



Hewlett Packard
Enterprise

HPE Security Research

Cyber Risk Report 2015



Table of contents

3	Introduction	36	Top Android malware families in 2014
4	About Hewlett Packard Enterprise Security Research	36	Notable Android malware in 2014
4	Our data	39	Conclusion
4	Key themes		
6	The security conversation	40	Risks: Spotlight on privacy
8	Threat actors	42	Exposures
8	Nation-state supported activity	42	Emerging avenues for compromise: POS and IoT
12	The cyber underground	42	The evolution of POS malware
12	Conclusion	46	The Internet of Things
		49	Conclusion
13	Vulnerabilities and exploits	49	Controls
15	Weaknesses in enterprise middleware	50	Distribution by kingdom
15	Vulnerability and exploits trends in 2014 (Windows case)	52	Breakdown of top five Web application vulnerabilities
18	Malware and exploits	53	Top 10 Web application vulnerabilities
18	Top CVE-2014 numbers collected in 2014	55	Breakdown of the top five mobile application vulnerabilities
19	Top CVE-2014 for malware attacks	56	Top 10 mobile application vulnerabilities
20	Top CVE numbers seen in 2014	58	Open source software dependencies
22	Defenders are global	62	The Heartbleed effect
23	Conclusion	64	Remediation of static issues
		67	Conclusion
24	Threats	68	Summary
24	Windows malware overview		
27	Notable malware		
29	Proliferation of .NET malware in 2014	70	Authors and contributors
31	ATM malware attacks		
32	Linux malware		
34	Mobile malware	71	Glossary
35	Android anti-malware market		

Editors' note: While our previous Cyber Risk Reports were numbered according to the year of data covered (e.g., "Cyber Risk Report 2013" was released in 2014), we are updating our numbering convention to match industry practices.

Introduction

Welcome to the HPE Cyber Risk Report 2015. In this report we provide a broad view of the 2014 threat landscape, ranging from industry-wide data down to a focused look at different technologies, including open source, mobile, and the Internet of Things. The goal of this Report is to provide security information leading to a better understanding of the threat landscape, and to provide resources that can aid in minimizing security risk.

It is my pleasure to welcome you to our 2015 Cyber Risk Report. HPE Security Research publishes many documents throughout the year detailing our research and findings, but our annual Risk Report stands slightly removed from the day-to-day opportunities and crises our researchers and other security professionals face.

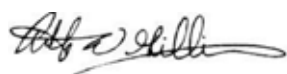
A look back at security developments over the course of a full year serves an important purpose for those charged with shaping enterprise security responses and strategies. In the wake of the significant breaches of 2014, I believe it's more important than ever that our cyber security research team continues to provide an elevated perspective on the overall trends in the marketplace.

The global economic recovery continued this year, and it was probably inevitable that as businesses rebounded, the security challenges facing them became more complex. Enterprises continued to find inexpensive access to capital; unfortunately, so did adversaries, some of whom launched remarkably determined and formidable attacks over the course of the year as documented by our field intelligence team.

Our researchers saw that despite new technologies and fresh investments from both adversaries and defenders alike, the security realm is still encumbered by the same problems—even in some cases by the very same bugs—that the industry has been battling for years. The work of our threat research and software security research teams revealed vulnerabilities in products and programs that were years old—in a few cases, decades old. Well-known attacks were still distressingly effective, and misconfiguration of core technologies continued to plague systems that should have been far more stable and secure than they in fact proved to be.

We are, in other words, still in the middle of old problems and known issues even as the pace of the security world quickens around us. Our cyber security research team has expanded over the course of the year, and so has this Risk Report, both covering familiar topics in greater depth and adding coverage of allied issues such as privacy and Big Data. In addition, our people work to share their findings and their passion for security and privacy research with the industry and beyond. This Risk Report is one form of that; our regular Security Briefings and other publications are another form, and we hope to remain in touch with you throughout the year as themes presented in this Report are developed in those venues.

Security practitioners must ready themselves for greater public and industry scrutiny in 2015, and we know that threat actors—encouraged by public attention paid to their actions—will continue their attempts to disrupt and capitalize on bugs and defects. The HPE Security Research group continues to prepare for the challenges the year will doubtless pose, and also intends to invest in driving our thought leadership inside the security community and beyond it.



Art Gilliland

SVP and General Manager, Enterprise Security Products

About Hewlett Packard Enterprise Security Research

HPE Security Research (HPSR) conducts innovative research in multiple focus areas. We deliver security intelligence across the portfolio of HPE security products including HPE ArcSight, HPE Fortify, and HPE TippingPoint. In addition, our published research provides vendor-agnostic insight and information throughout the public and private security ecosystems.

Security research publications and regular security briefings complement the intelligence delivered through HPE products and provide insight into present and developing threats. HPSR brings together data and research to produce a detailed picture of both sides of the security coin—the state of the vulnerabilities and threats comprising the attack surface, and, on the flip side, the ways adversaries exploit those weaknesses to compromise victims. Our continuing analysis of threat actors and the methods they employ guides defenders to better assess risk and choose appropriate controls and protections.

Our data

To provide a broad perspective on the nature of the attack surface, the report draws on data from HPE security teams, open source intelligence, ReversingLabs, and Sonatype.

Key themes

Theme #1: Well-known attacks still commonplace

Based on our research into exploit trends in 2014, attackers continue to leverage well-known techniques to successfully compromise systems and networks. Many vulnerabilities exploited in 2014 took advantage of code written many years ago—some are even decades old. Adversaries continue to leverage these classic avenues for attack. Exploitation of widely deployed client-side and server-side applications are still commonplace. These attacks are even more prevalent in poorly coded middleware applications, such as software as a service (SaaS). While newer

exploits may have garnered more attention in the press, attacks from years gone by still pose a significant threat to enterprise security. Businesses should employ a comprehensive patching strategy to ensure systems are up to date with the latest security protections to reduce the likelihood of these attacks succeeding.

Theme #2: Misconfigurations are still a problem

The HPE Cyber Risk Report 2013 documented how many vulnerabilities reported were related to server misconfiguration. The trend is very similar again in 2014, with server misconfiguration being the number-one issue across all analyzed applications in this category. Our findings show that access to unnecessary files and directories seems to dominate the misconfiguration-related issues. The information disclosed to attackers through these misconfigurations provides additional avenues of attack and allows attackers the knowledge needed to ensure their other methods of attack succeed. Regular penetration testing and verification of configurations by internal and external entities can identify configuration errors before attackers exploit them.

Theme #3: Newer technologies, new avenues of attack

As new technologies are introduced into the computing ecosystem, they bring with them new attack surfaces and security challenges. This past year saw a rise in the already prevalent mobile-malware arena. Even though the first malware for mobile devices was discovered a decade ago, 2014 was the year when mobile malware stopped being considered just a novelty. Connecting existing technologies to the Internet also brings with it a new set of exposures. Point-of-sale (POS) systems were a primary target of multiple pieces of malware in 2014. As physical devices become connected through the Internet of Things (IoT), the diverse nature of these technologies gives rise to concerns regarding security, and privacy in particular. To help protect against new avenues of attack, enterprises should understand and know how to mitigate the risk being introduced to a network prior to the adoption of new technologies.



Theme #4: Gains by determined adversaries

Attackers use both old and new vulnerabilities to penetrate all traditional levels of defenses. They maintain access to victim systems by choosing attack tools that will not show on the radar of anti-malware and other technologies. In some cases, these attacks are perpetrated by actors representing nation-states, or are at least in support of nation-states. In addition to the countries traditionally associated with this type of activity, newer actors such as North Korea were visible in 2014. Network defenders should understand how events on the global stage impact the risk to systems and networks.

Theme #5: Cyber-security legislation on the horizon

Activity in both European and U.S. courts linked information security and data privacy more closely than ever. As legislative and regulatory bodies consider how to raise the general level of security in the public and private spheres, the avalanche of reported retail breaches in 2014 spurred increased concern over how individuals and corporations are affected once private data is exfiltrated and misused. The high-profile Target and Sony compromises bookended those conversations during the period of this report. Companies should be aware new legislation and regulation will impact how they monitor their assets and report on potential incidents.

Theme #6: The challenge of secure coding

The primary causes of commonly exploited software vulnerabilities are consistently

defects, bugs, and logic flaws. Security professionals have discovered that most vulnerabilities stem from a relatively small number of common software programming errors. Much has been written to guide software developers on how to integrate secure coding best practices into their daily development work. Despite all of this knowledge, we continue to see old and new vulnerabilities in software that attackers swiftly exploit. It may be challenging, but it is long past the time that software development should be synonymous with secure software development. While it may never be possible to eliminate all code defects, a properly implemented secure development process can lessen the impact and frequency of such bugs.

Theme #7: Complementary protection technologies

In May 2014, Symantec's senior vice president Brian Dye declared antivirus dead¹ and the industry responded with a resounding "no, it is not." Both are right. Mr. Dye's point is that AV only catches 45 percent of cyber-attacks²—a truly abysmal rate. In our review of the 2014 threat landscape, we find that enterprises most successful in securing their environment employ complementary protection technologies. These technologies work best when paired with a mentality that assumes a breach will occur instead of only working to prevent intrusions and compromise. By using all tools available and not relying on a single product or service, defenders place themselves in a better position to prevent, detect, and recover from attacks.

¹ online.wsj.com/news/article_email/SB10001424052702303417104579542140235850578-IMyQjAxMTA0MDAwNTEwNDUyWj.

² securitywatch.pcmag.com/security/323419-symantec-says-antivirus-is-dead-world-rolls-eyes.



“Malware” was the top key word of 2014, outstripping even “security” as a favored key word

The security conversation

Reflecting on the 2014 threat landscape we undertook a broad top-level look at public security research and analysis published in 2014, using key word analytics targeting specific concepts.

As befitting a look at high-profile trends, our data was drawn strictly from sources available on the public Internet. The first set of data was drawn from the press covering the industry as well as other sources. We drew the second set from content presented at industry conferences such as BlackHat, DefCon, and Virus Bulletin. The yearly Cyber Risk Report is time-bound and so we resolved to do a time-oriented analysis.

