue. As will be shown later, the Microsoft® Anti XSS filter is not significantly reducing the problem. Basic security mistakes such as information leakage and insecure communications are still being made at all organization size levels. Measures should be taken to ensure that information potentially important to attackers is not included in the application. Ultimately, the solution is for security to be "baked in" to the development process, not brushed on.

Vulnerabilities are rising in nontraditional enterprise infrastructures

Figure 10 defines legacy infrastructure as that of the physical world, the typical server/client model. Historically this type of infrastructure has been the primary target for attackers, but as infrastructure matures and changes with new technology advancements, vulnerabilities introduce additional risk.

Figure 10: Overview of the enterprise infrastructures



Legacy infrastructure	Forgotten infrastructure	New infrastructure
• Servers	• SCADA	• Virtualization
• Client	• VoIP	• Cloud (public, private, multi-tenant)
• Applications		• Mobile

As infrastructures evolve, so does the vulnerability landscape—to an extent. Let's look at each group individually:

SCADA, VoIP: the forgotten risk

The figure above uses the term "forgotten infrastructure" to describe SCADA and VoIP simply because, while there was much hype about VoIP security during the mid-2000s, there hasn't been much attacker activity in this area outside of carrier fraud. Even in the case of espionage, there are likely easier and potentially more rewarding ways of carrying out such activity than targeting the phone system.

SCADA is the ultimate example of a forgotten infrastructure. Many in the industry had stated that it would take a "digital 9/11" event to generate interest and investment in this area, and it seems that in Stuxnet, such an event finally occurred. The Stuxnet event led to a more serious discussion of what the SCADA risk really is, and what can potentially be done to combat it. Stuxnet also added the concept of "cyber war" to the vulnerability landscape—and brought about discussion of how SCADA might change in the future.

Newer infrastructure risks: virtual, cloud

As businesses adopt newer infrastructures such as virtualization, cloud, and mobile, the risk landscape is shifting. Given its relative newness, very little hard data is available on the threats posed by virtual systems—though it should be noted that the applications within these virtual and cloud environments are still susceptible to the same risks as their traditional counterparts.

Assessing the mobile risk

As smartphones and mobile devices outpace PCs, this infrastructure will have a greater impact on the security posture of the organization. In order to understand what is happening in the emerging attack vector of mobile applications, HP performed deep analysis of a number of them. Key findings include the following:

- **Mobile application security is still in its infancy.**
  The device-centric approach that antivirus companies sold five years ago (basically "drop an antivirus solution on every device") is outdated and misses all attack vectors but certain categories of malware.
- **A minimal approach is inadequate.**
  Organizations are becoming aware of the massively escalating threats inherent in a "bring your own device to work" culture and are concerned about mobile security, but are not yet sure how best to approach it. For instance, over half of the tested applications failed to properly encrypt data. The data stream could potentially be accessed by other applications. Applications must be protected from other applications stored on the device as much as from outside attackers.
- **Mobile applications are different, but the same.**
  They use different frameworks and Web services than their desktop counterparts, but mobile applications are susceptible to the same types of vulnerabilities as normal, non-mobile applications. We found multiple instances of cross-site scripting, easily broken authentication, and information leakage across the sample set.
- **Given the realities of the modern IT landscape, the information leakage problem is especially widespread and pernicious.**
  Over half of the mobile applications tested displayed information leakage behavior. The increasing demand for information from the mobile workforce—combined with increasingly pervasive cloud services and a wave of unmanaged consumer-grade devices on the corporate network—presents a noteworthy challenge for many organizations. Corporate data is being housed inside and outside the enterprise, as well as in mobile user devices.
- **In addition to the broader information leakage problems presented by changing use cases and platforms, mobile applications are designed to leak data.**
  During the course of testing, it was noted that many applications will leak various types of information from the device financial, personal, location, demographic, and even contact list data—and transmit it to third parties without permission. One very prominent free mobile phone gaming application monetizes itself by selling contact lists that often include customers, colleagues, and partners. Combine personal devices used for business purposes with the wrong kind of mobile application, and you get critical data made available to anyone willing to pay for it.
- **The number of attack types being conducted against mobile applications is on the rise.**
  Traditional man-in-the-middle attacks are being joined by newer techniques, such as mobile browser exploits that produce a remote shell through which an attacker can remotely run commands on a phone's OS. Also, multi-vector compound attacks using the Web browser, SMS, and email are appearing more frequently.
- **No platform is safe.**
  Google Android's relatively open architecture has made it a target for innovative, profit-driven attackers. For example, in March 2012 a new self-updating, remote-controlled form of Trojan malware appeared that steals banking data and targets devices running Google's Android platform. Closed platforms such as iOS, with its locked-down App Store, are harder to exploit, but there have nonetheless been successful exploits. In late 2011, one security researcher planted a sleeper application in the App Store that could phone home to a remote computer, which would then download unapproved commands and execute them, potentially leading to multiple privacy and security issues.

Attack trends

The preceding section provided a view into the vulnerability landscape—specifically where and how applications can be compromised. Knowing where the vulnerabilities exist is one component of threat prevention. Identifying the attacks exploiting those vulnerabilities and the frequency of attacks provides a greater understanding of enterprise risk. Attack data from this section is broken out into the following areas.

- **The motivations behind security attacks**
  In order to truly understand the risks associated with security attacks, it is critical to understand the motivations of the cyber actors executing those events. This section breaks down the variables that motivate attackers, including financial incentive and glory-seeking. This section also includes a brief look into the hacktivist group Anonymous.
- **A breakdown of threat data**
  The data in this section is obtained from a network of HP TippingPoint Intrusion Prevention System (IPS) devices and the HP DVLabs network of "honeypots." Included is information on the frequency and types of attacks occurring in the enterprise. This information is important to understanding the risk severity of particular vulnerabilities.
- **An in-depth look at attack techniques**
  This section takes a look at data from the previous section and provides additional analysis to show new techniques for executing attacks. Included are information on obfuscation and exploit toolkits and an in-depth look at the Blackhole toolkit, which utilizes both techniques.
- **Botnets: the universal threat**
  Botnets are a continued threat to enterprise at multiple points. There are security attacks designed specifically to grow a system of compromised hosts—and compromised systems can be bought and sold as a part of exploit toolkits to distribute attacks. This section uses data from HP DVLabs to highlight several growing botnet families.

The motivations behind security attacks

It is difficult to ascertain a clear motivation for every attack because so many variables are at play—variables that, when leveraged aptly, cloud the waters of attribution more than they would naturally. Though 2011 has been noted by many as the "Year of Anonymous," in fact more than just hacktivists were at work launching attacks, compromising hosts, and breaching enterprises around the world. Motivations and agendas vary. Individuals and groups may subscribe to singular or multiple agendas, demonstrating diverse motivation and intent, while maintaining what would otherwise be considered conflicting loyalty to various purposes or organizations. The fact remains that in 2011, as in many previous years, attacks have been launched in a variety of ways by individuals and groups for diverse purposes. The greatest motivator for attacks over the last decade has been profit. However, state-sponsored espionage, state-sponsored industrial espionage, and agenda-driven activism are also very high on the list—if not on the same level—of motivations behind today's modern, sophisticated attacks.

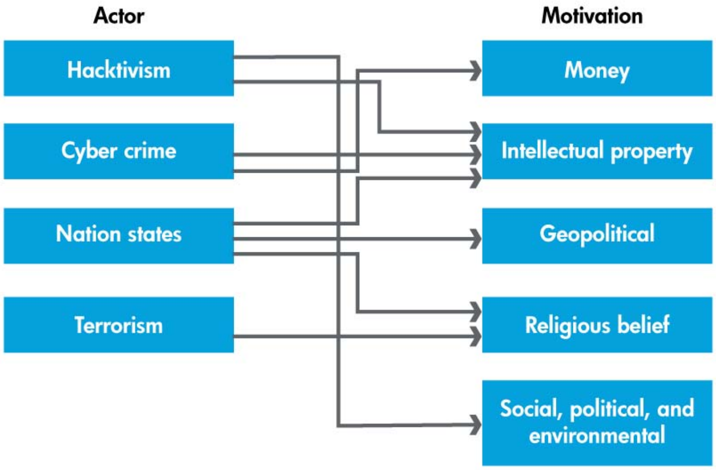
The hacktivist group Anonymous made headlines throughout 2011 with attacks against organizations and individuals alike. In the most recent large attack by Anonymous (its January 2012 response to the shutdown of Megaupload.com), many corporations and organizations associated with the recording and movie industries, as well as government agencies, were targeted[1]. These attacks focused primarily on voluntary group-based distributed denial of service (DDoS), rather than on botnet-driven DDoS. The Anonnews website reported that attack as the largest to date by Anonymous, with 5,635 participants[2].

The motives and agendas vary depending on the individual or group regardless of state or non-state sponsorship. **Figure 11** maps out the motivations for each cyber actor.

[1] http://www.us-cert.gov/cas/techalerts/TA12-024A.html

[2] https://twitter.com/#!/YourAnonNews/status/160135858895335424

Figure 11: Motivations of cyber attackers



The motives for attack include theft of money, intellectual property, and state secrets, to name a few. Religious and geopolitical beliefs are additional motives that can lead to non-kinetic or kinetic cyber attacks. Non-kinetic cyber attacks—for example, DDoS attacks—do not destroy the target, and these happen every day. Conversely, kinetic-based cyber attacks, such as Stuxnet, are specifically intended to cause damage to a physical infrastructure. The only actors sufficiently funded to carry out kinetic cyber attacks are state-sponsored entities and terrorist organizations. However, now that a decompiled version of the Stuxnet source code has been made available, it's plausible that an individual or group could execute another Stuxnet-like operation.

In the beginning of this section, we discussed motives at a high level, focusing on hacktivism with Anonymous and LulzSec.[3] According to Verizon's 2012 Data Breach Investigative Report (DBIR), hacktivism surpassed organized crime in the amount of data of stolen. More alarming is that approximately 50 percent of the corporations that were attacked had advance notice of the pending attack because they were publicly announced as targets. Often overlooked is how advancements in technology have created a low barrier of entry for an individual or group to participate in illegal online behavior. The low cost of a laptop coupled with Internet access and access to freeware tools has made it very easy for someone who believes in a cause like Anonymous to join the fight. For all that is known about Anonymous, far more is not known, because it is such a dynamic group. There is conflicting evidence of centralized leadership[4], but one thing that is known is that once ideas are agreed upon, they go viral—spreading through social media, chat rooms, Twitter, and YouTube. The rapid adoption and use of social media applications such as Twitter will continue to be used to promote various causes and cyber campaigns, often at the expense of private and public entities. As we will see in the subsequent sections, anonymity remains at the forefront of considerations for these organizations and individuals. Some are more successful at leveraging it than others.

Anonymous takes hacktivism to new levels

Anonymous generally uses three methods in its operations: DDoS, doxing, and website attacks.

In the first, a DDoS attack is set up and a call is put out to the Anonymous followers (Anons) of the operation. At a designated time, these followers will use common tools pointed at a specified target to generate traffic with such volume that the site is overwhelmed and becomes unavailable to normal users. The most common of these tools is the low orbit ion cannon (LOIC).

In the second, individuals associated with purported misdeeds will be identified and the order given to dox[5] them. Doxing—a slang term whose origins come from documents (or docs)—means to find personally identifiable information on targeted individuals and publish it. This information is then used to harass and intimidate the individuals.

In the third, once a target has been identified, the website that represents the target organization is hacked. More skilled followers will target the site with the purpose of defacing it, stealing credit card or account information, and/or deleting the site. This is often accomplished by exploiting Web application vulnerabilities such as SQL injection.

[3] http://www.esecurityplanet.com/hackers/hacktivists-organized-crime-data-theft-cybercrime-verizon-breach-report.html
[4] http://gawker.com/5783173/inside-anonymous-secret-war-room
[5] http://en.wikipedia.org/wiki/Dox

A breakdown of threat data

In the *2011 cyber security mid-year report*, the data showed that despite the decline in new vulnerabilities being reported, the number of attacks seen against those vulnerabilities continued to grow. The data pulled from all of 2011 not only supports these initial findings, but also actually demonstrates that the outlook is much worse. In the *2011 cyber security mid-year report*, all attacks tracked by the HP TippingPoint IPS were included, but we have come to the conclusion that using high-severity attack data—attacks that exploit known vulnerabilities—gives a better representation of the threat landscape, and all the data presented in this report reflects that. According to **Figure 12**, attacks measured by the HP TippingPoint IPS more than doubled in the second half of 2011.

Figure 12: Number of attacks measured by HP TippingPoint IPS, 2009–2011

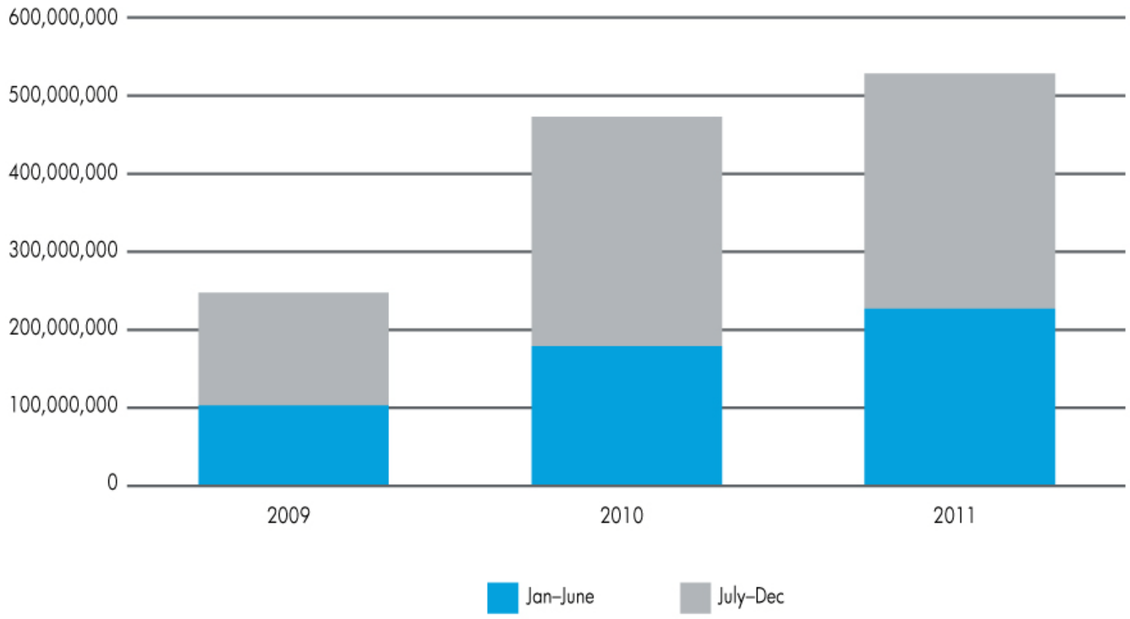
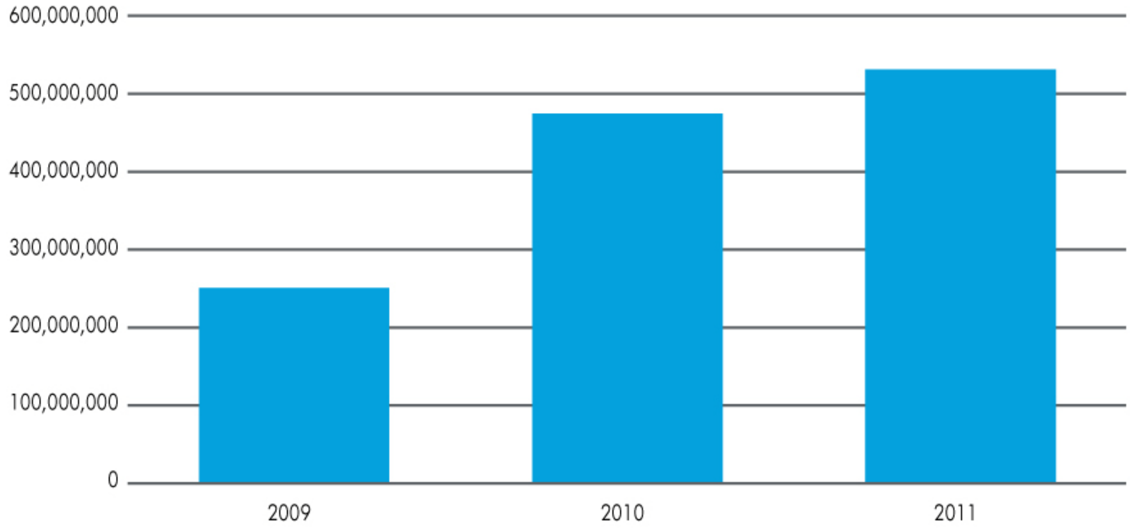


Figure 13: Total full-year attacks measured by the HP TippingPoint IPS, 2009–2011



Year by year, attacks increased but it should be noted that Web application attacks grew almost 50 percent from 2010 to 2011. Web application attacks also made up 13 percent of the total attacks observed by the HP TippingPoint IPS. This is an increase from 2010, during which the HP TippingPoint IPS measured Web application attacks making up 10 percent of overall attacks. The next section will go into more detail about the Web application attacks seen in 2011.