Working within that dataset, we analyzed two sets of terms for their frequency of appearance. The first set, the key words, are the security-associated words more familiar to a general audience; for instance, *attack*, *threat*, or *targeted*. These terms are also more likely to appeal to headline writers, because what they lack in specificity they make up for in brevity and “oomph.” The second set, the key phrases, describe more granular and complex concepts that tend to be used mainly by security practitioners. *Exploit kit* and *C&C server* are two examples of key phrases. This distinction allowed us to approach the data in a progression from less to more specificity. Between the two, we started our analysis with approximately 10,000 words and phrases we found to be of interest.

Our first dive, “total 2014+2013,” looked at which topics rose and fell in the English-language trade press over the last 24 months. If we assume that trade journalism is a good mirror of what’s actually happening in the real security world, it should follow that the frequency of key words and key phrases in the press is a good indicator of what those in the industry are thinking about.

One of the strengths of Big Data is its predictive power. From our 2013+2014 results, we made linear extrapolations to see

what might lie ahead in 2015, assuming that what is rising will continue to rise and what is falling will continue to fall.

Our analysis indicated that breaches and malware were weighing heavily on our minds in 2014. “Malware” itself was the top key word of 2014 (and of 2013), outstripping even “security” as a favored key word and making bold progress among security practitioners as part of the key phrase “malware family.” Key phrase analysis indicated that conversation about mobile malware, particularly Android malware, was rising even as the more neutral phrase “mobile devices” fell. The efficacy of anti-malware software was debated in 2014, but the analysis indicates that malware as a hot topic isn’t going anywhere anytime soon.

Digging a bit deeper, we returned to our lists of key words and key phrases and asked who “won” 2014—the good guys, the bad guys, or no one in particular. At this point human intervention was necessary, and we hand-sorted terms into categories of “good guys,” “bad guys,” and “neutral” in order to perform categorical analysis as to whether attackers or defenders were better represented over the course of the year.

We found that security experts’ view of the world may in fact be a bit dimmer than that of the general public. Though the public (as seen through our key words) was concerned about things such as malware (#1 on their list), attacks (#3), and exploits (#5), by and large consumers seemed to use fairly neutral terms when diving into security-related topics online.

The pros, on the other hand, are a skeptical lot. We classified nearly half of the most popular key phrases as negative in tone. The value-neutral “operating system” led the pack, but after that the misery began with “targeted attacks” (#2), “exploit kit” (#3), “social engineering” (#5), and “C&C server” (#6) and continued from there. Interestingly, the key phrase “security researchers” nearly doubled in usage between 2013 and 2014, while the more familiar term key word “hackers” turned in steady usage numbers and barely outperformed the longer phrase.

Of course, one can always argue that the bad guys get more attention because they are bad, and that it is merely human nature to take an interest in things that might be harmful. But, we asked ourselves, do people actually learn anything from all the excitement? Once again we turned to our data, asking which breaches and vulnerabilities caused the most excitement in 2014.

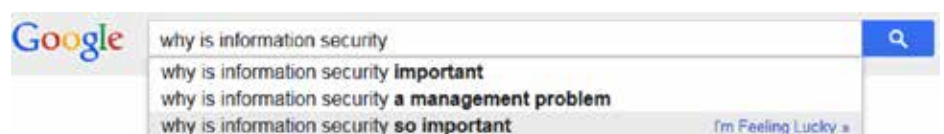
We saw human nature at work—particularly the parts of human nature easily bored when the same thing (or nearly the same thing) happens repeatedly, as well as the parts that like looking at unclothed people. Our comparison of four high-profile breaches (Target, Home Depot, Goodwill, and the theft of certain celebrity photos from Apple’s iCloud service) indicated that the photo scandal utterly dwarfed the others in public interest. More interestingly, of the other three breaches, Target (chronologically the first of the four) garnered the most attention, even though each of the remaining two were similar in either size (Home Depot) or demographic (Goodwill). Discussion of Target during the 2014 holiday season—a full year after the initial attack—far outstripped that of the other breaches. We expected to see that Target had raised consciousness about breaches; instead, a sort of burnout appeared to take place, with press paying less attention to subsequent events but looping back near the anniversary of the original breach to reflect.

[Editors’ note: As noted, our data was gathered and analyzed during the first eleven months of 2014. Ironically, at the time we were putting the Report together for publication, the Sony breach dominated not only tech but entertainment and political headlines. We have no doubt that with all that going on it would have posted some impressive numbers, but we concluded that far too much was in motion to provide a fair assessment of its impact for this Report.]

Despite the strong showing of *malware* and related terms, we found that the Internet as a whole took more interest in specific breaches than in specific vulnerabilities. Heartbleed, the most-referenced vulnerability of the year by several orders of magnitude, barely garnered the level of interest attracted by a moderately attention-getting breach such as that of JPMorgan Chase, and nothing like that of a Home Depot or a Target. In turn, Target at its most interesting was put in the shade by the celebrity-photo story. We did note that the photo story caused interest in celebrity photos themselves to spike, causing references to such things to spike by about a third.

What can security practitioners learn from this exercise? Where might one go with a Big Data-fueled analysis of security trends? One obvious path would be to deep-dive in tech-support threads and other venues where bugs are described, in search of reports that are not just bugs but probable security holes. At the moment, such forums can be useful reading to canny researchers, but the signal-to-noise ratio is poor; introducing efficiencies into sifting that data could be fruitful and might help companies with popular software to spot potential trouble before it spots them. Taking a more proactive tack, robust data analysis is already a powerful tool in the hunt to sift actual attacks from the avalanche of noise the average network’s parameter defense “hear” every day. As the security industry waits for automated security data exchange platforms to truly come to life, data analysis can provide us what those not-yet-viable systems cannot.

On the other end of the complexity spectrum, as we considered the possibilities for this Risk Report, one of our colleagues noted with disgust that some journalists seem to treat Google’s search-autocomplete function as some sort of Big Data-driven hivemind oracle. However, what makes for lazy journalism can provide an excellent reminder of the foundational questions at the base of security practitioners’ work:





Indeed. As we present our analyses of the threat landscape throughout this Report, we are reminded that what we examine, decide, and do is important. And a management problem. And, truly, so important.

Threat actors

2014 saw a shift in how technology was used in local and regional uprisings. Though hacktivism can be said to have declined—prompted by a decrease in anonymous activity following several high-profile arrests³—we saw an increase in the malicious use of technology both in and against protests. Attackers, reportedly from China, used remote access Trojans (RATs) masquerading as custom Android apps against protesters in Hong Kong.⁴ Attackers also reportedly intercepted Apple iCloud traffic to collect usernames and passwords.⁵ Elsewhere, the TOR network was hacked by unknown entities and its users were identified.⁶ As we closed the editing cycle for this Report, a massive data breach involving Sony Pictures Entertainment captivated world attention, though the provenance of that attack was unclear at press time.⁷

Attacks originating from groups based in China continued to target Western interests. Although historically these groups have focused on intellectual property theft, we observed a change in targets this year to focus on identity information as well. One high-profile example involved Community Health Systems, which disclosed a breach allegedly by a China-based group known as APT 18. In that breach, the Social Security numbers and other personal information of 4.5 million patients was compromised.⁸ This was the largest loss of patient data since the U.S. Department of Health and Human Services began keeping records of breaches in 2009.⁹ Adversaries acted quickly when observed: Mandiant reported that APT1, on which it had published an initial report one year before, immediately abandoned the command-and-control (C2) structure described in that report and set up a new one.⁹

2014 saw an increased response to this type of attacker group. In May 2014 the U.S.

Justice Department charged five officers in Unit 61398 of the Third Department of the Chinese People's Liberation Army (PLA) with hacking into U.S. entities for the purpose of intellectual property theft.¹⁰ In October, Novetta published reports on a cyber-espionage interdiction operation (referred to as Operation SMN), in which Novetta worked with U.S. security partners to take down 43,000 installations of tools used by a group called Axiom. It identified similarities in attacks seen as far back as Operation Aurora that could be attributed to this group. Evidence suggests that this group targeted organizations in China in addition to those in the West.¹¹

International law enforcement agencies increasingly worked together as well. In May Europol and the FBI conducted raids targeting users of the Blackshades RAT.¹² ¹³ The same month, an international effort identified the leader of a group responsible for the notorious Gameover Zeus botnet and CryptoLocker, leading to the dismantling of those networks.¹⁴ In November, agencies from 16 European countries, along with representatives from the United States, took down over 400 hidden services on the dark Web, including many carding and illegal drug markets.¹⁵

Nation-state supported activity

In 2014, we examined the state-sponsored or state-condoned cyber activity of actors in nations including Iran and North Korea. Among those nations we found three different levels of state involvement in cyber activity: indirect operational involvement, direct operational involvement, and condoning with plausible deniability of operational involvement. The degree of apparent state involvement was derived based on several factors, including:

- Evidence of state sponsorship of actor training
- The nation's cyber warfare infrastructure, capabilities, or doctrine
- The nation's cyber laws
- Threat actor group ties to government or military entities

³ wired.com/2014/06/anonymous-sabu/.

⁴ lagoon.com/chinese-government-targets-hong-kong-protesters-android-mrat-spyware/.

⁵ zh.greatfire.org/blog/2014/oct/china-collecting-apple-icloud-data-attack-coincides-launch-new-iphone.

⁶ blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack.

⁷ nbcnews.com/storyline/sony-hack/north-korea-behind-sony-hack-if-so-it-had-help-n271341.

⁸ reuters.com/article/2014/08/18/us-community-health-cybersecurity-idUSKBN0G16N20140818.

⁹ rsaconference.com/events/us14/agenda/sessions/1342/state-of-the-hack-one-year-after-the-apt1-report.

¹⁰ justice.gov/opa/pr/us-charges-five-chinese-military-hackers-cyber-espionage-against-us-corporations-and-lab.

¹¹ novetta.com/files/9714/1446/8199/Executive_Summary-Final_1.pdf.

¹² europol.europa.eu/content/worldwide-operation-against-cybercriminals.

¹³ fbi.gov/news/stories/2014/may/international-blackshades-malware-takedown/international-blackshades-malware-takedown.

¹⁴ fbi.gov/news/stories/2014/june/gameover-zeus-botnet-disrupted.

¹⁵ europol.europa.eu/content/global-action-against-dark-markets-tor-network.



Iran

In HPE Security Briefing Episode 11,¹⁶ we presented our findings on threat actors operating within the Islamic Republic of Iran. Iran's cyber doctrine pivots on the belief that "The cyber arena is actually the arena of the Hidden Imam"¹⁷ and relies heavily on warfare tactics.¹⁸ In November of 2010, Iran's Passive Civil Defense Organization announced a plan to recruit hackers for a "soft war" in cyberspace.¹⁹ On February 12, 2014, the Ayatollah Ali Khamenei delivered a message to the Islamic Association of Independent University Students, instructing them to prepare for cyber war:

"You are the cyber-war agents and such a war requires Ammar-like insight and Malik Ashtar-like resistance; get yourselves ready for such war wholeheartedly."

The Ayatollah stressed that this was the students' religious and nationalistic duty.²⁰ As noted in the report, Iran's cyber landscape has changed significantly from 2010 to the present. There was a noticeable transition from Iran's increasing awareness of cyber intrusions to the regime's institution of defensive cyber capabilities. The focus then shifted to implementing strategic offensive cyber capabilities. From the discovery of Stuxnet to the creation of a vast cyber army, Iran has made significant developments in the cyber war arena in a relatively short time.²¹

Our security research uncovered the following factors implying Iran's indirect operational involvement in the activities of the Iranian cyber underground:

- Threat actor group Shabgard's training portal at Webamooz.ir offered accredited IT training in conjunction with Shahid Beheshti University.²²
- Threat actor group Ashiyane offered training in conjunction with the Sharif University

- IT center.²³
- According to the Iranian Republic News Agency, Ashiyane's leader, Behrouz Kamalian,
- ordered the group to work for the Iranian government by attacking foreign government and media websites.²⁴
- Behrouz Kamalian's father, Hossein Kamalian, has served as the Iranian ambassador to Thailand, Laos, Myanmar, Bahrain, France, and Yemen.
- The European Union exposed Behrouz Kamalian's involvement in human rights violations—namely his involvement assisting the regime with cracking down on protesters during the 2009 political unrest in Iran.²⁵
- The EU report also linked Ashiyane to Iran's Revolutionary Guard.²⁶
- A report from Israel's Institute for Counterterrorism notes that it has been alleged that Ashiyane is responsible for training Iran's Cyber Army (ICR).²⁷
- Despite Iran's strict laws regulating Internet access and content, Ashiyane members do not fear being held accountable for their actions.²⁸
- Some of the threat actor groups profiled in the report use gamification as a training mechanism, including capture the flag (CTF) contests sponsored by Sharif University²⁹ and the Atomic Energy Organization of Iran (AEOI).³⁰

It is interesting to note that HPSR Security Briefing Episode 11 had a significant impact on some of the threat actors profiled in the report. After nearly 11 years of activity, the website and forums for Shabgard are now defunct.³¹

¹⁶ [h30499.www3.hp.com/hp/Security-Research-Blog/HPSR-Threat-Intelligence-Briefing-Episode-11/ba-p/6385243](https://www3.hp.com/hp/Security-Research-Blog/HPSR-Threat-Intelligence-Briefing-Episode-11/ba-p/6385243).

¹⁷ memri.org/report/en/print7371.htm.

¹⁸ inss.org.il/index.aspx?id=4538&articleid=5203.

¹⁹ forbes.com/sites/jeffreycarr/2011/01/12/irans-paramilitary-militia-is-recruiting-hackers/.

²⁰ haaretz.com/mobile/1.574043.

²¹ [h30499.www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf](https://www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf).

²² webamooz.ir/home/%D9%85%D8%AF%D8%B1%D8%B3%DB%8C%D9%86-2/.

²³ [h30499.www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf](https://www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf).

²⁴ defenddemocracy.org/behrouz-kamalian.

²⁵ ipost.com/International/EU-to-discuss-sanctions-against-over-80-Iranian-officials.

²⁶ eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2011D0235:20130312:EN:PDF.

²⁷ ict.org.il/LinkClick.aspx?fileticket=p02YcWmn_94%3D&tabid=492.

²⁸ [h30499.www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf](https://www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/177/1/Companion%20to%20HPSR%20Threat%20Intelligence%20Briefing%20Episode%2011%20Final.pdf).

²⁹ iran cybernews.org/2013/10/sharif-university-ctf-online-qualifier.html.

³⁰ ctf.aeoi.org.ir.

³¹ shabgard.org.



North Korea

In HPE Security Briefing Episode 16,³² we focused on the enigma that is North Korea's cyber threat landscape. North Korea's cyber warfare doctrine has not been clearly stated. However, based on cultural and technical observations, we may deduce that North Korea's cyber doctrine follows the tenets of *juche* nationalism and the *songun* doctrine. North Korea considers its cyber warfare capabilities an important asymmetric asset in the face of its perceived enemies.³³ In November 2013, Kim Jong Un referred to cyber warfare capabilities as a "magic weapon" in conjunction with nuclear weapons and missiles.³⁴

The North Korean regime plays a direct role in training its cyber warfare operators via primary and secondary education and the university system.³⁵ Successful students in the cyber warrior track often attend Kim Il-sung University, Kim Chaek University of Technology,³⁶ or the Command Automation University. Some students attend a two-year accelerated university program, then study abroad before they are assigned to a cyber-operator role.³⁷

Our research led to the conclusion that any activity originating from North Korea's IP space is a product of the state's direct operational involvement for the following reasons:

- North Korea's cyber infrastructure is divided into two major parts: an outward-facing Internet connection and a regime-controlled intranet.

- North Korea's outward-facing Internet connection is only available to select individuals, and is closely monitored for any activity that is deemed anti-regime.
- The North Korean regime strictly controls all Internet infrastructure,³⁸ meaning cyber activity by dissidents or autonomous hacker groups is very unlikely.
- The fact that North Korea reportedly spends so much of its limited resources on training and equipping cyber operators implies the regime is investing in a key military asset.

Additionally, we discovered that much of the computer network operations (CNO) conducted on behalf of the regime originates from the networks of third parties such as China, the United States, South Asia, Europe, and even South Korea.³⁹ When these networks are used to launch CNO on behalf of North Korea, attribution is difficult.

The Sony attack in late November was, at press time, not firmly linked to North Korea. Despite the fact that the regime denies responsibility for the attacks,⁴⁰ several factors seem to support that North Korea played a role in them. However, based on our previous research of North Korean cyber capabilities, it is difficult to discern whether the regime acted alone. It is plausible as of press time that the actors responsible for this attack relied on the assistance of an insider.

³² h30499.www3.hp.com/t5/HP-Security-Research-Blog/HP-Security-Briefing-episode-16-Profiling-an-enigma-North-Korea/ba-p/6588592.

³³ h30499.www3.hp.com/hpeb/attachments/hpeb/off-by-on-software-security-blog/388/2/HPSR%20SecurityBriefing_Episode16_NorthKorea.pdf.

³⁴ english.chosun.com/site/data/html_dir/2013/11/05/2013110501790.html.

³⁵ aljazeera.com/indepth/features/2011/06/201162081543573839.html.

³⁶ aljazeera.com/indepth/features/2011/06/201162081543573839.html.

³⁷ aljazeera.com/indepth/features/2011/06/201162081543573839.html.

³⁸ defense.gov/pubs/North_Korea_Military_Power_Report_2013-2014.pdf.

³⁹ [csmointer.com/World/Security-Watch/2013/1019/in-cyberarms-race-North-Korea-emerging-as-a-power-not-a-pushover/\(page\)/5](http://csmointer.com/World/Security-Watch/2013/1019/in-cyberarms-race-North-Korea-emerging-as-a-power-not-a-pushover/(page)/5).

⁴⁰ voanews.com/content/exclusive-north-korea-denies-involvement-in-cyber-attack-sony-pictures/2545372.html.

Notes on the threat modeling process

In May's HPE Security Briefing Episode 13—the first after the name of the series changed from the earlier "Threat Briefing," to reflect HPSR's growing scope of inquiry – we took a step back from the work of doing threat modeling and looked at the history and theories of the practice.⁴¹

At its base, threat modeling is yet another permutation of risk management, the soul of information security.⁴² Threat modeling asks that we assign value to our assets, examine them closely for potential vulnerabilities, assess what risks those vulnerabilities pose to our enterprise, and plan to mitigate them (or not). Threat modeling is not auditing—though auditing can be useful as we determine which assets or controls merit the modeling effort—but a way of learning from the past to manage future risk. Above all, threat modeling is, as most things are in the enterprise, a method of resource allocation.⁴³

There are three main approaches to threat modeling: software-centric, asset-centric, and attacker-centric. *Software-centric* threat modeling looks at how applications are built and how data flows through them. It tries to catch bad behaviors during the design and implementation stages—that is, before the application is released. Software vendors that follow a Software Development Lifecycle⁴⁴ process, or SDL, usually have multiple checkpoints to catch as many potential issues as possible. Such modeling doesn't catch every problem, but it cuts down on the "low-hanging fruit" and makes it obvious to developers where problems may lie.⁴⁵

Asset-centric threat modeling looks at what has value to an organization, or to its people or processes. Those valuations are a guide to appropriate protections.⁴⁶ The process often includes a look at what could happen to the assets, and the effect if something were

to happen to them. Outcomes are usually oriented toward appropriate controls and budget priorities. This kind of modeling is an effective way to approach highly regulated data or industries.

Attacker-centric threat modeling draws on experiences and theories about attacks on infrastructure, or against certain types of data or entities.⁴⁷ Its goal is to predict how anticipated attacks might progress, and how to deal with them if they do. This type of model may be appropriate when assets are less important than the motivation of adversaries towards an organization or state, or when a pattern of intrusions is repeated against multiple targets.

The analyses we do as we examine nation-state activity draw on multiple forms of information-gathering, including work done in partnership with HPE's Digital Investigation Services group and, when appropriate, independent third-party researchers and even competitors in the information-security field. The growing acceptance of prudent information-sharing is both a response to current conditions in the world and a hopeful sign that concerned enterprises and other entities can find a productive and effective balance between healthy competition and the workings of a healthy Internet.

We should note, by the way, that not all threat models attempt to identify who an attacker might be. Some models care about attribution, and some do not. All attacker-centric models, however, try to map controls to each phase of an attack.⁴⁸ This details the actions or tools that might be appropriate at each stage of the conflict. Understanding how individual attacks fit into the chosen schema can help enterprises prioritize spending and deployment of resources.