Figure 14: Total Web application attacks measured by the HP TippingPoint IPS, 2009–2011

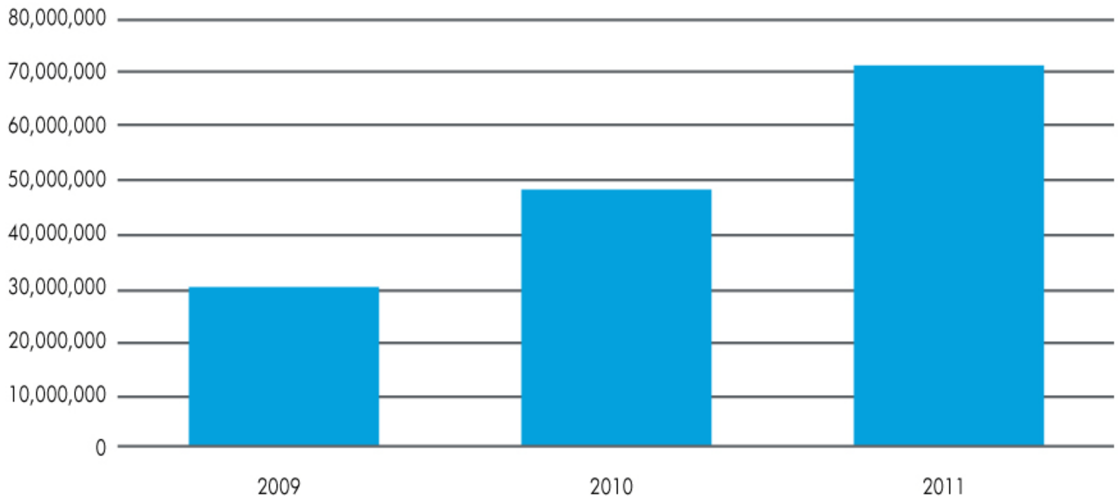
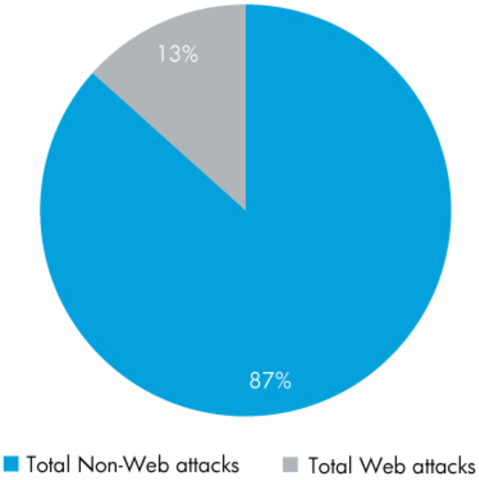
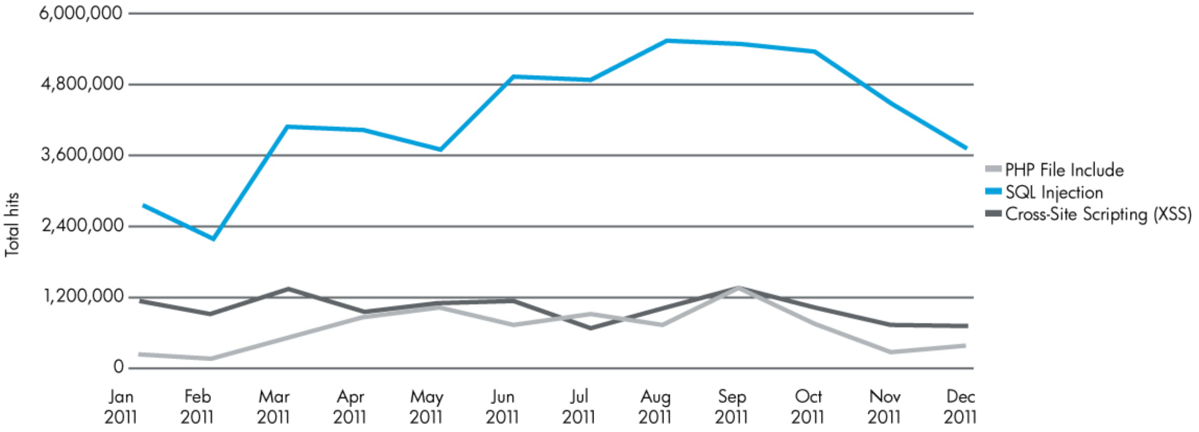


Figure 15: Breakdown of attacks measured by the HP TippingPoint IPS, 2011



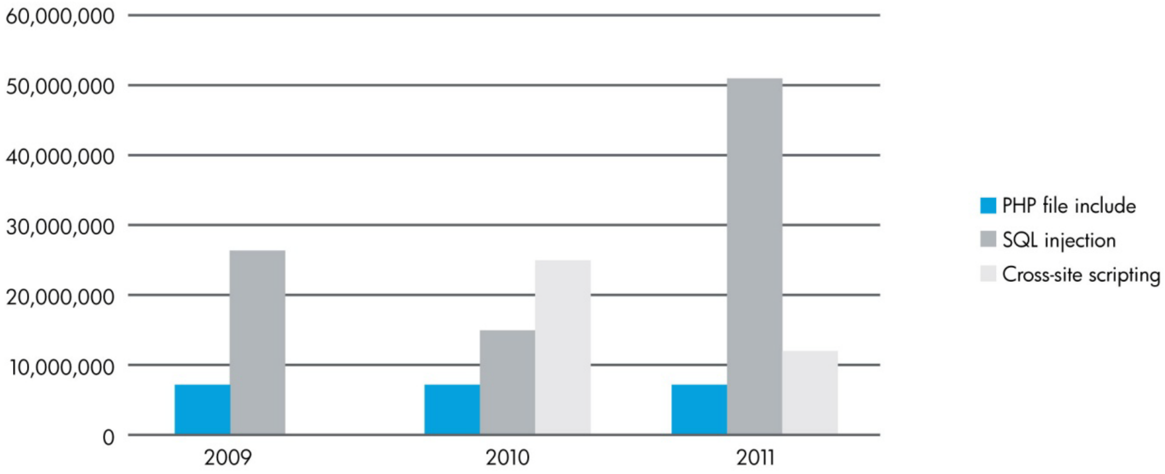
Web application attacks

Figure 16: Web application attacks by category measured by the IPS, 2011



SQL Injection has been in the news frequently over the last year and the data collected by HP TippingPoint IPS shows a significant rise in attacks, as well. As 2011 progressed, SQL Injection became one of the attack techniques of choice causing the rise in attacks shown in **Figure 16**. While XSS and RFI have remained relatively flat over the year, SQL Injection saw a steady rise throughout much of 2011 and also showed a wide variance in attack technique. The rise in SQL Injection attacks is likely due to its popularity from the Anonymous and LulzSec attacks, as well as mass SQL Injection campaigns like Liza Moon and Lilupophilupop. **Figure 17** shows a breakdown of Web application attacks over the last three years. A sharp decrease in XSS attacks is observed from 2010 to 2011, while a sharp increase in SQL Injection from 2010 to 2011 is seen after seeing a decrease from 2009 to 2010. RFI attacks remained steady from 2010 to 2011.

Figure 17: Web application attacks measured by the IPS, by category, 2009–2011



An in-depth look at attack techniques

The HP Web Security Research Group examined more than 170 applications to study certain aspects of application security more deeply than that described in earlier sections. To extend the metaphor of painting a picture, this section is a microscopic examination of several square inches of the fabric, one section at a time.

Web application attack surface

This section examines the Web application attack surface and its impact on the application’s security risk stature. For purposes of this study, the application attack surface refers to the surface prone to manipulation via user-controllable input exposed by the application developers. As Web applications become increasingly dynamic and complex, a greater attack surface is also exposed. A significant portion of the attack surface of a Web application comes from parameters that accept user input. Improper or insufficient validation of user input is by far the most common cause of Web application vulnerabilities. Developers inadvertently expose a great number of these parameters to manipulation by malicious users.

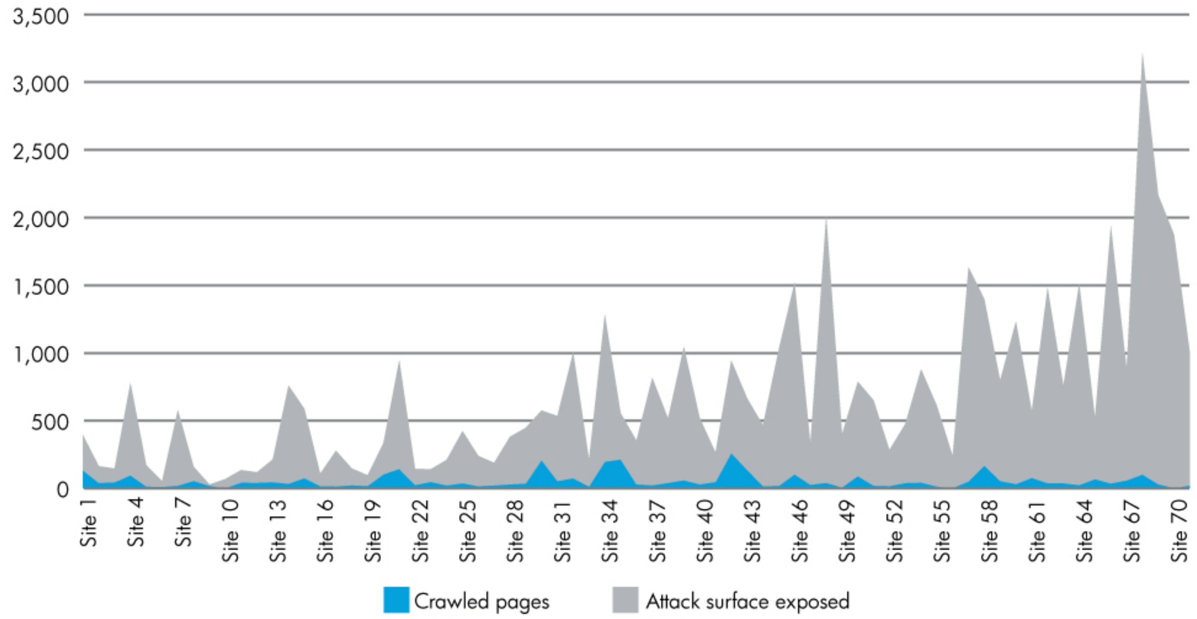
To gauge the impact, more than 70 scans of real-world Web applications were analyzed to:

- Evaluate the attack surface exposed by an application in proportion to the crawled surface
- Determine the impact of Web application size (measured by the crawled surface) and the magnitude of its dynamic nature (attack surface) on the vulnerable surface size

For this exercise, the attack surface is defined as all GET, POST, cookie, and header parameters. As well, the vulnerabilities considered in the study belonged to the OWASP Top 10 2010 categories.

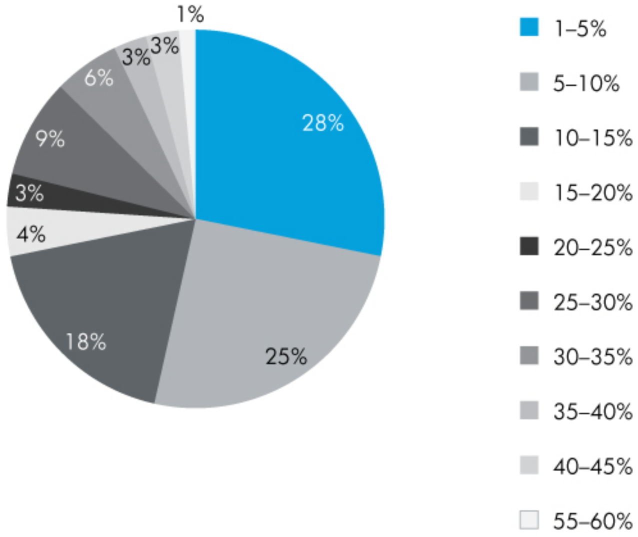
Figure 18 demonstrates a significant amount of the attack surface being exposed by the applications in the sample set. Of course, the greater the attack surface, the more control the attacker can wield over the application's behavior. A higher percentage of attack surface in the sample applications significantly improves an attacker's chances of finding an exploitable vulnerability in them.