⁴¹ h30499.www3.hp.com/t5/HP-Security-Research-Blog/HP-Security-Briefing-episode-13-The-art-and-near-science-of/ba-p/64493964#VR2EofnF8nM

⁴² ssd.eff.org/en/module/introduction-threat-modeling

⁴³ Chen, Yue. "Software Security Economics and Threat Modeling Based on Attack Path Analysis; A Stakeholder Value Driven Approach" Retrieved from sunset.usc.edu/csse/TECHRPTS/PhD_Dissertations/files/ChenY_Dissertation.pdf

⁴⁴ microsoft.com/en-us/SDL/about/benefits.aspx

⁴⁵ Jangam, Ebenezer. "Threat Modeling and Its Usage in Mitigating Security Threats in an Application." Retrieved from isea.nitk.ac.in/publications/ThreatModeling.pdf

⁴⁶ Kayem, Anne. "Attack Centric Threat Modelling Approach to Social Networks: Hacking and Counter Measures." Retrieved from people.cs.uct.ac.za/~rratshidaho/Project/documents/Lit_Maoyi.pdf

⁴⁷ noord-group.com/process-attack-simulation-and-threat-analysis-pasta-risk-centric-threat-modeling

⁴⁸ Ramkumar Chinchani, Anusha Iyer, Hung Ngo, and Shambhu Upadhyaya. "A Target-Centric Formal Model For Insider Threat and More" Retrieved from cse.buffalo.edu/tech-reports/2004-16.pdf



Recent credit-card information breaches occurred in point-of-sale (POS) systems used by retailers, which proved to be a **target-rich** environment for many attackers.

The cyber underground

Activity in the cyber underground primarily consists of cyber crime involving identity theft and other crimes that can be easily monetized. There have been other motivations observed though, as was the case in August when suspected Russian hackers stole gigabytes of customer data from U.S.-based JPMorgan Chase during a period of escalated tension between Russia and the United States, in which Russia was amassing troops on the border with Ukraine and the United States had imposed economic sanctions against Russia in response. Reports stated that the FBI believed the attackers to be state-sponsored.⁴⁹

In September, F-Secure published research on “Quedagh,” a Russian cyber crime organization, and its use of BlackEnergy malware to target Ukrainian government organizations to steal information.⁵⁰ In its research it found evidence to suggest that this group could have been operating during the 2008 Russo-Georgian war, in which cyber operations paralleled military offensives.

In October, FireEye published research showing likely Russian state sponsorship of attacks targeting various organizations, with a focus on Georgia and the Caucasus, and primarily using spear-phishing techniques with embedded malware.⁵¹ The research indicated that this group may have been acting as far back as 2004.

However, one of the biggest stories this year was the ongoing spate of credit-card information breaches targeting U.S. retailers, including Staples, Kmart, Dairy Queen, Jimmy John’s, Home Depot, PF Chang’s, Goodwill, Sally Beauty Supply, Michaels, and Neiman Marcus. This followed the major Target breach during the 2013 holiday season. Many of the cards stolen from these stores ended up on carding forums being sold in groups by Russian actors.⁵² These breaches occurred in point-of-sale (POS) systems used by retailers, which proved to be a target-rich environment for many attackers. More detailed discussion is found [later in this report](#).

Conclusion

There’s the Internet that we see and the Internet that most of us don’t, and even though it is mostly invisible, the darker side of the Internet is pervasive and influential. Our investigations certainly suggest that the machinations and maneuvers of criminals and state-sponsored cyber operators in the cyber underground have significant and lasting effects on the security of the greater Internet and society at large. Looking into nation-state-sponsored cyber activity highlights the many levels at which cyber operations and state-sanctioned activity can occur, and how malware and the tools and techniques of cyber criminals can be utilized in different ways to accomplish different goals. (The same techniques nation states might use to stifle protest or target opposing state interests can just as easily be used by criminals to perpetrate fraud or steal intellectual property.)

Of most concern to enterprises, intellectual property continues to be targeted.⁵³ In 2014, responses to this long-recognized threat, and international cooperation to address these attacks, improved and continued to gain momentum. Cyber crime comes in many flavors but it remains vastly driven by financial interests. We expect attackers to continue to focus on intellectual property, identity data, and card information. While systems such as “chip and pin” are likely to prove more resistant to breach as particular points in financial processes get hardened, other points become more attractive to attackers. In a similar vein, as technology develops to improve the security of systems, it also conversely develops to make particular targets increasingly accessible. We expect escalations in this area to continue.

⁴⁹ bloomberg.com/news/2014-08-28/russian-hackers-said-to-loot-gigabytes-of-big-bank-data.html.

⁵⁰ f-secure.com/documents/996508/1030745/blackenergy_whitepaper.pdf.

⁵¹ fireeye.com/blog/technical/2014/10/apt28-a-window-into-russias-cyber-espionage-operations.html.

⁵² krebsonsecurity.com/2013/12/whos-selling-credit-cards-from-target/.

⁵³ ipcommission.org/report/ip_commission_report_052213.pdf.



Vulnerabilities and exploits

In the past year, significant shifts occurred in how researchers go about finding weaknesses in an enterprise's attack surface. The hunt is on for vulnerabilities in foundational technologies on which corporations rely to provide core business functionality. Multiple times in the last year, high-profile vulnerabilities were discovered that left enterprises scrambling to deploy patches and clean up compromised machines.

Even with this shift in focus, our adversaries are still leveraging classic avenues for attack. Exploitation of widely deployed client-side and server-side applications are still commonplace. Browser-based use-after-free (UAF) vulnerabilities are the vector of choice when attacking enterprises and government agencies. Corporations are also unaware of the risk imposed on them by using poorly coded middleware applications. Trends in submissions to the HPE Zero Day Initiative highlight the dynamic nature of the attack surfaces to which enterprises are exposed on a daily basis. Having insight into these trends will help users better prepare for the evolving nature of threats.

Exposing weaknesses in OpenSSL

Watching the industry respond to the Heartbleed vulnerability highlighted how unprepared we were for this type of disclosure. The flaw allowed for an unauthenticated remote attacker to disclose the memory of an application that uses the vulnerable version of OpenSSL. Successful attacks could result in the disclosure of SSL private keys, username/password combinations, and session tokens. Exploitation of this flaw typically would not leave any signs of abnormal behavior in the application's logs. It was a silent but serious threat to secure communication on the Internet.

This vulnerability also changed how the industry responded to vulnerability disclosures. Due to the severity and active exploitation of the vulnerability, corporations were forced to respond quickly, patching servers that were not routinely patched. The issue existed in an application library that did not have a clear update path. Enterprises did not have a solid understanding of which applications were using this library and where it was located inside their networks. Large software companies such as Microsoft have patching schedules so IT staff (users) can plan for the rollout of an update. In this case, network administrators were left to hunt all applications using the vulnerable version of the library and then manually apply the patch. Many organizations could not deploy the patches fast enough and struggled to defend against incoming attacks.



This activity rekindled the conversation around the security offered by open-source projects and the lack of financial support provided to the projects used in critical infrastructure. Many entities that rely heavily on OpenSSL to work correctly began donating financial support to the project. Meanwhile, researchers were upping their efforts to review OpenSSL source code to find additional vulnerabilities. It didn't take long for another critical OpenSSL vulnerability to show up in the queues at the Zero Day Initiative. Jüri Aedla is credited for the original discovery of this vulnerability.

The issue exists wholly within `ssl/d1_both.c` and occurs when handling Datagram Transport Layer Security (DTLS) fragments. DTLS has a fragmentation mechanism to break up large messages for UDP. Each fragment contains a 3-byte length field, which should be the same for all fragments in a message. OpenSSL incorrectly assumes that all DTLS fragments specify the same message size. Specifically, it trusts that the message length specified within the header of the first fragment will be invariant across all fragments.

Another significant observation is that the Wireshark protocol decoder highlights the mismatch of the length values in the DTLS fragments as a protocol error. Unfortunately, OpenSSL did not recognize this as an error condition.

Just sending this single UDP packet results in the application segfaulting and causing a denial-of-service condition, but more malicious actions are possible. An attacker could leverage this issue to corrupt adjacent metadata, and possibly execute code in the context of the process using OpenSSL.

The OpenSSL code does some sanity checking on the length fields in the DTLS fragments but, unfortunately, the check occurs too late and could be bypassed. The developers even left a prophetic comment in the code about what would happen if the validation failed.

This vulnerability is interesting from a development perspective. According to the commit logs, Robin Seggelmann introduced this vulnerability into the OpenSSL code base four years ago. Robin Seggelmann

is also responsible for introducing the Heartbleed vulnerability. Seggelmann is not completely to blame, of course. OpenSSL is an open source project. The “many eyes” that look at this code failed to catch this bug prior to 2014, but a new breed of individuals are looking at this code. This code is now known for having vulnerabilities and white-hat researchers are now focusing their efforts on auditing and securing this critical infrastructure.

Value of information disclosure

Discovery of information disclosure vulnerabilities such as Heartbleed is highly valued by the exploitation community. These issues can also be used in conjunction with remote code execution vulnerabilities to bypass modern exploit mitigations. For example, Microsoft® Internet Explorer® relies heavily on a mitigation technology called Address Space Layout Randomization (ASLR) to increase the complexity of exploitation of a vulnerability existing in the browser.

ASLR randomizes the base address of loaded DLLs. In the past, attackers relied on known addresses in DLLs to craft exploits. With the introduction of ASLR, attackers must either find a way to load a non-ASLR'd DLL or try to leak a DLL address. Using information disclosure vulnerabilities, attackers can render this mitigation useless by cherry-picking pointers within the leaked data, allowing them to learn the base address of the randomized DLLs.

Heartbleed was a nice demonstration of a highly controllable information disclosure vulnerability due to a buffer over-read, but these types of issues can also occur due to race conditions in an application. In April, the HPE Zero Day Initiative received an interesting vulnerability in Apache `httpd mod_status` from several Polish researchers. The root cause of the vulnerability was a race condition between the updating of `httpd's "scoreboard"` and `mod_status`, leading to a heap overflow with attacker-supplied data. ZDI concluded it was possible, with a well-crafted exploit, to disclose application memory containing internal server configuration details, `htaccess` credentials, and other application data.

Many corporations are unaware of the risk imposed on them by using **poorly coded** middleware and IT management applications.