Figure 18: Vulnerable attack surface exposed in the tested applications



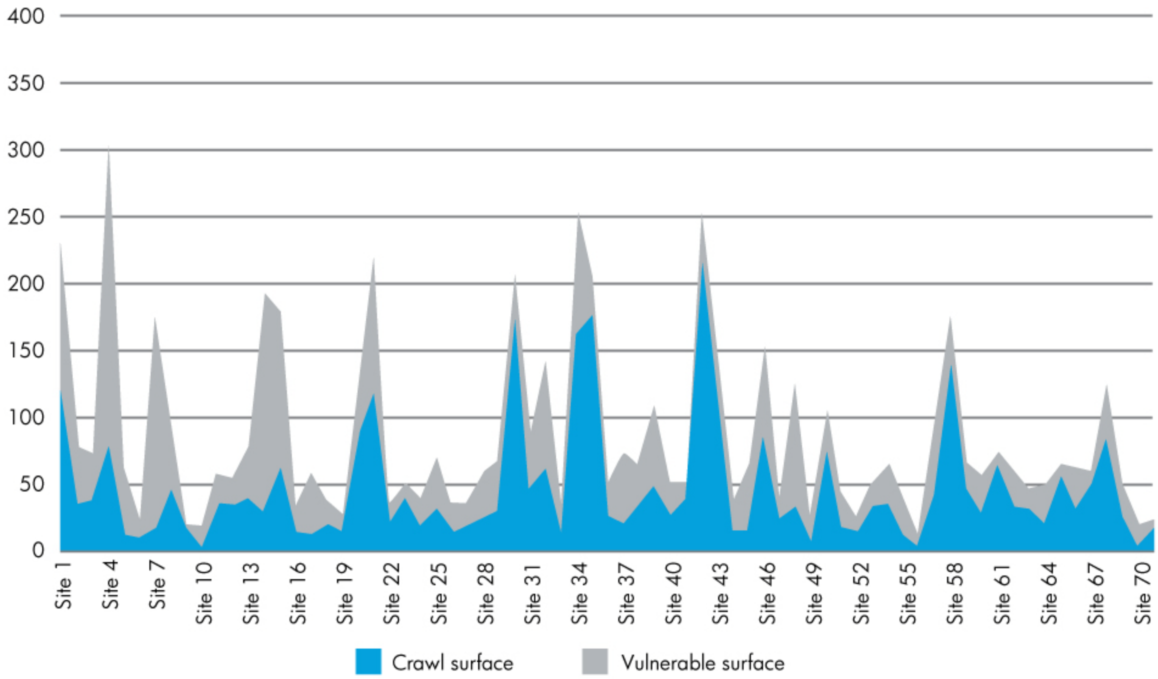
Of course, the mere existence of a parameter does not mean it's vulnerable. **Figure 19** shows the distribution of vulnerable surface over the attack surface of applications in the sample set. In a majority of cases, malicious users only need a single exploitable entry point to achieve their goals. These numbers clearly indicate that malicious users need not work too hard to discover security holes in modern Web applications, given their average size and architecture.

Figure 19: Distribution of vulnerable surface in applications in the sample



The browsing behaviors of non-malicious users differ greatly from those of attackers. Non-malicious users interact with an application only through the interfaces made explicitly available to them by the developer. This is represented by the crawl surface in **Figure 20**. Malicious users, on the other hand, are constantly looking for ways to access and manipulate interfaces or actions to which they have no authorized access. On most occasions they succeed, primarily because developers tend to expose more functionality than necessary and fail to implement proper security controls to restrict access to such functionality. This is clearly evident from a comparison of the vulnerable surface with the crawled surface that developers believe is the only accessible surface to the users.

Figure 20: Comparison of crawl surface and vulnerable surface in the sample applications



The information obtained during this analysis emphasizes the need to give security thorough consideration during design of Web applications. Limiting client-side opportunities to control application behavior will provide an additional layer of protection against users with malicious intent and reduce the effort required to implement security controls.

New features, old exploits

Developers have always wanted an open, cross-domain browser communication policy; it just makes their lives easier. While much cross-domain functionality already exists (<SCRIPT>, <IMG>, and <IFRAME> tags, for example, and other provisions such as JSONP and Flash), it is restrictive and does not allow for a free flow of information across domains. HTML5 goes a long way towards creating a more open, cross-domain policy and really helps expand the sharing between domains. This feature is called cross-origin resource sharing (CORS).

CORS allows JavaScript from a.com to read contents from b.com as long as b.com allows access to those contents from a.com or any other domain. This is achieved with the help of two special headers:

- **Origin**
  This header lists the domain where the request originated and is appended to the cross-domain request by the browser.
- **Access-control-allow-origin**
  This header contains a list of domains that are allowed to access a requested resource.

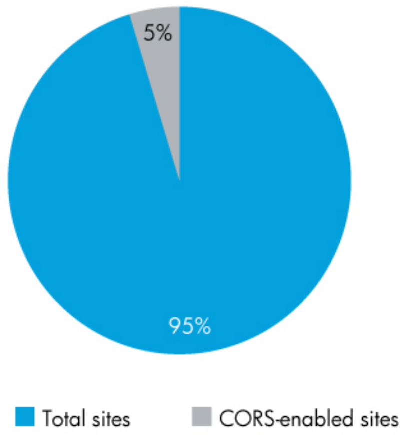
It is expected that developers will check the origin header value from the request and respond with valid contents and an access-control-allow-origin header that has the requesting domain listed if the domain from the origin header is allowed to access that resource.

Of course, the more access and ways of sharing data that are created, the more potential security issues arise. There are some significant concerns with this approach. All of these can lead to a serious compromise of the site.

- If the origin header comes from a request object and even if JavaScript cannot be used to add that to request, the header can be spoofed if the request is crafted in a program written in Java, Perl, etc.
- A response header with a "*" wild card in the response header allows the resource to be accessed from anywhere regardless of the value of the origin header. If there is an overwhelming list of origins that are allowed access, then the developer may opt for "*" instead of checking each origin.
- Depending on how many resources on a server are accessible via cross-domain access, developers may choose the option of configuring this header at the directory or domain level in the server configuration.
- Even though a site might adopt a policy that all the resources on a particular subdomain of the site are accessible from the Internet, there can be instances where the subdomain and main domain both access the same physical location and any vulnerability in the subdomain content could lead to access of all resources on the rest of the system.
- There might also be cases where the accessible subdomain is unintentionally used to store backup contents, which might be enumerated from cross-domain and end up in the wrong hands.
- Resources for an intranet might be allowed to access all the resources from an internal IP space. However, when an internal user visits a site on the Internet which contains a cross-site scripting vulnerability and exploit, that exploit might be able to enumerate anything that is accessible on the intranet and allow access to all of those internal resources. This can very well lead to targeted attacks.

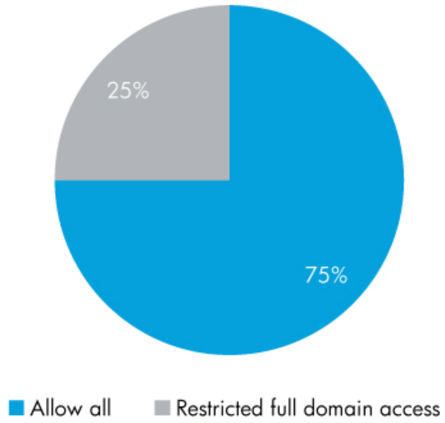
Even though HTML5 is not yet widely adopted, five percent of the sample data contained usage of access-control-allow-origin headers, indicating that there are sites adopting CORS for resource sharing, as shown in **Figure 21**.

Figure 21: CORS adoption



A high percentage of these sites utilized wildcards for the access-control-allow-origin header. This is surprising given the prevalence of vulnerabilities created by the misuse of open Flash cross-domain policies in significant security breaches.

Figure 22: CORS usage



In **Figure 22**, restricted full domain access indicates that *.domain.com is included in the access-control-allow-origin. Even though it restricts the resource usage to a particular domain, it does not completely control it—and that can be exploited if there is any vulnerability on *.domain.com.

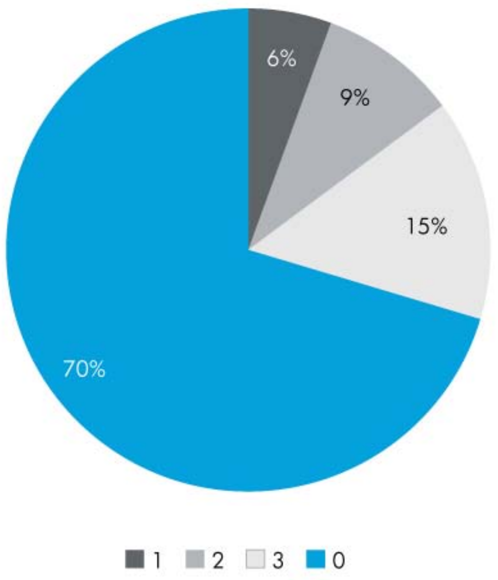
After studying what kinds of resources were shared by these sites, the data showed all of them were sharing JavaScripts, fonts, style sheets, and images. There is already a safe provision for sharing each one of these in a cross-domain manner, and CORS is not required. While it is great to see the adoption of new techniques, using them without evaluating their security impact can have dire repercussions. CORS should only be used when absolutely necessary.

Declarative security—well intended, poorly executed

Declarative security allows Web administrators or managers who are not necessarily a site's developers to configure it to prevent critical attacks against them. Starting with Internet Explorer 8, Microsoft introduced three response headers to combat specific serious security concerns. This sample set was examined to determine if declarative security was being used correctly.

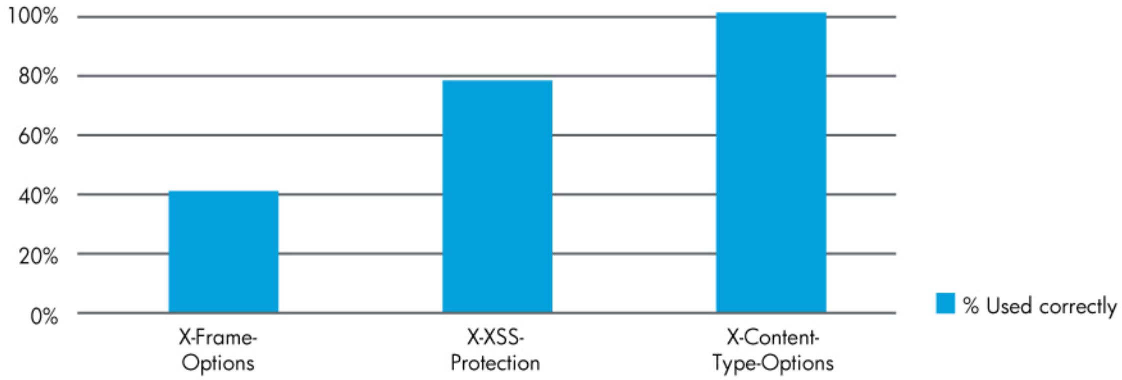
The three headers that are used by a server to help the browser perform security functions are X-XSS-protection, X-frame-options, and X-content-type-options. **Figure 23** shows the percentage of sites in the sample set that are utilizing any of these protections, and the percentage that are using none.

Figure 23: Percentage of sites that use security headers



The majority of the sites still don't take advantage of declarative security, and only 15 percent have embraced it completely. **Figure 24** shows the percentage of the sites that used each of the headers safely.

Figure 24: Percentage of sites in the sample that use security headers correctly

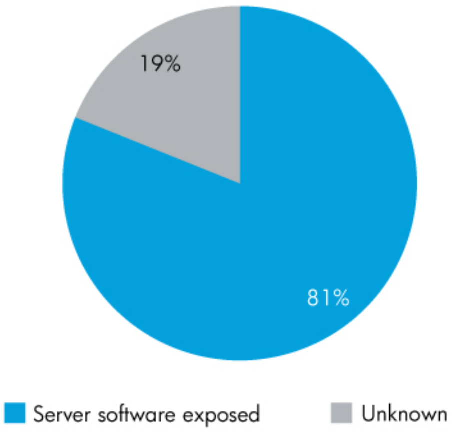


Values for X-frame-options include "Deny" to completely deny inclusion of the resource in the frame, or "SAMEORIGIN" to allow a resource to be included if it originates from the same location. There is also an option of "Allow-from origin" for this header. However, only 40 percent of the sample sites used one of these values. 60 percent end up specifying "Allow ALL"—which, from a security standpoint, is not the best option. 22 percent of the sample sites that used the X-XSS-protection header specified its value incorrectly. The intended value for this header is "1" with "block" set to block any reflective cross-site scripting attacks. However, specifying the value as "0" completely disables this filter. The X-content-type-header is the most widely adapted header. Because there is only one possible value for this, if it is implemented at all, it is implemented correctly. One possible interpretation of these results is that developers do better if there are only a few options available for a declarative security provision.

Server headers—what do they reveal?

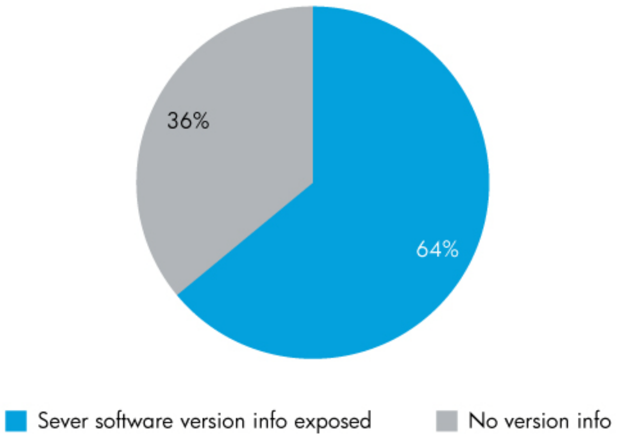
Standard security recommendations include hiding information about the server environment on which the site resides. Information leakage vulnerabilities don't receive the attention that more critical vulnerabilities do, and they are often ignored during site audits. However, a review of server headers reveals that a wealth of information of potential importance to attackers is being exposed.

Figure 25: Percentage of sites that expose server software information



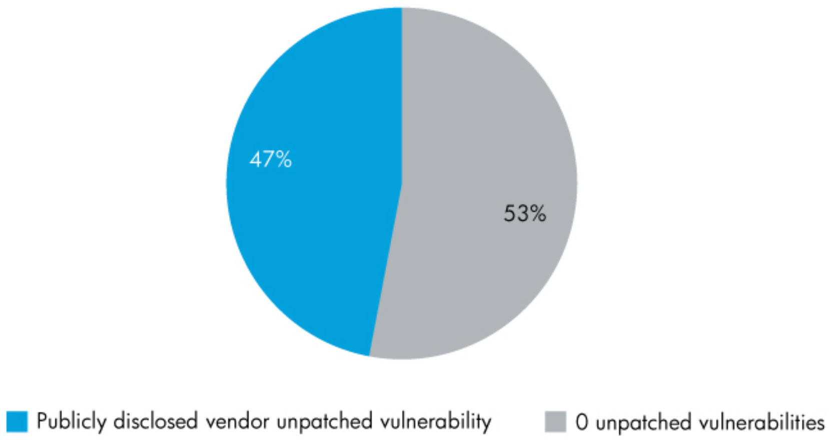
It is not a trivial task to compromise a site successfully using only the server name information. However, knowing the specific version of the particular server allows attackers to look for version-specific vulnerabilities, which makes compromise more likely. **Figure 26** shows that 64 percent of the sites that show their server information also expose their software version information.

Figure 26: Percentage of sites with both software and version information exposed



Site administrators often fail to patch their systems in a timely matter for a variety of reasons. As shown in **Figure 27**, 47 percent of the sites tested that had server software and version information exposed are potentially vulnerable to a publicly disclosed vulnerability in the commercial piece of their software. These vulnerabilities may be critical, or they may involve information disclosure.