Weaknesses in enterprise middleware

Corporations are embracing software as a service (SaaS) and other middleware solutions to shorten the time it takes to deliver business applications. These applications contain copious amounts of sensitive corporate data and personally identifiable information. Middleware applications rely heavily on protocols such as HTTP, Simple Object Access Protocol (SOAP), and JSON to communicate with each other. Most of these communication protocols are exposed to the network and are accessible without authentication. The attack surface exposed by these applications can be large and riddled with weaknesses.

These services have become an increasingly popular target for researchers and the number of vulnerabilities discovered in 2014 was astonishing. The HPE Zero Day Initiative worked with numerous middleware and IT management software vendors to shore up their code. In fact, during just one week, a single researcher submitted over 40 remotely exploitable vulnerabilities in ManageEngine's product line. These vulnerabilities ranged from information disclosure issues to denial of service conditions and remote code execution vulnerabilities.

To highlight the ease with which these issues could be exploited, this Report takes a deeper look at one of the resolved information disclosure issues. CVE-2014-8678 (ZDI-14-386⁵⁴) was a vulnerability in the ManageEngine OpUtils ConfigSaveServlet servlet. This vulnerability allowed remote attackers to disclose files on vulnerable installations of ManageEngine OpUtils. Authentication was not required to exploit this vulnerability.

The issue lies in the failure to properly sanitize the saveFile parameter for directory traversal characters. A remote attacker can exploit this vulnerability to disclose files from the system.

Using directory traversal, an attacker can easily disclose sensitive information residing on the server running ManageEngine OpUtils. The impact of this attack can be visualized further by understanding the type

of data handled by the OpUtils software. According to ManageEngine, OpUtils helps network engineers manage their switches and IP address space.⁵⁵ Specifically, OpUtils would have details about a corporation's IPv4 and IPv6 subnets, backups of configuration files of Cisco routers and switches, and bandwidth usage statistics. It's possible for an attacker to leverage the vulnerability to disclose this valuable information to aid them in future attacks. Many corporations are unaware of the risk imposed on them by using poorly coded middleware and IT management applications. Updates to these applications should be applied as soon as they are available to reduce exposure.

Vulnerability and exploits trends in 2014 (Windows case)

2014 saw Microsoft Internet Explorer, Microsoft Office and Adobe® Flash Player zero days in the wild. Notably in 2014 there were no major Oracle Java zero days discovered exploited in the wild. This is likely due in part to the click-to-play feature Oracle recently introduced. This section takes a deeper look at the security technologies and how they were bypassed.

Defeating ASLR and DEP security protections

Most of the exploits observed in the wild were successful at defeating ASLR and Data Execution Prevention (DEP). DEP, like ASLR, is a security feature. It marks areas of memory as either "executable" or "nonexecutable," allowing only data in an executable area to be run. DEP protects against some program errors and helps prevent certain malicious exploits. The ability to bypass these protections has become a common feature of modern exploits. While many different techniques may be used to defeat these protections, the most popular method is to corrupt application objects on the heap and change the length field of the object, as seen in the multitude of Microsoft Internet Explorer UAF exploits. Often surgical precision memory manipulation is performed resulting in a very high exploit success rate. Object corruption and code reuse attacks are typical techniques currently used to defeat ASLR and DEP.

⁵⁴ zerodayinitiative.com/advisories/ZDI-14-386/

⁵⁵ manageengine.com/products/oputils/ Page 15.

Object corruption

For example, the CVE-2014-1761 exploit in the wild corrupted an object created from GFX.dll. The method corrupted is related to a graphical object and is called constantly with a short interval of time between each call. This allows the attacker's code to execute immediately following the corruption of the vtable of the object.

Another example exploit used CVE-2014-0515 and involved corrupting multiple objects, including the vector object and FileReference object. Vector object exploitation uses a buffer overflow vulnerability resulting in an abnormally large field value length. This allows the attacker to access a vast range of memory freely from the script.

Some IE exploits also used a similar technique to defeat ASLR and DEP. The exploit code for CVE-2014-1776 used a maliciously constructed SWF file to load Flash Player-related modules and setup memory containing vector objects. The UAF vulnerability may be used to overwrite 2 bytes of memory at an arbitrary location. By corrupting the first 2 bytes of the vector object, which is used as a length field, the exploit will gain the ability to access broader read and write memory space through the vector object.

Code reuse attack

After acquiring out-of-bounds memory read/write access, the exploit would corrupt another object's vtable, reusing existing code fragments from loaded DLLs to defeat ASLR. For example, the exploit code for CVE-2014-0515 uses a fake FileReference function table to run code from the existing location of the Adobe Flash Player executable location.

With fake function table setup, the exploit code calls the related method, "cancel." This passes the execution to a location the attacker designates. This becomes interesting when the location to which the control flow is passed is inside the Adobe Flash Player executable itself.

The function calls other functions that in turn call VirtualProtect to add an executable bit to the designated memory region. This is a very sophisticated way of defeating DEP—not exactly return-oriented programming (ROP), but a type of code-reuse attack.

It is possible to use ROP to defeat DEP and is the more traditional approach. CVE-2014-1776 exploit code used ROP; after first acquiring read and write access to the full process memory area, it sets up the ROP code. When the ROP code is running, it adds an executable bit to the shellcode area that follows the ROP code.

Rise of legacy code vulnerabilities

Deprecated features

Vulnerabilities found in legacy code were also a significant factor in 2014. CVE-2014-0515 was a vulnerability in the Pixel Bender feature of Adobe Flash Player. The feature is officially deprecated, but the code remains in the executable. We noted that the DWORD value at offset 0xEA of Pixel Bender data was responsible for triggering the vulnerability.

The metadata of defaultValue is intended to be just 4 bytes long, but the code tries to convert all 16 arguments and put them in the memory array, incurring a memory corruption error. The original binary data is parsed and translated into a bytecode that is later used for further operations. The second value from the defaultValue argument overwrites the index value inside the SEL instruction's byte code in memory.

Based on the nature of the issue and the age of the code, we suspect that this vulnerability was found using dumb fuzzing that replaced random bytes in the target sample file. The bug itself appeared very old and was likely present from the earliest versions of the Pixel Bender feature; now, even though the feature is no longer supported, the flaw may still hurt the security of the products.



Implementation of mitigations in software such as Windows raises the bar on exploitation difficulty, and attackers respond with **sophisticated** attacks.

CVE-2014-1761 was a vulnerability in RTF parsing code, as well as a simple buffer overflow. The RTF parser has existed in Microsoft Office for decades, leading us to reasonably assume the bug has been there for just as long. The overflow bug class has long been hunted with stack overflow as the most well-known and easy to locate. Additionally, the code patched for this vulnerability is a known problem spot. The patch for CVE-2014-1761 was released with the MS14-017 security bulletin, but there was another vulnerability (CVE-2012-2593) in the same function two years ago and patched with the MS12-079 security bulletin. While the bug classes of the vulnerabilities are different, they involve the same RTF key word and are very similar in nature, using an edge case value for the key word. CVE-2014-4114, another legacy code vulnerability, exploited the OLE packager feature that has existed since Windows® 3.1.

The exploitation of vulnerabilities in legacy code is of significant concern from two angles. It's important to apply timely patches in the enterprise environment; however, it's just as important for vendors to invest time on legacy code testing and patching. Creating and implementing new managed languages and new security features on decades-old code base is not secure. While matters have improved with the help of the security community reporting legacy issues, expect to see these from time to time in the future.

Highly successful rate vulnerabilities

Oracle introduced click to play as a security measure making the execution of unsigned Java more difficult. As a result we did not encounter any serious Java zero days in the malware space. Many Java vulnerabilities were logical or permission-based issues with a nearly 100 percent success rate. In 2014, even without Java vulnerabilities, we still saw high success rate exploits in other areas.

Logical issue

CVE-2014-4114, found in the wild, was a logical issue bug involving the OLE packager component. When properly exploited, it was always successful. The bug involves the OLE object insertion feature in Office and enabled users to package a non-OLE object into a document.

In this case, what is included inside the Packager is a UNC path to an INF file. The document containing this OLE object would launch for the INF file automatically without a victim's knowledge. Use of the INF file allows for a number of dangerous operations.

By trusting an INF file from an untrusted source it opens a gap that an attacker can exploit to use an INF file to do various dangerous things like renaming files and launching programs.

Surgical precision exploits

Even memory-related vulnerabilities such as CVE-2014-0515, CVE-2014-1761, and CVE-2014-1776 showed high exploitation success rates. CVE-2014-0515, an Adobe Flash Player vulnerability, was used in an exploit with a heap-spray technique. By laying out memory such that the memory corruption changes the length field of one heap-spray element, it can achieve full memory read and write access to the process. Once this is achieved, the attacker has full power over the process itself.

The CVE-2014-1761 vulnerability was used in a way that changed the adjacent GFX object with surgical precision. When the exploit tries to allocate multiple array members, it can fully control the contents of the memory data. The controllability of the data, such that it overwrites a GFX object, is very important. In this case, every byte of the data is fully controllable through RTF key words. The attackers were sophisticated enough to figure out which bytes are controlled by which RTF key word.

Implementation of mitigations in software such as Windows raises the bar on exploitation difficulty, and attackers respond with sophisticated attacks. The time when exploits were dependent on luck with memory layout is nearly past. Today's exploits are highly calculated with memory layout and exploitation techniques. Many zero-day exploits that emerged in 2014 demonstrated a near-perfect success rate. There has been a decline in attackers using large heap sprays that take a long time and get the victim's attention.

Conclusion

Software vendors continue to make it more difficult for attackers with the implementation of security mitigations, but they aren't enough when they are built on decades-old code still inherently vulnerable. The one exception seems to be the success of Oracle's click-to-play mitigation in thwarting Java attacks. While it is more difficult for attackers to succeed, we are experiencing very high success rates with exploits in the wild, which may indicate they were authored by professional exploit developers with high exploit development skills. The quality of exploits is improving and sometimes reveals a deep understanding of the nature of the vulnerability and the internals of the target applications.

Malware and exploits

Year after year, exploits have been the main vector for a wide range of malware attacks. They serve as one of the early steps in achieving control over the target in a cyber-attack sequence.⁵⁶ Over the years we have seen hundreds of vulnerabilities exploited with different applications and operating systems being affected, ranging from Web browsers to multimedia apps and run-time environments such as Oracle Java.