Figure 27: Percentage of server software with at least one unpatched vulnerability (*source: secunia.com)



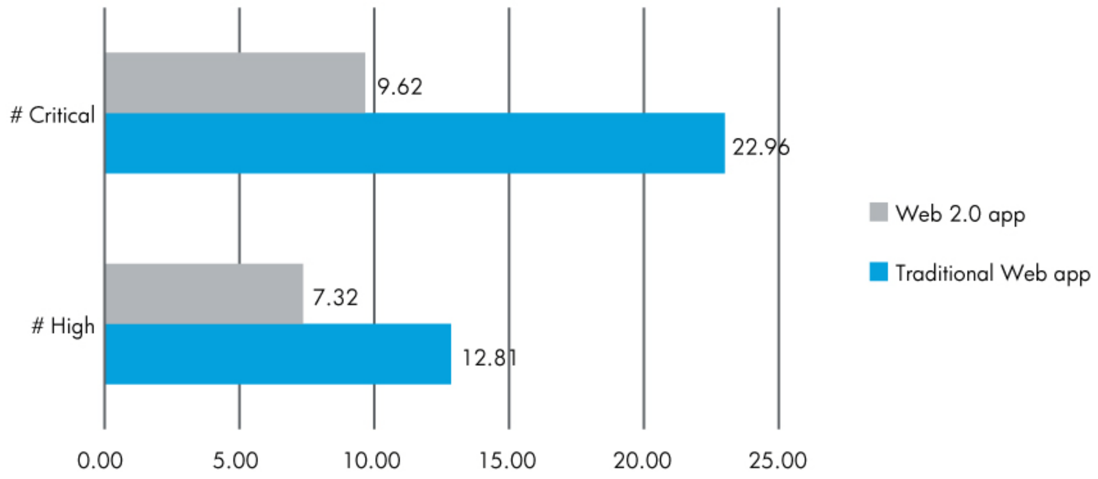
Web 2.0 applications

Web 2.0 applications are feature-rich applications that incorporate multiple, disparate Web programming technologies to maximize the user experience. A common technology often associated with Web 2.0 is AJAX, or Asynchronous JavaScript and XML. From social networking sites to mapping applications, it's no secret that the percentage of Web applications incorporating these features—either partially or completely—is on the rise.

For this study, the HP Web Security Research Group examined the sample of 111 distinct scans in order to discover how many sites incorporate Web 2.0 features. The group also examined the critical- and high-severity findings within these scans and compared the vulnerability counts discovered in Web 2.0 applications versus traditional Web applications, as shown in **Figure 28**.

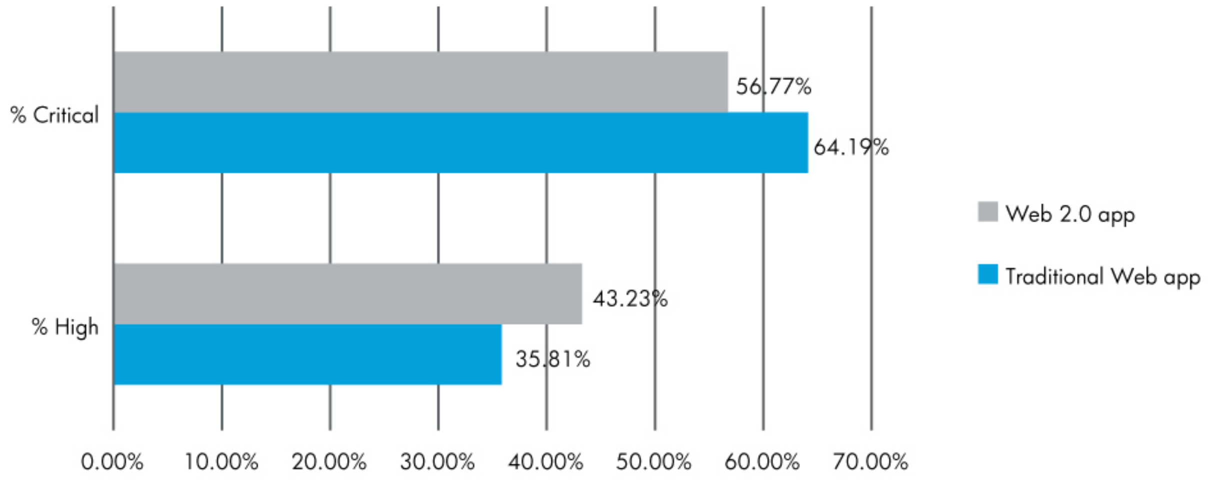
Of the total sample size, 78.38 percent were traditional Web applications; Web 2.0 applications comprised just 22.52 percent. Of the vulnerabilities present per scan, the traditional Web applications contained approximately twice as many critical and high vulnerabilities (22.96 and 12.81 percent, respectively).

Figure 28: Average vulnerabilities per scan for traditional vs. Web 2.0 applications



As shown in **Figure 29** below, the severity distribution of vulnerabilities in the sample was compared across both traditional and Web 2.0 applications. While the distribution is closely related, the actual vulnerabilities that comprise the aggregate totals vary.

Figure 29: Severity distribution



The table below lists the top six vulnerabilities by severity discovered across technologies. While SQL injection was observed in the traditional applications, it was not represented in the Web 2.0 sample size.

The top critical vulnerability across both traditional Web applications and Web 2.0 applications remains cross-site scripting, which comprised 95 percent of the vulnerabilities in Web 2.0 technologies, and 73 percent in traditional Web applications. AJAX sessions are just as susceptible to cross-site scripting vulnerabilities as are traditional HTTP sessions.

Table 1: The top six vulnerabilities by security

Traditional Web application		Web 2.0 application	
Severity	**Vulnerability**	**Severity**	**Vulnerability**
Critical	Cross-site scripting	Critical	Cross-site scripting
Critical	Database server error message	Critical	Database server error message
Critical	SQL injection	Critical	Command execution
High	Backup file disclosure	High	Unencrypted authentication
High	ASP.net filter bypass XSS	High	Backup file disclosure
High	Unencrypted authentication	High	Admin section requires authentication

It almost seems nonsensical that Web 2.0 applications would contain fewer vulnerabilities than standard applications. One possible explanation is that the composition of Web 2.0 applications broadens or obscures the attack surface, making it a little less straightforward to ensure that the application uses best practices to thwart attacks such as SQL injection and cross-site scripting.

Information leakage

For this section, developer comments were examined and analyzed for vulnerabilities. Frankly, the level and pervasiveness of the vulnerabilities were surprising. One finding that stood out was the questionable practice of encapsulating verbose stack trace messages inside HTML comments, especially stack traces that may result in a confirmed SQL injection vulnerability. This technique was represented in 13 percent of the sample set. **Table 2** details the types of discovered vulnerabilities and their severity.

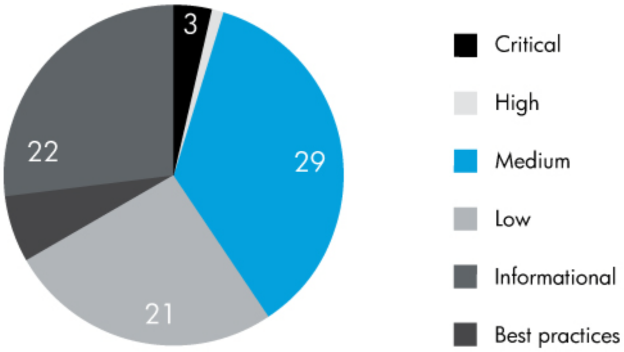
Table 2: Types of discovered vulnerabilities

Severity	Vulnerability
Critical	Database server error message
High	Unencrypted login form

Severity	Vulnerability
Medium	Java runtime error message .NET verbose errors enabled ASP.NET stack trace disclosure
Low	Possible file upload capability Internal IP disclosure Unexecuted server side include Subversion keyword information disclosure Possible server path disclosure (win32) reCAPTCHA implementation detected
Informational	Flash object detected Unsafe Flash embed settings—allow script access
Best Practices	Form auto complete active Possible insecure cryptographic hash (MD family) Possible insecure cryptographic hash (SHA-0/SHA-1)

Figure 30 represents the types of vulnerabilities found in developer comments, grouped by severity ranking.

Figure 30: Information leakage vulnerabilities by severity



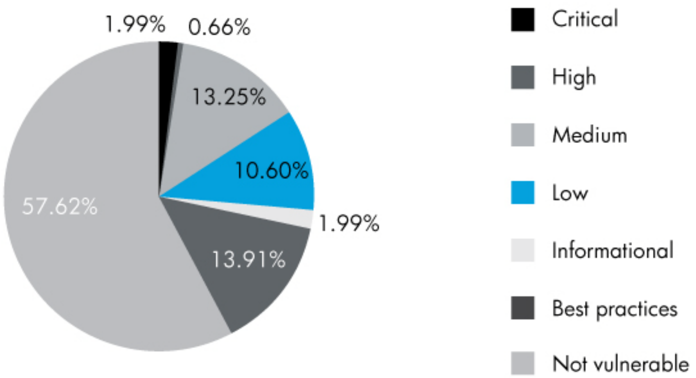
Code- and database-level error messages were discovered throughout the sampling of scans. Surprisingly, multiple applications used this security-through-obscurity approach to keep verbose error messages from being presented to the user instead of suppressing them at the application/server level.

Many if not all of the verbose error messages were generated by the scanner when testing the respective target application. Approximately 13 percent of the scans included this method to prevent error messages from being displayed to the end user.

Attackers use verbose error messages to both find and exploit applications. If verbose error messages are correctly handled by the application, the attacker will then turn to behavioral characteristics of the application that may indicate an error condition (e.g., timing-based attacks or correctly handled error messages). Developers should avoid the practice of including verbose stack messages within the HTTP responses, and especially HTML comments, as it aids attackers by providing further insight into the application's implementation.

As for the entire sample, **Figure 31** represents the percentage of scans containing each vulnerability severity, including those scans that revealed no vulnerabilities through HTML comments. Nearly 60 percent of the scans revealed no risks through HTML comments; 16 percent contained vulnerabilities of medium severity or higher.

Figure: 31: Severity distribution percentages



Typically, HTML comment vulnerability distributions fall into the information leakage threat classification. It's clear from this study that, of these information leakage classifications, much more severe attacks can result such as SQL injection, server-level compromises, or even credential theft.

Application and developer version information

A commonly encountered practice throughout the sample scan set included the discovery of plentiful comments and version information left by the developer. Version and modification-related notations ranged in severity from innocuous software version detection logic to comments referring to limitations of older versions and workarounds supplied to enable the application to function correctly.

Specific developer information, including name, employer, date of change, and other revision-related information, was also observed within the sample. This type of information can lead to further targeted attacks against the application—or provide more criteria for a determined attacker to search against for known vulnerabilities. Using specific version information from the implementation, an attacker may be able to find public vulnerability disclosure notices for the pertinent application.

Credentials and other authentication information

Authentication-related information, ranging from anonymous ftp servers to usernames and passwords for applications, was discovered throughout our sample. In certain instances, HTML comments disclosed the underlying authentication scheme's implementation, for example, by describing specific instances when LDAP vs. database authentication is used. These details may help a malicious user tailor attack payloads to increase the probability of success.

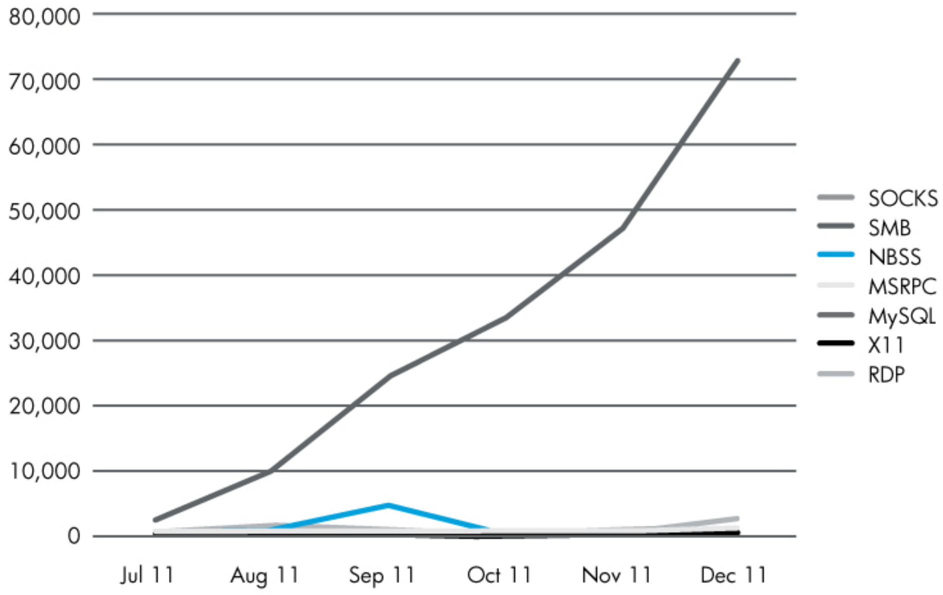
In other instances, searches against "private/public key," "username," and "password" keywords also produced a successful hit. In this specific instance, the authentication credentials were present within commented JavaScript.

Server-side attacks

The past year showed continuation of the tendency for existing vulnerabilities to be targeted much more than new ones. The most popular server-side attacks over the last year were existing ones: SQL Slammer and attacks against the MS08-067 vulnerability to spread Conficker.

As shown in **Figure 32**, the HP DVLabs honeypot network also identified an extremely large number of Conficker exploit attempts over the second half of last year. Attacks from Russia comprised the majority of the attempts, but Taiwan, Brazil, U.S., Hungary, and Romania also had very strong showings. The honeypots collected over 1,700 unique Conficker binaries that allowed for verification of existing HP TippingPoint filters. The filters that exist for protection of the MS08-067 vulnerability that Conficker exploits continue to provide protection regardless of how many variants exist, because they protect against the vulnerability and not a specific binary. The honeypots in Brazil saw by far the most Conficker attacks, but the U.S.-based honeypots were not far behind. The attacks targeting Brazil may be due to a lack of patching that may come from software piracy. Non-legitimate versions of Windows® do not allow users to retrieve important security updates, which leaves them vulnerable to many older security holes.