Every year thousands of CVE numbers are issued for various vulnerabilities, but malicious actors are interested in the most serious class of vulnerabilities—the ones that allow the attacker to achieve remote code execution. HPE Security Research, together with ReversingLabs, has a catalog of more than 100,000 exploits collected over the course of the year. In this Report we display and discuss 2014's top trends.

Top CVE-2014 numbers collected in 2014

The most common CVE-2014 exploit discovered by our teams is CVE-2014-0322.⁵⁷ First reported by FireEye in February,⁵⁸ CVE-2014-0322 exploits a use-after-free vulnerability in Internet Explorer and commonly uses an Adobe Flash stage to bypass exploit mitigations in Windows to deliver its final executable payload. The exploit was first seen in Operation SnowMan, which allegedly targeted U.S. government entities and defense companies.

CVE-2014-6332,⁵⁹ also known as “Windows OLE Automation Array Remote Code Execution Vulnerability,” is another vulnerability that attracted a lot of attention in the security community, especially because the vulnerability has been present in various versions of Windows for over 18 years,⁶⁰ since the days of Windows 95. The exploit is delivered through VB Script, so it can only be delivered to Internet Explorer. It allows for an easy sandbox escape if combined with a routine that changes flags to disable Internet Explorer's Safe Mode.

In the wild, however, the exploit often uses a combination approach, similar to the delivery of CVE-2014-0322. The exploit is triggered by redimensioning an array to transfer control to Adobe Flash shellcode. This bypasses exploit mitigations, including older versions of Microsoft's Enhanced Mitigation Experience Toolkit (EMET). A specially crafted JPG image with an appended encrypted data buffer is loaded into memory space which, when decrypted by shellcode present in the SWF file, drops and runs the final executable payload.⁶¹

⁵⁶ lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf.

⁵⁷ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0322.

⁵⁸ fireeye.com/blog/threat-research/2014/02/operation-snowman-deputydog-actor-compromises-us-veterans-of-foreign-wars-website.html.

⁵⁹ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6332.

⁶⁰ securityintelligence.com/ibm-x-force-researcher-finds-significant-vulnerability-in-microsoft-windows/.

⁶¹ researchcenter.paloaltonetworks.com/2014/11/addressing-cve-2014-6332-swf-exploit/.

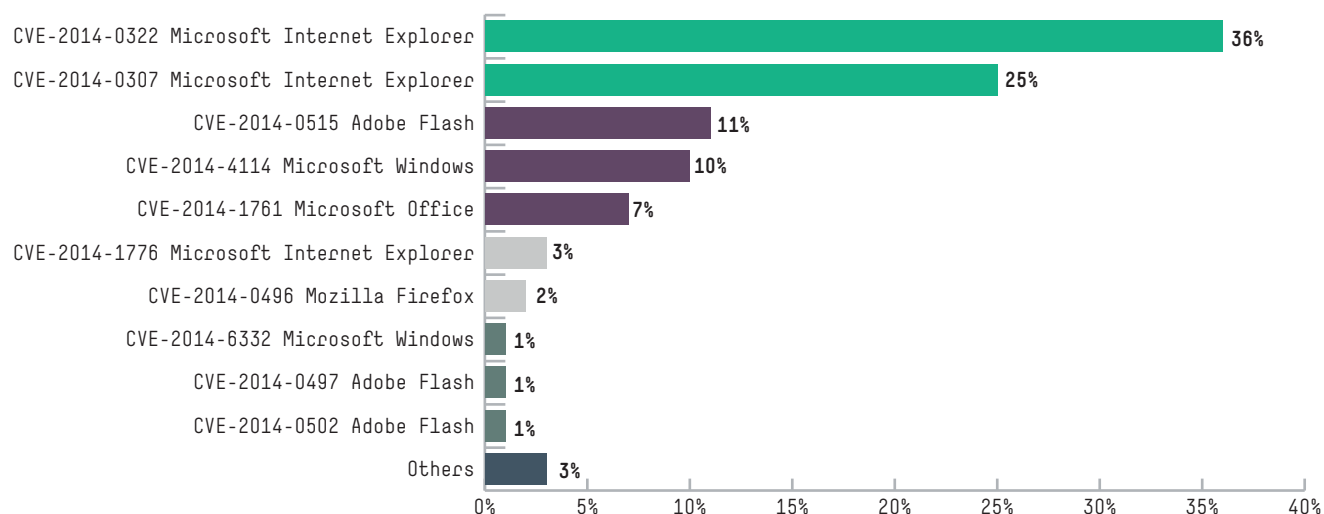


Figure 1. Top discovered CVE-2014 exploits

Top CVE-2014 for malware attacks

As discussed above, of the top 10 CVE-2014 exploits seen, none of them targets Java, one of the most commonly exploited targets in previous few years. This may indicate that the security push,⁶² which caused delay in the release of Java 8, is getting some results, although it may be too early to tell. It may also be a consequence of browser vendors blocking outdated Java plugins⁶³ by default, making the platform a less attractive target for attackers.

The breakdown of the top 10 discovered exploits over different applications is as follows. Four exploits are delivered through Internet Explorer; these four together account for almost two-thirds of all CVE-2014-based exploits discovered this year. Two Windows exploits are delivered using Microsoft Office files, three using Adobe Flash, and one through Adobe Reader.

In-depth analyses of CVE-2014-0505,⁶⁴ CVE-2014-1761,⁶⁵ CVE-2014-4114,⁶⁶ and CVE-2014-1776⁶⁷ have been published on the HPE Security Research blog over the course of the year.

⁶² threatpost.com/does-java-8-delay-mean-oracle-finally-serious-about-security/99908.

⁶³ blogs.msdn.com/b/ie/archive/2014/08/06/internet-explorer-begins-blocking-out-of-date-activex-controls.aspx.

⁶⁴ h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-0515-Adobe-Flash-Player-Exploit/ba-p/6482744.

⁶⁵ h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-1761-RTF-Vulnerability/ba-p/6440048.

⁶⁶ h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-analysis-of-the-SandWorm-Vulnerability-CVE-2014-4114/ba-p/6449758.

⁶⁷ h30499.www3.hp.com/t5/HP-Security-Research-Blog/The-mechanism-behind-Internet-Explorer-CVE-2014-1776-exploits/ba-p/6476220.

Top CVE numbers seen in 2014

Although we have seen over 30 CVE-2014 exploits used by malware, the majority of exploits discovered by our teams attempt to exploit older vulnerabilities. By far the most common exploit is CVE-2010-2568,⁶⁹ which roughly accounts for a third of all discovered

exploit samples. This vulnerability in shell32.dll allows the attacker to plant a specially crafted .PIF or .LNK file, which triggers the vulnerability when a user browses the content of the folder containing the malicious files. The exploit was used as one of the infection vectors for Stuxnet and quickly gained popularity in the world of malware writers.

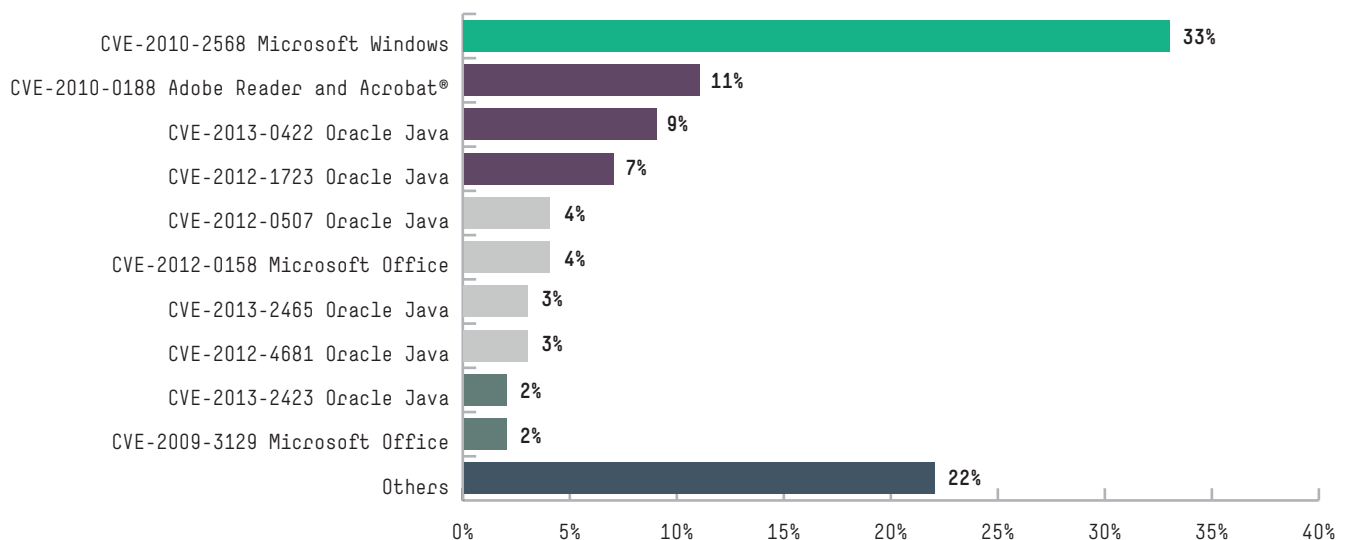


Figure 2. Top exploit samples in 2014; note CVE numbers, which are a useful guide to when the vulnerability was first reported

In fact, CVE-2010-2568 is the only exploit for which the number of discovered samples grew month over month throughout the year.

The breakdown of the top 10 overall exploit samples discovered this year is quite different compared to only CVE-2014 exploit samples. Oracle Java holds the top place in terms of numbers with six exploits in the top 10, accounting for 29 percent of all discovered samples, with CVE-2013-0422⁶⁹ being the most popular of Java exploits. These are followed by the already mentioned CVE-2010-2568 targeting Windows; CVE-2010-0188,⁷⁰ which targets Adobe Reader, accounting for 11 percent of samples; CVE-2012-0158⁷¹ targeting Microsoft Office with 4 percent of samples; and CVE-2009-3129⁷² targeting Microsoft Excel®, with less than 2 percent of all exploit samples discovered in 2014.

The discovered exploit samples indicate that there is still a significant percentage

of Windows users who do not regularly update their systems with security patches. This issue may have been exacerbated by Microsoft ending support for Windows XP security updates in April⁷³ for most users (and not counting the emergency MS14-021 patch released in late April).

Looking at the operating systems targeted by exploits, it is obvious that attackers are still concentrating on Windows, despite high-profile vulnerabilities in other technologies, such as CVE-2014-6271⁷⁴ (Shellshock), that were discovered in 2014.

The most common exploit encountered for non-Windows operating systems targeted CVE-2013-4787,⁷⁵ also known as the Android Master Key vulnerability. Samples targeting this vulnerability accounted for a little over one percent of all exploit samples.

⁶⁸ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2568.

⁶⁹ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0422.

⁷⁰ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0188.

⁷¹ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0158.

⁷² cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3129.

⁷³ microsoft.com/en-us/windows/enterprise/end-of-support.aspx.

⁷⁴ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271.

⁷⁵ cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4787.

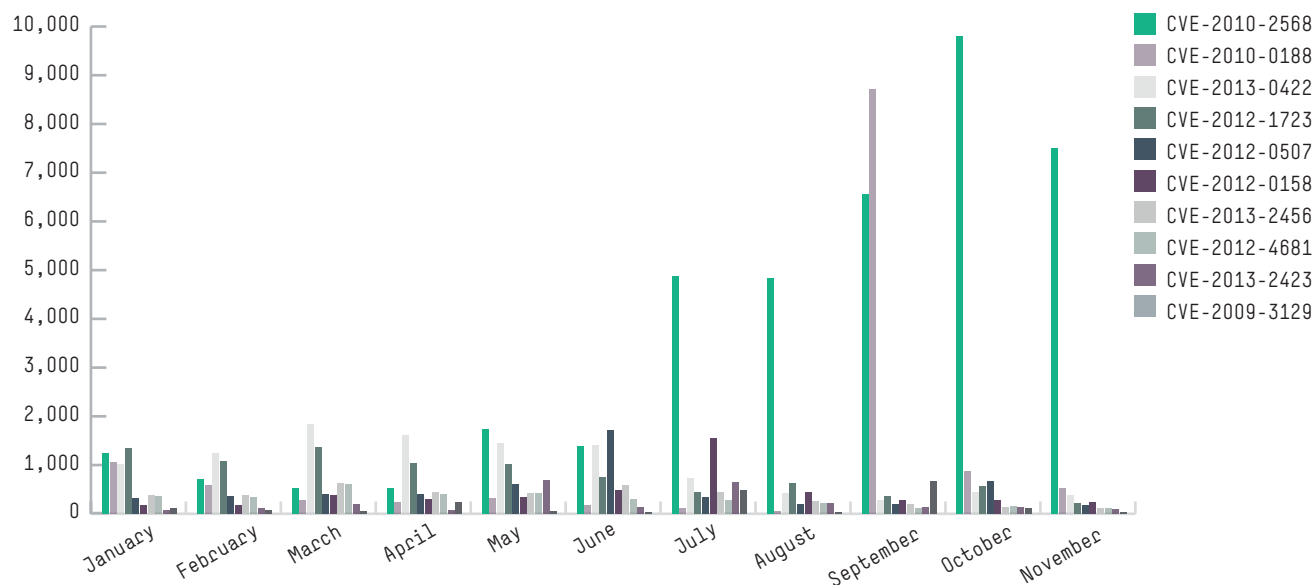


Figure 3. Monthly breakdown of the top 10 exploit samples discovered in 2014

Looking at the file types used to deliver exploits through Web browsing or email attachments, Java applets and class files are the most common and account for 48 percent of all samples of this set, followed

by PDF files, HTML (JavaScript), and Word (OLE2) documents. These account for 33 percent, 10 percent, and 5 percent of discovered malware samples respectively.

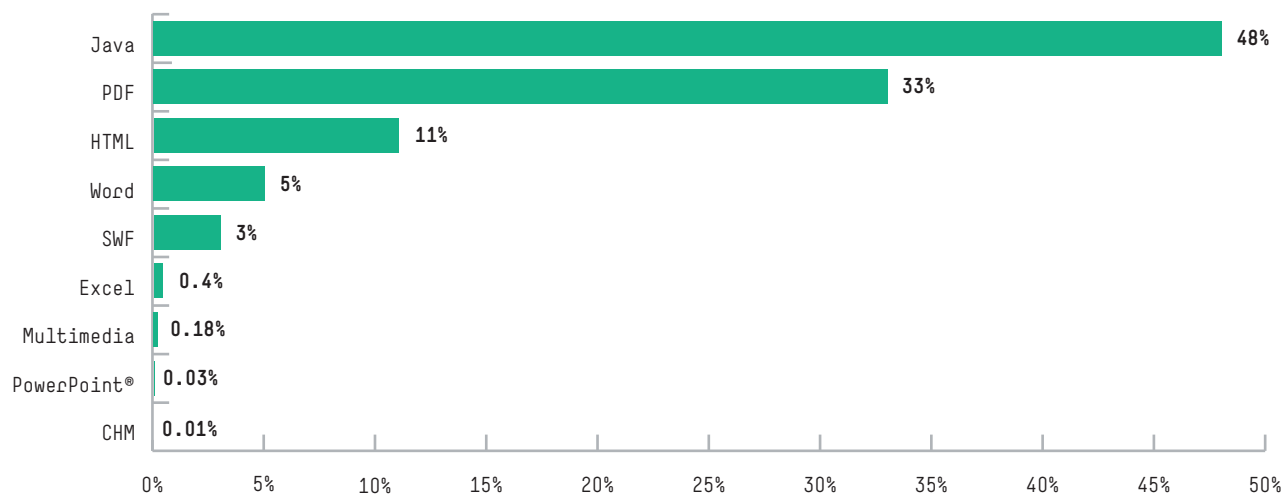


Figure 4. Web or email exploit samples by file type

Defenders are global

Zero Day Initiative's researchers: Geographic distribution

In its first decade, the HPE Zero Day Initiative has received over 7,000 submissions from 80 countries around the world. Taking a closer look at the data, an interesting perspective emerges on where in the world vulnerability research is occurring. The following list shows the countries with the highest submission rate since the program's inception:

1. United States
2. Canada

3. Italy
4. France
5. Poland

Over the past two years, several new hot spots popped up with high submission rates and quality technical analysis including Germany, South Korea, China, and the Russian Federation. Researchers in these countries are not only focusing on vulnerability discovery but also on innovative exploitation techniques. The coverage map below pinpoints the countries actively submitting unpatched vulnerabilities to the program.



Figure 5. ZDI researcher coverage map



Looking ahead, we will continue to see a focus on **browsers** and plugins.

Conclusion

Researchers and analysts in the HPE Zero Day Initiative were busy coordinating the disclosure and remediation of over 400 high-severity vulnerabilities in 2014. This year marks the highest number of disclosures in a single year. 2013 brought quite a few Oracle Java sandbox bypasses to the program. In 2014, however, researchers shifted to browser vulnerabilities, focusing most of their efforts on Microsoft Internet Explorer.

ZDI researchers tuned their browser fuzzers to discover dozens of UAF vulnerabilities. A use-after-free vulnerability can occur when memory is allocated to an object that is used after it is deleted (or deallocated). Good programming practice dictates that any reference pointing to an object should be modified when the memory is deallocated, to keep the pointer from continuing to make the area of memory where the object once resided available for use. (A pointer in this abandoned condition is broadly called a “dangling pointer.”) If the pointer isn’t modified and tries to access that area of memory, the system can become unstable or corrupt. Attackers can use a dereferenced pointer in a variety of ways, including execution of malicious code.

Examining 2014 submissions revealed a mix of “old” and “new” vendors at the top for most disclosures:

1. Microsoft
2. Hewlett Packard Enterprise
3. Advantech
4. SAP
5. Apple

In 2013 there were a number of SCADA vulnerabilities, but 2014 marks the first year where a SCADA vendor is among the top vendors with vulnerabilities disclosed against its products. Advantech focuses on automation controllers, industrial control products, and single board computers. SAP is on the list due to an audit ZDI analysts conducted against one of its products, which yielded a large number of findings.

Looking ahead, we will continue to see a focus on browsers and plugins that support them. The attack surface offered by the complex software is used heavily when targeting governments and high-profile organizations.

Threats

The end game of many attackers that exploit vulnerabilities is to install various types of malware. In 2014 the malware problem continued unabated, and while the anti-malware industry has introduced multiple new approaches to the issues it faces, the impact of those measures on the security of organizations and the public is questionable. Increasingly, anti-malware technologies rely on monitoring for particular behaviors rather than monitoring for the presence of particular files, and they harness Big Data and cloud capabilities in order to detect and address new malware families by aggregating multiple data points and dimensions. By utilizing these technologies, the ability to detect malicious files heuristically (that is, to identify malware not seen before based strictly on its characteristics) has improved—but nowhere near enough. The defenders are worried—are we winning the war against malware, or are we going to be swept away by the rising tide?

Windows malware overview

State of protection

Year after year, the number of newly created malware samples balloons. In 2013, AV-Test.

org, a reputable independent anti-malware testing organization, collected 83 million malware samples. For 2014 the final number is expected to be close to 140 million. If we simply extrapolate the numbers, we can be almost certain we will reach the 200 million mark in the coming year.

If we consider that 200 million number, we see that to reach it over the year, AV-Test—or any reputable anti-malware vendor—should be capable of processing an average of 600,000 samples every single day. The increasing number of samples poses great challenges for anti-malware engines, and the rates of detection for previously unknown malware instances are declining.

Our tests on standard scanning engines, conducted over a set of over 80 million samples in cooperation with ReversingLabs, show that detection of previously unknown samples at the moment of discovery significantly varies from vendor to vendor. This illustrates the need for complementary protection technologies that provide more dynamic protection. These technologies are usually built into most endpoint security products.