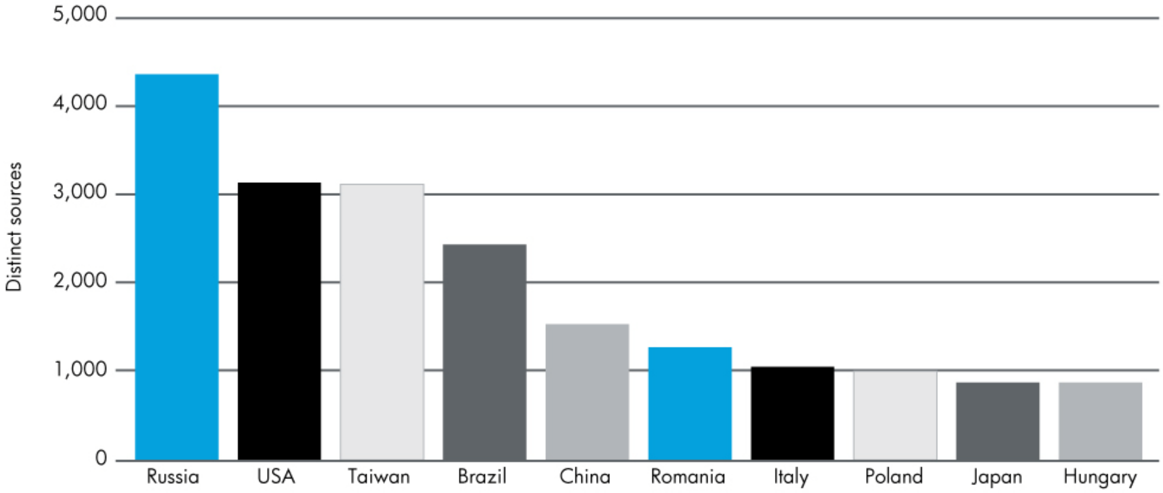
Figure 32: Port attacks by destination



Honeypot trends

Over the past year, HP DVLabs has deployed a honeypot network using cloud infrastructure in geographically diverse locations. **Figure 33** shows the number of distinct IP addresses or sources for the top 10 countries. The attacks targeting the honeypots show that, while there are some countries from which attacks appear to originate more than others, all countries are involved in attacks. Russia leads the way as the origin for the largest number of attacks, but it does not dwarf the other countries to a large enough degree to claim that it is significantly worse than the other observed origin countries. The data collected from the honeypot network shows older vulnerabilities are still extremely popular—an assertion backed up by attack data collected from the HP TippingPoint IPS network.

Figure 33: Distinct sources from honeypots



The majority of the attacks observed on the honeypots were attacks by automated tools or bots. These attacks ranged from previously mentioned attempts to exploit MS08-067 to authentication bypass attempts and brute-force login attempts. A large number of open proxy scan attempts in the Asia Pacific region were also seen. These scans may be precursors to launching a larger number of attacks upon the successful location of a usable proxy.

Attacks originating in Russia targeted SMB almost exclusively. Attacks originating in China were seen to target MS-SQL Server the most, followed by Microsoft Remote Desktop and Microsoft SMB. The MS-SQL attacks seen were attempting brute-force login attempts, and the remote desktop attacks were attempting the same. One of the most interesting statistics that the honeypot network observed was the targeting of Telnet by Sweden, specifically in the European region. Sweden was seen targeting Telnet more than any other country, and the majority of the attempts were isolated to Europe.

Client-side attacks

Web browser attacks comprised the majority of the client-side attacks seen by HP DVLabs in 2011. Internet Explorer saw the most attacks, Firefox was second, and Google Chrome and Safari were a distant 3rd and 4th. The most popular attacks seen targeting Internet Explorer were all from late 2010 or 2011. The most popular attack targeting Firefox was also attempting to exploit a vulnerability disclosed in 2011. A large number of attacks using obfuscated JavaScript to attempt to load exploits were also observed. A large number of malicious PDF attacks were also detected that targeted Adobe Reader using older vulnerabilities.

Attack techniques

Exploit kits over the past year

The past few years have seen an explosion in the popularity of exploit kits. Exploit kits were first covered in the *2010 full year HP DVLabs cyber security risks report,* and the growth in their popularity—and in the number of new exploit kits—warrants further coverage.

The past year has seen many updates of existing kits as well as the addition of many new kits. Late in the year, Chinese exploit kits started turning up for the first time. Older exploit kits such as Bleeding Life and Phoenix are still going strong; Blackhole's popularity seems to be growing exponentially; and many new kits—such as Sakura Pack, Yang Pack, and Siberia—have emerged with exploits for many recent vulnerabilities in an effort to challenge established kits such as Phoenix, Eleonore, and Blackhole.

Updates to Eleonore and Yang Pack over the last year saw each add three CVEs from 2011. Blackhole added just one 2011 CVE very late in the year, but still grew to be the most popular exploit kit observed—and that merits a look at the last year's Blackhole–related activities.

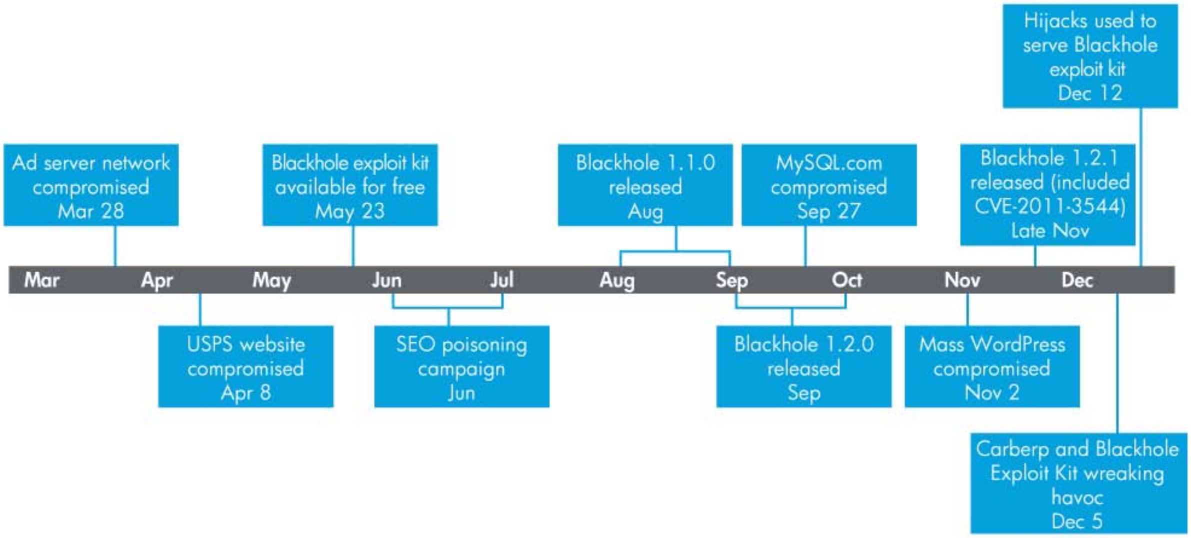
Blackhole exploit kit

The popularity of the Blackhole exploit kit increased significantly in 2011, and the kit now appears to be the exploit kit of choice for the majority of cyber criminals. Instances of compromised sites serving and/or redirecting to Blackhole sites over the last year grew dramatically. This trend was also evidenced by the increase in samples collected as the year progressed. Many high-profile sites such as the USPS and MySQL.com websites were hacked and made to serve up Blackhole pages in an attempt to infect their visitors with various malware. Blackhole was also used in a massive compromise of Wordpress sites through a vulnerable plug-in. A patch was available for this plug-in months prior to the launch of the campaign that compromised the vulnerable sites.

The surprising fact about Blackhole's success is that despite using known, patched bugs from 2010 and prior, it still achieved infection rates comparable to or better than other exploit kits tracked by HP DVLabs earlier in 2011. In late November, an exploit for CVE-2011-3544 was added that achieved a very high infection rate—above 80 percent near the end of the year, and above 70 percent in early 2012.

Blackhole's prevalence started to increase as free versions were released using a "try before you buy/fermium" model.

Figure 34: Timeline for Blackhole exploit


Hijacks used to serve Blackhole exploit kit Dec 12
Ad server network compromised Mar 28
Blackhole exploit kit available for free May 23
Blackhole 1.1.0 released Aug
MySQL.com compromised Sep 27
Blackhole 1.2.1 released (included CVE-2011-3544) Late Nov
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
USPS website compromised Apr 8
SEO poisoning campaign Jun
Blackhole 1.2.0 released Sep
Mass WordPress compromised Nov 2
Carberp and Blackhole Exploit Kit wreaking havoc Dec 5

Blackhole has also become a popular method for spreading the ZeuS, Cutwail, Spyeye, and Carberp botnets via compromised ad networks, spam campaigns, and other compromised sites. As **Figure 35** shows, the number of compromised sites serving Blackhole for the purpose of spreading Carberp variants increased in November 2011, and the increase lasted through December.

Figure 35: Elements of that make up the Blackhole exploit toolkit

Vulnerabilities	Exploit delivery	Anti-detection	Compromised hosts
CVE-2012-0507	Exploited sites	Obfuscations	Add to botnet
CVE-2011-3544	• Compromise through SQL injection, XSS, etc.	• JavaScript	• Zeus
CVE-2010-3552	Spam campaigns	• Exploit	• SpyEye
CVE-2010-1885	• Send malicious email with links to sites	• PDF	• Carberp
CVE-2010-1423			Use botnet to launch attacks
CVE-2010-0886			Rent out botnet
CVE-2010-0842			PROFIT
CVE-2010-0840			
CVE-2009-1671			
CVE-2009-0927			
CVE-2008-2992			
CVE-2007-5659			

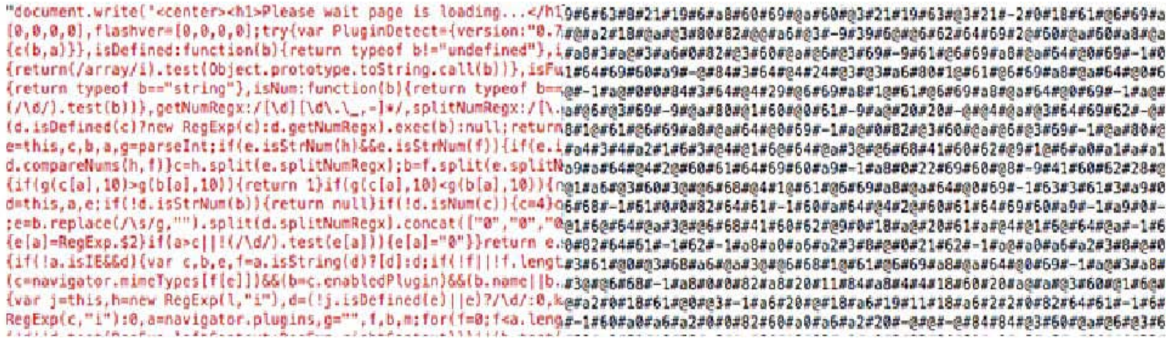
Blackhole obfuscation

One of Blackhole's strengths is that it employs constantly changing obfuscation techniques. A number of different obfuscation methods from Blackhole were seen among the samples collected from compromised sites for analysis in 2011.

These methods ranged from simple string concatenation techniques, such as "from" and "charCode," to more complex techniques requiring complex string manipulation, value adjustment through math functions, and then additional decoding. **Figure 36** shows a de-obfuscated version of Blackhole side by side with the obfuscated version, demonstrating the difference between what is rendered by a browser and what is seen on the wire.

Blackhole uses obfuscations in its PDF and Java exploits as well, but its JavaScript obfuscations are the most challenging to deal with due to the rapid changes they exhibited in 2011.

Figure 36: Blackhole exploit kit normal vs obfuscated



```
"document.write('<center><h1>Please wait page is loading...</h19#6#63#0#21#19#6#a8#60#69#@a#60#@3#21#19#63#@3#21#-2#0#18#61#@6#69#a
[0,0,0,0],flashver=[0,0,0,0];try{var PluginDetect={version:"0.7#@#a2#18#@a#@3#80#82#@@#a6#@3#-9#39#6@#@6#62#64#69#2@#60#@a#60#a8#@a
{c(b,a)}},isDefined:function(b){return typeof b!="undefined"},i#a8#3#a@#3#a6#0#82#@3#60#@a#@6#@3#69#-9#61#@6#69#a8#@a#64#@0#69#-1#0
{return(/array/i).test(Object.prototype.toString.call(b))},isFu1#64#69#60#a9#-@#84#3#64#@4#24#@3#@3#a6#80#1@#61#@6#69#a8#@a#64#@0#6
{return typeof b=="string"},isNum:function(b){return typeof b==@#-1#a@#0#0#84#3#64#@4#29#@6#69#a8#1@#61#@6#69#a8#@a#64#@0#69#-1#a@#
(/\d/).test(b))},getNumRegx:/[\d][\d\.\_,-]*/,splitNumRegx:/[\.a#@6#@3#69#-9#@a#80#@1#60#@0#61#-9#a@#20#20#-@#@4#@a#@3#64#69#62#-@#
(d.isDefined(c)?new RegExp(c):d.getNumRegx).exec(b):null;return8#1@#61#@6#69#a8#@a#64#@0#69#-1#a@#0#82#@3#60#@a#@6#@3#69#-1#@a#80#@
e=this,c,b,a,g=parseInt;if(e.isStrNum(h)&&e.isStrNum(f)){if(e.i#a4#3#4#a2#1#6#3#@4#@1#6@#64#@a#3@#@6#68#41#60#62#@9#1@#6#a0#a1#a#a1
d.compareNums(h,f)}c=h.split(e.splitNumRegx);b=f.split(e.splitNa9#a#64#@4#2@#60#61#64#69#60#a9#-1#a8#0#22#69#60#@8#-9#41#60#62#28#@
{if(g(c[a],10)>g(b[a],10)){return 1}if(g(c[a],10)<g(b[a],10)){n@1#a6#@3#60#3@#@6#68#@4#1@#61#@6#69#a8#@a#64#@0#69#-1#63#3#61#3#a9#0
d=this,a,e;if(!d.isStrNum(b)){return null}if(!d.isNum(c)){c=4}o6#68#-1#61#0#0#82#64#61#-1#60#a#64#@4#2@#60#61#64#69#60#a9#-1#a9#0#-
;e=b.replace(/\s/g,"").split(d.splitNumRegx).concat(["0","0","0@1#6@#64#@a#3@#@6#68#41#60#62#@9#0#18#a@#20#61#a#@4#@1#6@#64#@a#-1#6
{e[a]=RegExp.$2}if(a>c||!(/\d/).test(e[a])){e[a]="0"}}return e.0#82#64#61#-1#62#-1#a8#a0#a6#a2#3#8#@#0#21#62#-1#a@#a0#a6#a2#3#8#@#0
{if(!a.isIE&&d){var c,b,e,f=a.isString(d)?[d]:d;if(!f||!f.lengt#3#61#@0#@3#68#a6#@a#3@#@6#68#1@#61#@6#69#a8#@a#64#@0#69#-1#a@#3#a8#
(c=navigator.mimeTypes[f[c]])&&(b=c.enabledPlugin)&&(b.name||b.#3@#@6#68#-1#a8#0#0#82#a8#20#11#84#a8#4#4#18#60#20#a@#a#@3#60#@1#6@#
{var j=this,h=new RegExp(l,"i"),d=(!j.isDefined(e)||e)?/\d/:0,k@#a2#0#18#61#@0#@3#-1#a6#20#@#18#a6#19#11#18#a6#2#2#0#82#64#61#-1#6#
RegExp(c,"i"):0,a=navigator.plugins,g="",f,b,m;for(f=0;f<a.leng#-1#60#a0#a6#a2#0#0#82#60#a0#a6#a2#20#-@#@#-@#84#84#@3#60#@a#@6#@3#6
```

Obfuscation

Obfuscation is commonly referred to as hiding the original meaning of a message by modifying its appearance. In terms of programming, the intent of obfuscation is either to make the compiled code so difficult to understand that intellectual property theft will be hard, or to make it difficult to understand in order to evade security devices such as AV or IPS. Popular penetration testing frameworks like Metasploit have their own custom obfuscation libraries to help their exploits avoid detection when used in penetration tests.

Obfuscation has become increasingly popular among exploit toolkits, malware authors, and commercial software vendors. The number of malware samples collected over the past year that contain some form of obfuscation demonstrate how popular it has become.

JavaScript

JavaScript is a programming language that allows programmers to write code that appears to be gibberish until evaluated. Code such as the following:

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$
],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:
++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=(
$.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!
""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.
$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$
.__$+$.$$_+$._$_+$.__+"(\\\"\\"+$.__$+$.__$+$.___+$.$$$_+(![]+"")[$._$_]+(![]+"")
[$._$_]+$._$+"!\\\")"+"\"")())();
```

evaluates to:

```
alert(“Hello!”)
```

It is thus difficult to determine how malicious the code is without an evaluation.

Another trick is to take functions such as eval and then use a combination of string concatenation and reassignment to evade any detection mechanisms that specifically look for the eval statement:

```
e = eval;

e(“alert(‘hello!’)”);
```

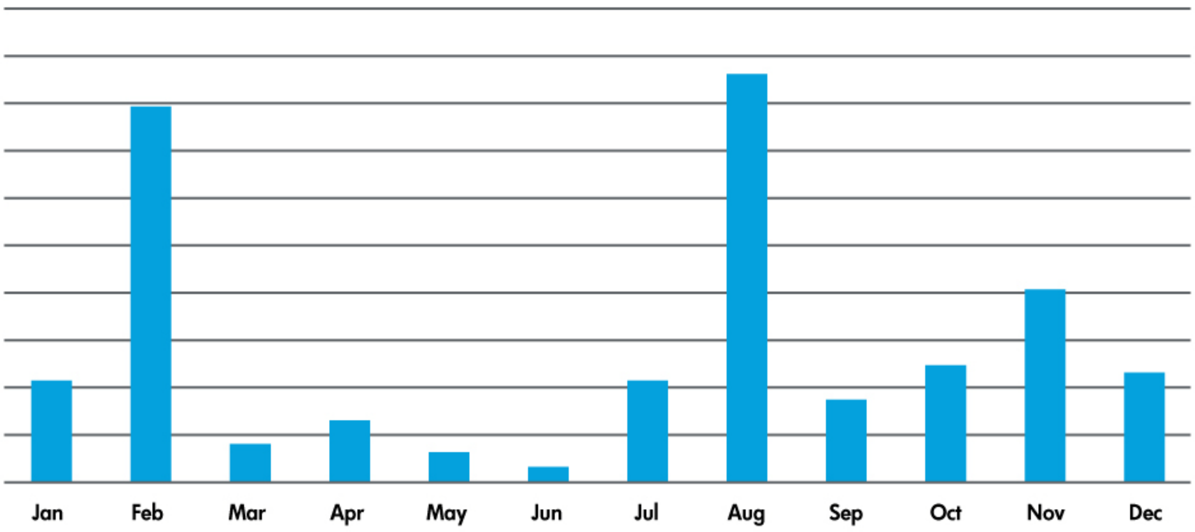
Still another option is to use character decoding from numeric values and string concatenation to create a string that will then be evaluated as valid code:

```
a=String.fromCharCode("97")+String.fromCharCode("108")+String.fromCharCode("101")
+String.fromCharCode("114")+”t”+”(1);”;

eval(a)
```

There are endless language “features” in JavaScript that can be combined to create code that, at first glance, appears to do nothing; but when evaluated, shows both valid and malicious code. A few examples were shown above, but many others exist, and more are discovered regularly. **Figure 37** shows the number of samples collected by HP DVLabs in 2011 that contained some form of obfuscation. Except for a spike in February and a dip in September, more obfuscated samples were collected in each month during the second half of 2011 than in any month in the first half of the year. The rise in exploit kit popularity and obfuscation in exploit kits may help account for this increase.

Figure 37: Malware samples observed using obfuscation 2011



PDF

Obfuscations are also becoming increasingly common in PDF documents. Using things like compressed streams and AsciiHexDecode in combination with selective character replacement makes detection of malicious PDFs more difficult, but not impossible. Some of the more sophisticated obfuscation techniques are similar to this:

```
<</Length 33225 /Filter [ /ASCIIHexDecode /LZWDecode /ASCII85Decode
/RunLengthDecode ]

>>


stream

801C8E6421988C604C1B9C8B0353994856691A958A04A340C88836868C8C033349646E3634984CA35
2B8E89A62191147A232E0DCA42F3588E22621196462631B938AC33139748 …
```

This obfuscation uses multiple layers of data encoding in combination with compression to hide the true intent.

Simple obfuscations like this:

```
/Flate#44ecode
```

also remain very popular in PDF documents as they evade any signatures that look for a static string, but are very easy to detect as very few legitimate documents do this.

PDFs gain additional benefit from every JavaScript obfuscation because JavaScript is supported in the PDF specification, so the combination of obfuscations described in the JavaScript section—along with a few other PDF techniques—can be combined to create some very interesting and malicious PDF documents.

Botnets

Introduction

In 2011, research performed by HP DVLabs showed an interesting uptick in activity related to the presence of diverse botnet families and malicious code. Botnets and malicious code and content remain a high-growth area for HP DVLabs. These events were reported from within the HP TippingPoint IPS appliance footprint—from those that have opted to share attack-related statistical data with HP DVLabs for research and trend analysis. That data—along with information harvested from HP's global threat intelligence ecosystem ThreatLinQ, its IP reputation database, and its Lighthouse program, as well as other open source data—indicates that botnet activity and growth are subject to variability and interruption but are in no way diminishing. HP DVLabs believes it is important to note that because the data reported by participating customer environments—specifically data concerning botnet-related attacks—is a representative sampling of data flowing through the total footprint of the HP TippingPoint IPS, it can infer only plausible conditions and trends while being positioned to extrapolate on a gross basis.

Botnet event statistics

The data presented in **Figure 38** represents 12 families of botnets monitored over the course of 2011.

Figure 38: Botnets and command and control (C&C)–enabled Trojans tracked by HP DVLabs

Botnets and C&C–enabled Trojans tracked by HP DVLabs	Basic characteristics
ZeuS (aliases Zbot, Kneber ZeuS)	Known colloquially as the king of the banking fraud Trojans[6]. ZeuS[7] began as a Trojan horse program[8] designed to conduct man-in-the-middle browser keystroke logging and form credential collection. In 2009, it spread wildly, with estimates of total bot members in excess of 3.6 million in the U.S. alone.[9] The ZeuS package contains a builder that can generate a bot executable along with web server files (PHP, images, SQL templates) for use as the command and control server. While Zbot[10] remains largely a generic backdoor that allows full control by an unauthorized remote user, the primary function of Zbot is financial gain—stealing online credentials such as FTP, email, online banking, and other online passwords.[11] Recent versions of the ZeuS botnet use classical copy protection mechanisms to prevent the use of unlicensed pirate copies. Some versions are key-centric and will only execute when authorized, fully credentialed system-specific key is used. A hardware configuration inventory (asset analysis) is created and leveraged by the vendor as part of the unique authenticators needed to execute the personalized key on the specified hardware platform[12]. The latest versions of ZeuS are even more sophisticated—their heightened, decentralized approach to management is characterized by peer-to-peer (P2P) communication, making detection, identification, and shutdown more difficult.[13]
Asprox (aliases Badsrc and Aseljo)	The Asprox Trojan horse program[14] was discovered in 2008 and quickly rose to prominence because it could be added via SQL injection to websites in order to promulgate malicious payloads in phishing scams and exploits. The Asprox Trojan was initially released as a spamming component of a password-stealing Trojan known as Danmec[15]. The Trojan contained many features and capabilities such as advanced SQL injection, its ability to embed malware on legitimate (non-malicious) websites using iframe tags for malicious redirection, and its fast-flux detection-avoidance capabilities[16]. In its early days, Asprox was

[6] www.infosecurity-magazine.com/view/12907/zeus-is-king-of-bank-fraud-trojan-viruses/

[7] www.fbi.gov/news/stories/2010/october/cyber-banking-fraud

[8] www.threatexpert.com/threats/trojan-spy-win32-zbot-gen.html

[9] www.hindu.com/thehindu/holnus/008200907271321.htm

[10] www.symantec.com/security_response/writeup.jsp?docid=2010-011016-3514-99

[11] http://pastehtml.com/view/1ego60e.html

[12] www.secureworks.com/research/threats/zeus/?threat=zeus

[13] www.abuse.ch/?p=3499

[14] www.symantec.com/security_response/writeup.jsp?docid=2007-060812-4603-99

[15] www.symantec.com/security_response/writeup.jsp?docid=2005-112214-3654-99

[16] www.zdnet.com/blog/security/fast-fluxing-sql-injection-attacks-executed-from-the-asprox-botnet/1122

Botnets and C&C–enabled Trojans tracked by HP DVLabs	Basic characteristics
	largely promulgated via the Pushdo/Cutwail spambot.[17,18] Approximations regarding standing bot memberships are contradictory; however, it is estimated that more than 10,000 hosts are currently active in the Asprox ecosystem[19].
SpyEye	The SpyEye Trojan[20] first came into prominence in mid-2011. A derivative of the ZeuS (Zbot) Trojan horse/botnet, SpyEye captures keystrokes and steals login credentials through a method known as "form grabbing." SpyEye will send captured data to a remote attacker, download updates, and use its rootkit component to hide malicious activity from local and network administrators[21]. The SpyEye Trojan[22] collects a great deal of information about the infected host it has compromised; it then sends that data to a command and control server for use by the botmaster[23]. The information that Spyeye attempts to gather and send to a remote server includes the following: • Bot GUID (a unique identifier associated with the malware) • Current user name • Computer name • Volume serial number • Process name associated with captured data • Name of hooked API function (for example *PR_Write*) • Keystrokes • Information specific to computer locale, such as local time and time zone • Operating system version • Language Some samples of SpyEye have been known to demonstrate advanced efficacy in injecting code into running processes and performing the following functions: • Traffic capture • Firewall bypassing • Obfuscation to prevent registry detection • Obfuscation to prevent access to the binary code • Obfuscation to prevent its process(es) within other running processes • Internet Explorer and Mozilla Firefox session captures
Sasfis	Discovered in 2010, the Sasfis Trojan[24] program opens a backdoor on the compromised computer. The Trojan may arrive as a spammed email. Once executed, it injects itself into processes running on the computer so that it can operate stealthily. It may then download more files on to the compromised computer[25]. Sasfis is typically delivered to the susceptible host via spam or drive-by infection. Once the Trojan has been installed on the compromised computer, it connects with a command and control (C&C) server to register itself as a bot. The Trojan then awaits instructions from the C&C server. These instructions often download configuration files or malware for promulgation purposes[26].
Shellbot (aliases Backdoor.Perl.Nois.a Backdoor.Perl.Shellbot.a Perl/Shellbot)	This Trojan[27] was written in Perl. It is designed to attempt to establish a connection to an Internet relay chat (IRC) server joining as a predetermined, password-enabled channel established by the botmaster[28]. Once the bot has successfully logged into the channel, it is fully enabled to receive command strings from the botmaster.
Gbot (aliases Trojan-Backdoor-Gbot)	The Gbot Trojan is a remote access tool (RAT) that may allow an attacker to gain unauthorized access to a system or enterprise. Generally, the Gbot Trojan

[17] www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Pushdo-Gen.aspx
[18] www.techrepublic.com/blog/security/pushdocutwail-botnet-second-to-none-when-it-comes-to-spamming/1637
[19] www.theregister.co.uk/2008/05/14/asprox_attacks_websites/
[20] www.threatexpert.com/threats/trojan-spyeye.html
[21] www.microsoft.com/security/portal/threat/Encyclopedia/Entry.aspx?Name=Trojan%3AWin32%2FSpyeye
[22] www.threatexpert.com/report.aspx?md5=9d2a48be1a553984a4fda1a88ed4f8ee
[23] www.eset.eu/encyclopaedia/win32-spy-spyeye-b-trojan-pincav-shd-backdoor
[24] www.threatexpert.com/report.aspx?md5=a389bdcaacd565169904707e4adc2536
[25] www.symantec.com/security_response/writeup.jsp?docid=2010-020210-5440-99
[26] www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/pay_per_install.pdf
[27] www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Shellbot-A.aspx
[28] www.mcafee.com/threat-intelligence/malware/default.aspx?id=130620

Botnets and C&C–enabled Trojans tracked by HP DVLabs	Basic characteristics
	is distributed as an embedded file infection via email. Also known as Trojan-Backdoor-Gbot[29], this Trojan may engage in any of the following activities: • Manage files on your computer, including creating, deleting, renaming, viewing, or transferring files to or from your computer • Engage a program manager that allows a hacker to install, execute, open, or close programs • Establish remote control of your cursor and keyboard • Execute and run in the background, effectively hiding its presence from anti-malware
Phatbot	Derived from the Agobot Trojan, Phatbot[30] is a backdoor that affects Microsoft Windows operating systems. Phatbot uses a client/server relationship, where the server component is installed in the victim's system and the remote attacker has control of the client.[31]
Powerbot	Powerbot is a botnet designed for use in the PC online game RuneScape.
Gumblar (aliases Troj/JSRedir-R)	Gumblar first emerged in 2009[32]. It was primarily known for hijacking users' Google searches and installing unauthorized applications and programs. It was largely promulgated via PDF exploit through SMTP communications. Once infected the installed on the host, it could conduct a variety of activities: • Enable promiscuous mode on the hosts NIC for network traffic sniffing • Enumerate host for credentials to download sites such as FileZilla, Dreamweaver, DropBox Gumblar uses passwords obtained from site admins.[33] The host site will access a website via FTP and infect the website. It will download large portions of the website and inject malicious code into the site's files before uploading the files back onto the server. The code is inserted into any file that contains a **<body>** tag, such as HTML, PHP, JavaScript, ASP, and ASPx files. The inserted PHP code contains base64-encoded JavaScript that will infect any computer executing the code. In addition, some pages may have inline code inserted into them. Typically, iframe code contains hidden links to certain malicious websites[34]. Gumblar will also modify .htaccess[35] and HOSTS files, and create images.php files in directories named "images." The infection is not a server-wide exploit; it will only infect sites on the server to which it has passwords.
Conficker A/B/C Mal/Conficker-A(Sophos) Win32/Conficker. A (CA) W32.Downadup (Symantec) W32/Downadup.A (F-Secure) Conficker. A (Pandac Net-Worm.Win32.Kido.bt (Kaspersky) W32/Conficker. Worm (McAfee) Win32.Worm.Downadup.Gen (BitDefender) Win32:Confi (avast!) WORM_DOWNAD (TrendMicro) Worm.Downadup (ClamAV)	Conficker[36] infected an estimated seven million government, business, and home computers in over 200 countries, making it the largest known computer worm infection since the 2003 Welchia. Conficker was first detected in 2007. Conficker uses flaws in Windows software and dictionary attacks on administrator passwords to propagate while forming a botnet, and has been unusually difficult to counter because of its combined use of many advanced malware techniques[37, 38]

[29] http://research.webroot.com/search.php?serialnumber=DW4ORB8V

[30] www.iss.net/security_center/reference/vuln/Phatbot_P2P.htm

[31] www.google.com/url?sa=t&rct=j&q=phatbot%20trojan&source=web&cd=3&ved=0CDoQFjAC&url=https%3A%2F%2Fsupportforums.cisco.com%2Fthread%2F61662&ei=USx2T7-LO6iYiAKn-tGnDg&usg=AFQjCNG8YvSKWiEMXGm2SV3b0qouMX3bdw

[32] http://en.wikipedia.org/wiki/Gumblar

[33] www.symantec.com/connect/blogs/gumblar-botnet-ramps-activity

[34] http://securitylabs.websense.com/content/Blogs/3401.aspx

[35] http://en.wikipedia.org/wiki/.htaccess

[36] http://mtc.sri.com/Conficker/

[37] http://en.wikipedia.org/wiki/Conficker

[38] www.confickerworkinggroup.org/wiki/

The following were among the most active botnets seen by HP DVLabs during the course of 2011:

Botnets and C&C–enabled Trojans by family

ZeuS

An interesting pattern emerged with respect to phone-home requests from the ZeuS botnet. 2011 saw a volatile pattern in ZeuS phone-home requests, as illustrated in **Figures 39** and **40**.

Figure 39: 2011 Attack trend for backdoor ZeuS botnet C&C phone-home request

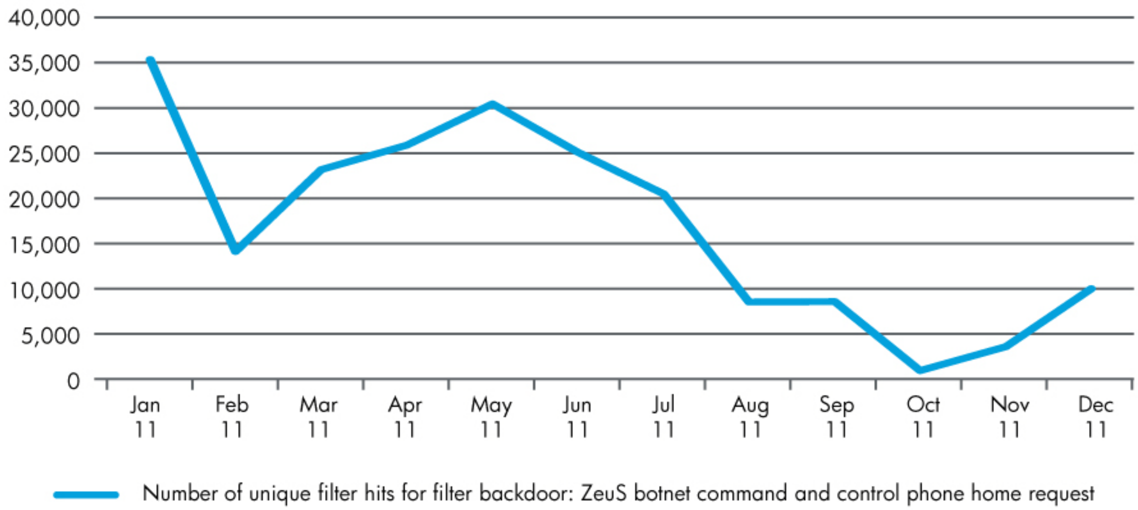
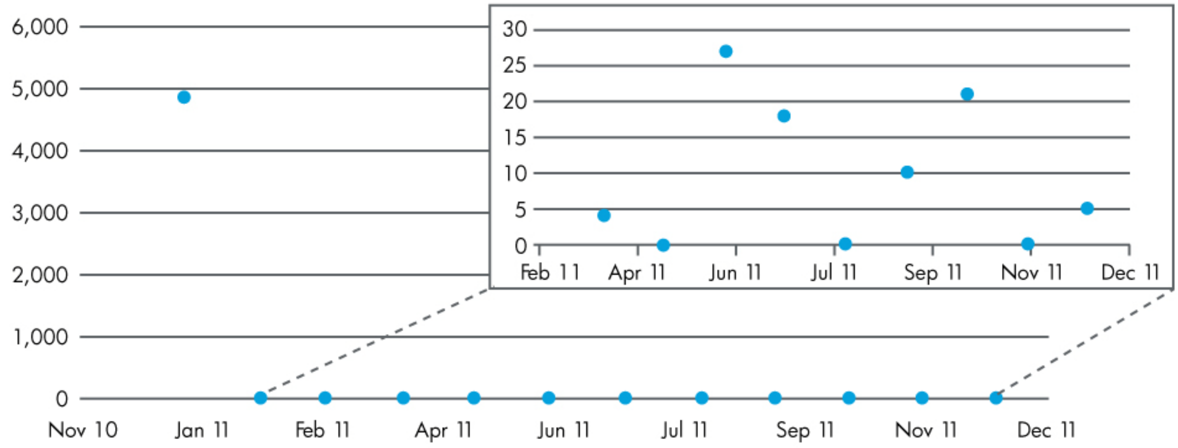


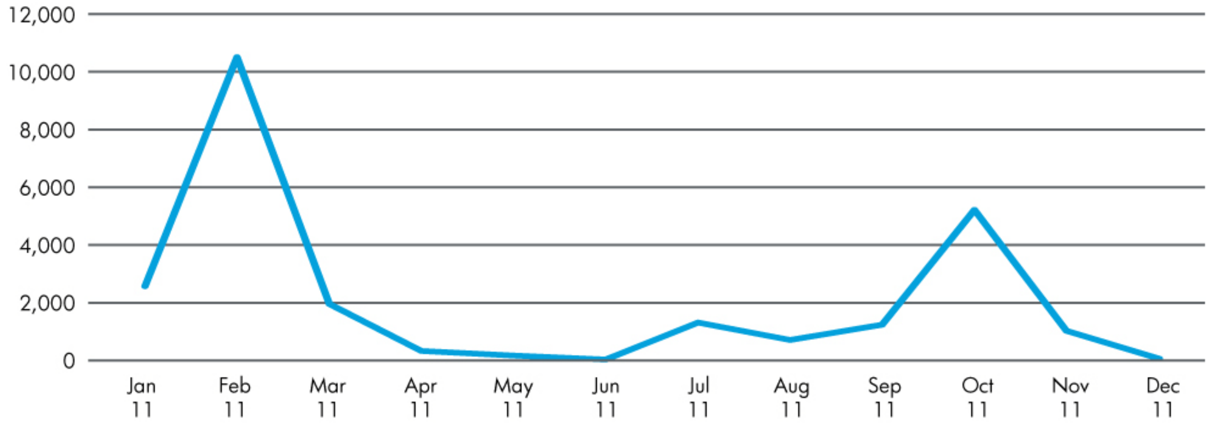
Figure 40: 2011 Hits for ZeuS botnet 2.0 phone-home request



Asprox

The Asprox botnet demonstrated a sharp decline towards the end of 2011. Interestingly enough, this was followed by a relatively sharp spike in activity beginning in August and peaking in October, as seen in **Figure 41**.

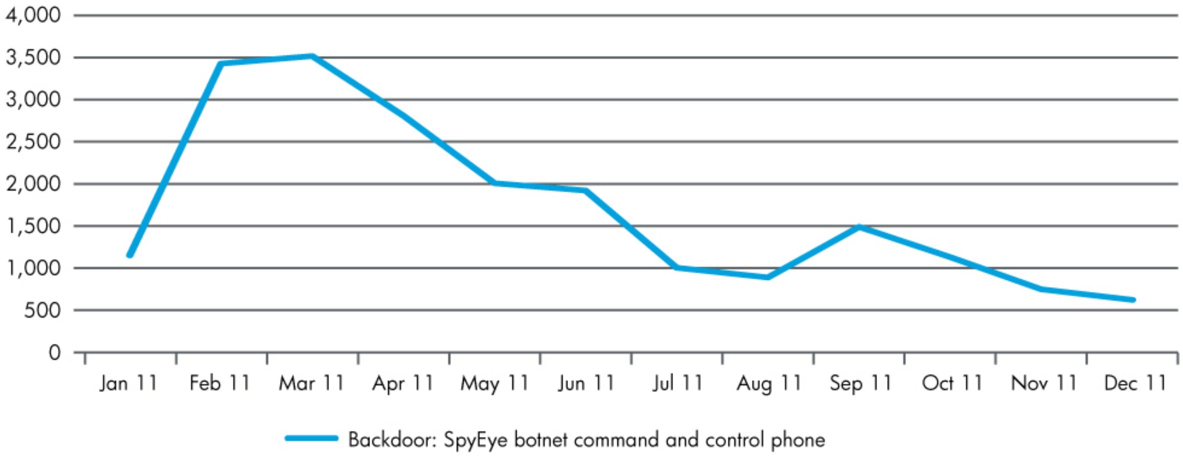
Figure 41: 2011 Number of unique attacks for HTTP: SQL Injection tool with Asprox botnet



SpyEye

As shown in **Figure 42**, 2011 data also showed a decrease in activity associated with the SpyEye botnet. The data suggests that phone-home requests from hosts compromised by SpyEye began declining in March 2011, achieved a slight plateau in June 2011, and then declined throughout the remainder of 2011.

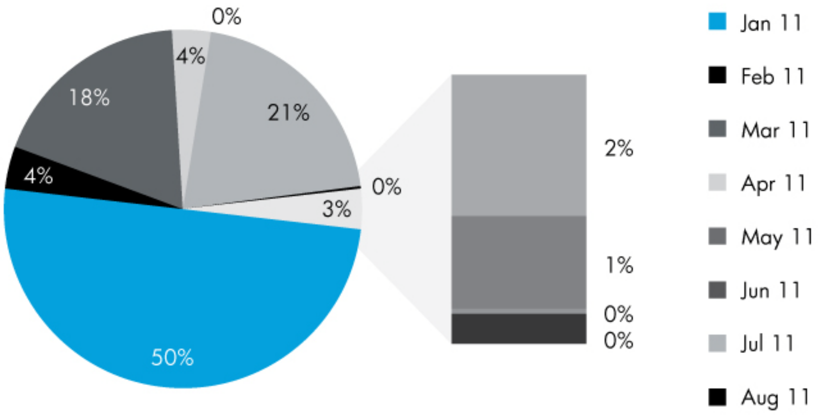
Figure 42: 2011attacks for backdoor: SpyEye botnet C&C phone-home request



Sasfis

Though a lesser-known botnet, Sasfis was active on many networks from which HP DVLabs collected data. Sasfis is often used in the propagation of malicious code and content designed or manipulated by authors who lack their own mechanisms for distribution. As shown in **Figure 43**, HP DVLabs noted that activity related to the Sasfis botnet peaked early in 2011 and tapered off towards the end of the year; little to no activity was reported beyond September 2011.

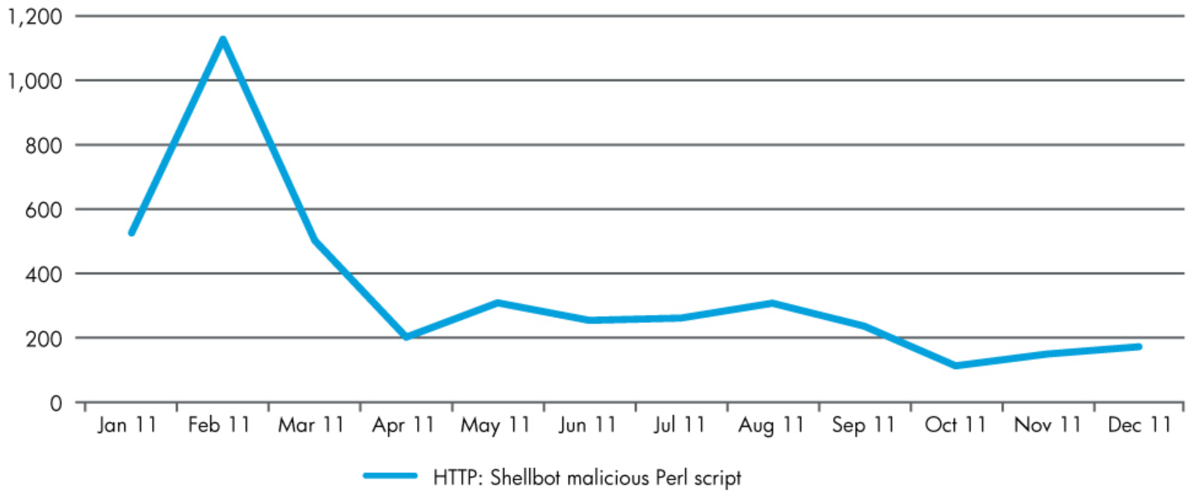
Figure 43: 2011 Attacks for Sasfis botnet C&C request



Shellbot

Activity associated with the Shellbot Trojan, a malicious Perl script, was also collected during 2011. As shown in **Figure 44**, activity associated with this botnet peaked in February 2011 and then declined swiftly thereafter.

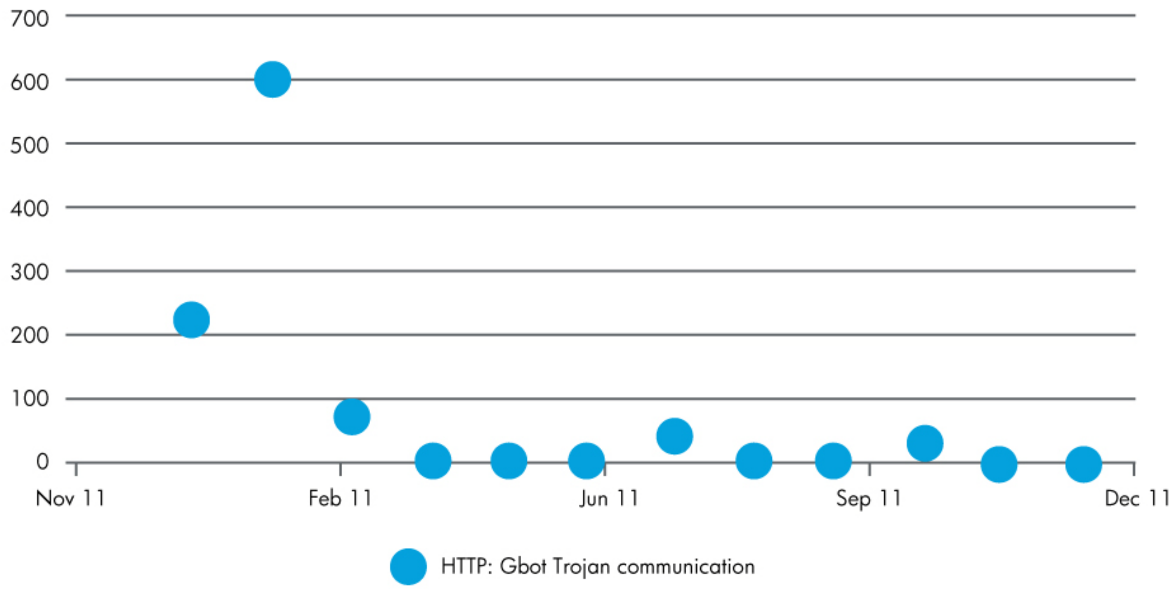
Figure 44: 2011 attacks for HTTP Shellbot malicious Perl script



Gbot

Activity associated with the Gbot botnet/C&C–enabled Trojan was also collected in 2011. This activity, as evidenced by **Figure 45**, demonstrated an erratic, relatively short-lived pattern. Communications associated with this Trojan were initially very steady for the first three months of the year; however, by April, they appeared to go completely flat. The inactivity remained until June, when activity resumed and spiked once more. Another sharp decline was noted in August, followed by a period of inactivity until September when communications spiked again. That spike lasted through November; little to no activity occurred throughout the remainder of 2011.

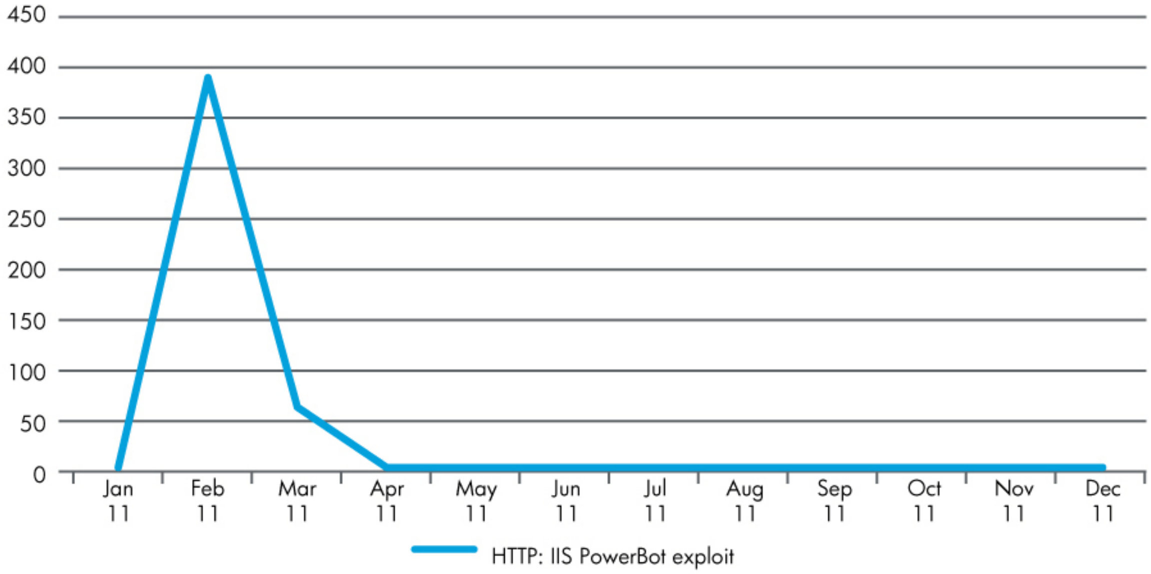
Figure 45: 2011 Attacks for HTTP Gbot Trojan communication



Phatbot

As shown in **Figure 46**, data collected by HP DVLabs showed that the Phatbot botnet exhibited activity in the first five months of 2011, but none in the latter half of the year.

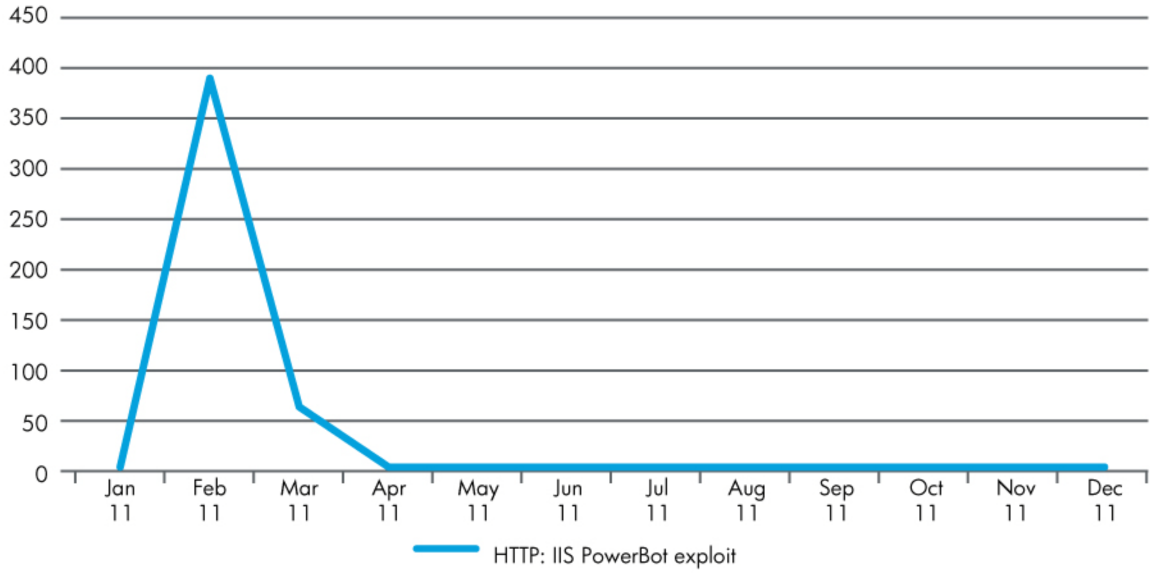
Figure 46: 2011 Phatbot activity



Powerbot

Activity associated with the PowerBot botnet during 2011, as shown in **Figure 47**, peaked and fell in less than four months.

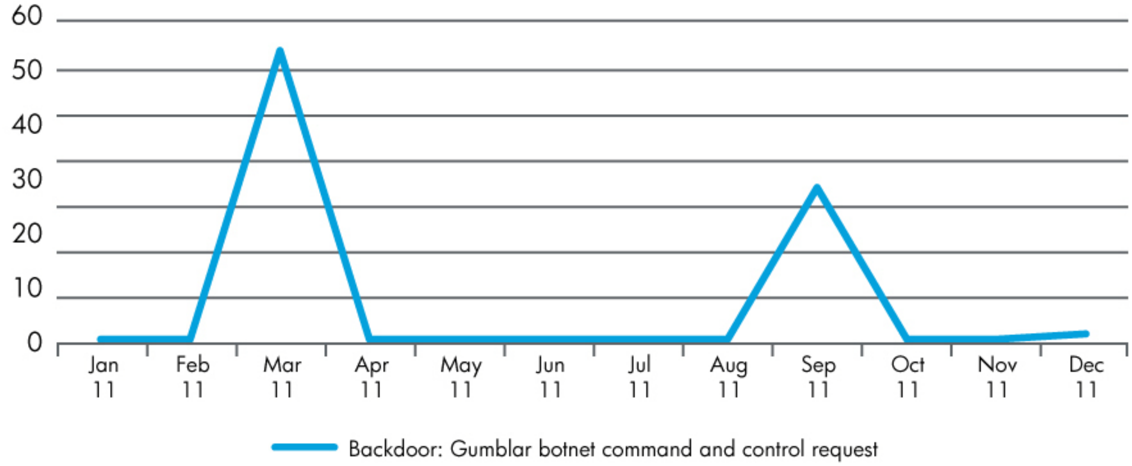
Figure 47: 2011 PowerBot activity



Gumblar

Though this botnet was quite well known and prolific in many areas of the world, HP DVLabs data on the Gumblar botnet demonstrates a limited view into its activity. As seen in **Figure 48**, March and September 2011 showed uncharacteristically high activity.

Figure 48: 2011 Gumblar activity



Mitigation thoughts

Infrastructure has been an information security industry hot topic over the last several years. Virtualization, cloud, mobile, and SCADA, the various infrastructure types have captured much attention regarding how they affect the way we view security—and how we perhaps *should* view security. In the Vulnerabilities section, legacy infrastructure is defined as that of the physical world—the typical server/client model. SCADA is a really interesting problem because ideally security technology should not be bolted on but rather embedded as much as possible into the application delivery; however, with SCADA, bolted-on security is for the time being likely to remain the only good way to tackle a very large problem.

Lastly, there are newer types of infrastructure: virtualization, cloud, and mobile. Many have pointed to the hypervisor and management aspects of virtualization and said "Watch out!" Many more have pointed to the multi-tenant aspects of cloud computing and the control and potential privacy that one gives up and said "Watch out!" With the constant proliferation of smart mobile devices, the risks are obvious; the rollout of the new model of computing and the interaction of mobile devices with applications being hosted in heavily virtualized environments are the basis for a cloud model. At the end of the day, regardless of the device being used to access this data and regardless of the model in which these applications are hosted, application access and delivery are the key to managing security risk. Therefore, the discussion of new and aging infrastructure is a healthy one. SCADA aside, the focus should not stray from the applications. This is exactly why there is such a focus on both the Web browser and Web applications, because regardless of what changes in the background, these technologies will still be used in every infrastructure type mentioned.

Alongside the discussion of infrastructure, two other topics are gaining momentum in the information security industry. One is the feeling that attackers are winning, and the second is that we should do everything in our power to make it harder for them to win—and not to allow such attackers to stand on the shoulders of the security industry.

At one recent talk, Adobe's Brad Arkin recently proposed that security researchers focus more on the defensive aspects of security rather than the offensive; raising the cost of attacks, he reasoned, should be the priority. It also seems that the buzz term of 2012 is "big data"—which can represent many different ideas depending on what the data is. In the security industry specifically, it seems that enterprises want to enable better security decisions based on their security data. The apparent need for more actionable information implies a need for actionable defenses to take advantage of this data. It is no longer enough to write more exploits to enable visibility into how insecure and potentially compromised the environment is.

Historically, internal compliance, external compliance, visibility into patching, and overall security practice and process have been driven in large part by penetration testing and other similar assessment models and technology. However, there are growing themes of "We know it's bad; do we really need to know how bad it is anymore?" and "Why aren't we preventing what we already know exists?" If security is predominantly about risk management, then why the focus on highlighting the risk that industry professionals seem to be aware of? Why not spend money and time actually reducing risk rather than producing more reports proving compliance or constantly justifying budget? Prevention can take many forms, and there is even risk in putting technology in place that can take action to reduce the attack surface available to attackers. Nevertheless, it seems that the pendulum has swung so far in the attackers' favor that the excuse of not wanting to interrupt business due to security concerns is no longer valid—just ask any of the high-profile cases of 2011.

HP DVLabs is heavily invested in vulnerability discovery and disclosure as well as overall threat landscape research and forecasting. This research has enabled a number of products in the HP security portfolio—historically intrusion prevention technology. However, some in the industry were recently confused when referencing both the ZDI and Pwn2Own during conversations about offensive security, simply because of the focus on vulnerabilities. The fact is both the ZDI and Pwn2Own are leveraged for defensive purposes via IPS protection. Assessing the impact that both have had on the rest of the industry, very little has been related to offensive security, but a great deal has had a very positive influence on the defensive aspects of security.

At the end of the day, it isn't possible to report one's way out of an attack. If all these reports and "big data" aren't both driving better security decisions and enforcing risk reduction via compensating controls, then the organization's security posture isn't maturing in line with the always-growing attacker threat that we are all working to mitigate.

Glossary

CVSS score

A standard method for rating security vulnerabilities in software systems. CVSS (for Common Vulnerability Scoring System) provides a standard set of criteria for measuring how easy a given vulnerability is to exploit and the damage that vulnerability could inflict. This score allows vendors and users to prioritize mitigation. The scoring system ranks vulnerabilities from 1 to 10, 10 being the most severe.

Command execution

A type of vulnerability that takes advantage of a lack of input validation on a website in order to run operating system commands on the vulnerable application server. Typically this vulnerability category allows attackers to exploit Web applications that pass user data as parameters to I/O operations by appending OS commands to user-supplied input using special characters such as a pipe (|).

SQL injection

A type of Web application vulnerability that takes advantage of a lack of input validation on a website in order to execute unauthorized database commands on a Web applications database server. When successfully exploited, data can be extracted, modified, inserted, or deleted from the database servers used by the vulnerable Web application. In certain circumstances, SQL injection can be used to take complete control of a system.

In an SQL injection attack, an attacker can exploit vulnerable code and the type of data an application will accept, and any application parameter that influences a database query can also be exploited. Examples include parameters within the URL itself, post data, or cookie values. If successful, SQL injection can give an attacker access to backend database contents, the ability to remotely execute system commands, or, in some circumstances, the means to take control of the server hosting the database.

Blind SQL injection

Normal SQL injection attacks depend in large measure on an attacker reverse engineering portions of the original SQL query using information gained from error messages. However, your application can be susceptible to blind SQL injection, even if no error message is displayed. The consequences of a successful attack are the same as those for regular SQL injection.

Cross-site scripting (XSS)

A type of Web application vulnerability that takes advantage of a lack of input validation to enable an attacker to inject malicious client-side code into a Web page that is viewed via a victim's Web browser. Various forms of XSS are currently being used to phish website users into revealing sensitive information such as usernames, passwords, and credit card details. XSS can generally be divided into stored, reflected, and document object model (DOM)-based attacks. Stored XSS results in the payload persisting on the target system, either in the database or the file system. The victim will retrieve and execute the attack code in the browser when a request is made for the stored information. Execution of the reflected XSS attack, on the other hand, occurs when user input from a Web client is immediately included via server-side scripts in a dynamically generated Web page. A DOM-based XSS attack relies on malicious modification of the DOM environment in a victim's browser. It differs from the stored and reflected XSS in that the malicious data is never sent to the server. Via some social engineering, an attacker can trick a victim—through means such as a malicious link or "rigged" form—into submitting information that will be altered to include attack code and then sent to the legitimate server.

Denial of service (DoS)

A type of vulnerability that allows an attacker to exhaust computer resources on a vulnerable system to a point where legitimate use of the system in question is impossible.

Distributed denial of service (DDos)

A type of DoS attack that employs a number of separate computers, which simultaneously launch denial of service attacks against a single application or system.

Remote file include

A type of Web application vulnerability that takes advantage of a lack of input validation on a website in order to execute unauthorized code (typically PHP or ASP) on a vulnerable server. Remote file-include attacks typically arise from a scripting language's inherent ability to include code from external URLs or arbitrary local files.This ability allows the attacker to include unauthorized code from an external source.

Cross-site request forgery (CSRF)

A type of Web application vulnerability that takes advantage of a lack of authorization on a vulnerable Web application to allow an attacker to execute application commands on behalf of another user of the application. The typical scenario of a CSRF attack involves an attacker tricking a victim into clicking on a link that is specially crafted to perform a malicious operation on behalf of the victim. For example, a victim may click on a malicious link that forces a transfer of money from the victim's bank account to the attacker's bank account.

Database server error

Database server error messages are returned when an unhandled exception is generated in Web application code. Unhandled exceptions are circumstances in which the application has received user input that it did not expect and does not know how to handle. When such as exception is successfully exploited, an attacker can gain unauthorized access to the database by using the information recovered from seemingly innocuous error messages to pinpoint flaws in the Web application and discover additional avenues of attack.

Get connected

hp.com/go/getconnected

Current HP driver, support, and security alerts
delivered directly to your desktop

4AA4-XXXXENW, Created September 2011