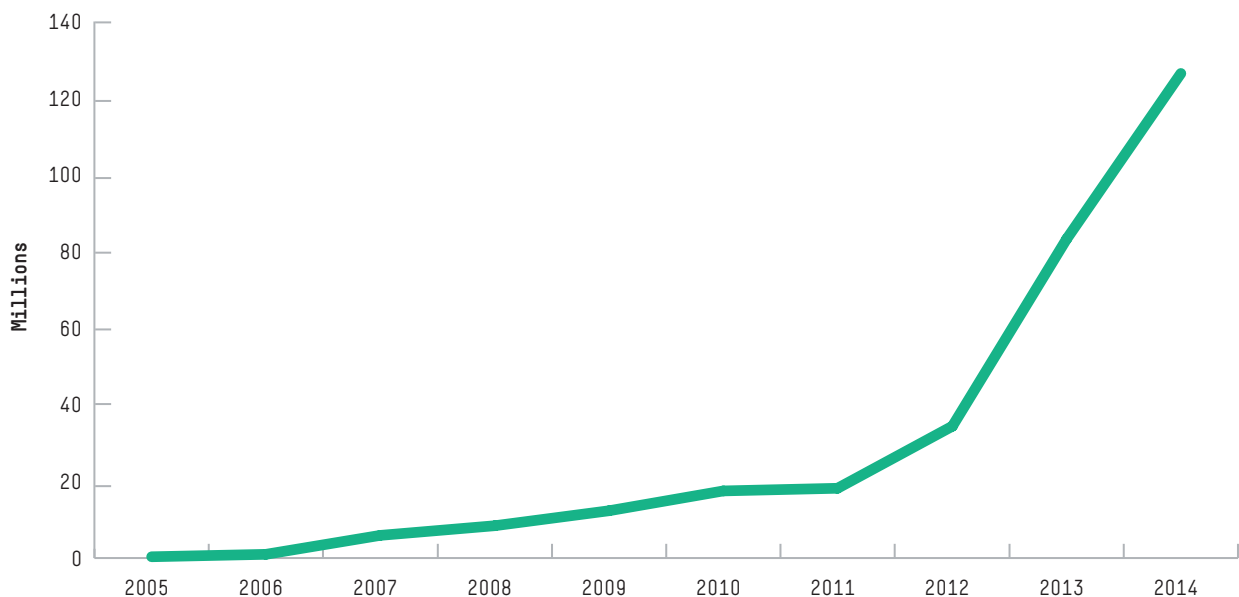


Figure 6. Unique malware samples collected by AV-Test

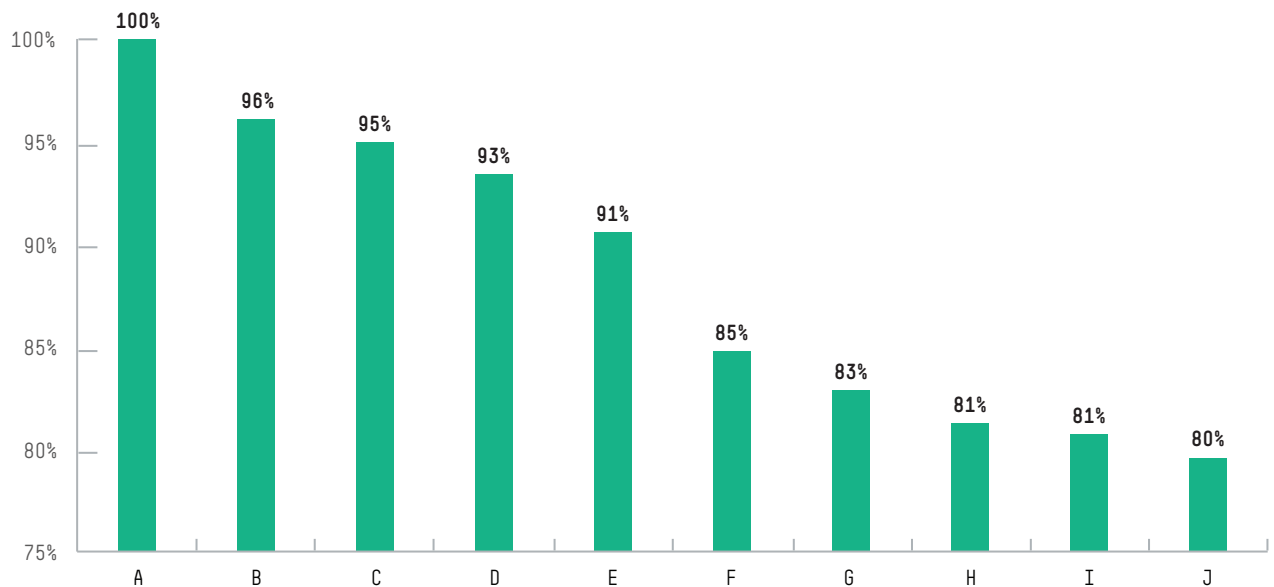


Figure 7. Various anti-malware vendors' detection rates on previously unknown samples [normalized relative to the best-performing engine]

Relatively low rates of change of detection for samples a week or longer after their discovery show that many malware threats are transient, with their initial distribution lifecycle lasting a day or less. In addition, this may indicate the inability of anti-malware vendors to process an ever-growing number of incoming samples.

The sheer volume of malware samples that appears every day plays into the hands of actors with sufficient funds to conduct highly skilled targeted attacks and evade all layers of traditional protection. Large organizations have recognized the need to build security operations centers (SOCs) with skilled staff able to recognize, respond to, and remediate attacks when they happen.

Unfortunately, the level of technical skill, experience, and knowledge required to address targeted attacks is high. There is a skill shortage, usually addressed by installing a combination of incident response software and systems (such as sandboxes) designed to detect whatever portion of the attacker's tools and malware managed to penetrate traditional layers of defense.

The focus for organizations is not just how to protect, but rather how to respond and remediate attacks—understanding with certainty that attacks will be successful if carefully planned and executed.

Top malware discovered

The top Windows malware discovered shows a slightly different view on previously unknown samples. In our practice we have gotten used to a way of counting unique binary files as single instances, which works well in the case of Trojans—that is, malware that is unable to replicate itself. Our data

shows that the most commonly encountered malware families are the ones that either have the ability to replicate and create a functionally identical copy (worms) or an ability to modify another executable to include its own functionality (parasitic viruses).

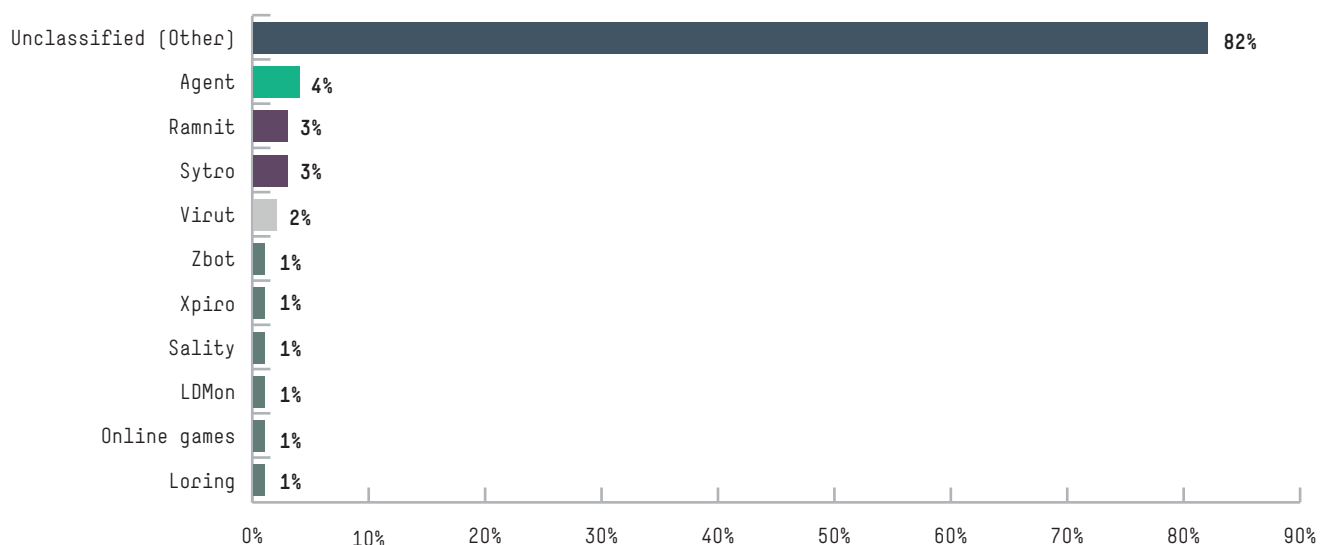


Figure 8. Top malware samples discovered by ReversingLabs in 2014, by family

Agent is a malware name that is used as a name space for all malware samples that cannot be easily classified.

By far the most commonly reported malware name is Agent. However, this name is not used for a single family, but rather as a name space for all malware samples that cannot be easily classified into any other known existing families. Again, this is directly related to the volume of malicious files that need to be processed. The ability to fully analyze all malware, recognize what it is, and determine what it does is often beyond the means of many AV companies. When a file is determined to be malicious, the pressure to detect large volumes of files means that once researchers know enough to add detection for the file, they move onto the next—only doing the minimum amount of security research necessary.

Most of the other top 10 collected malware has the ability to replicate: Virut, Sality, and Expiro are polymorphic infectors that have been present for many years and may be used for information stealing, while Ramnit is a worm designed to steal information such as online banking credentials. Onlinegames is a password stealing malware designed to steal the credentials of online games. In fact, it seems that the majority of the top 10 malware is geared toward stealing data as opposed to immediately obtaining financial benefits, as is the case with fake anti-malware and ransomware samples.



Notable malware

Ransomware

Although the concept of ransomware goes back to the days of the DOS operating system, it is only in the last couple of years that it has become a contender to fake or rogue anti-virus software in prevalence and the potential to cause damage to victims' data.

Perhaps the most notable ransomware is CryptoLocker, which appeared at the end of 2013 and caused a lot of damage to end users and organizations until the FBI's operation Tovar⁷⁶ disrupted its distribution channel and brought down a large Gameover Zeus botnet.

Nevertheless, the business model in which users' data is held for ransom by malware using asymmetric encryption algorithms to encrypt it has spawned a number of copycats, with CryptoWall⁷⁷ being the most well-known.

In addition to the ransomware that actually encrypts the data (so that the only way to recover is to restore it from unaffected backup media), another class of malware that simply locks user access to the operating system (e.g., Reveton⁷⁸) or to the Web browser (e.g., Krypterade⁷⁹) is also very prevalent but thankfully much easier to remove.

Ransomware threats are here to stay and organizations must have a sound backup and restore policy in place for all business data in order to mitigate the potentially destructive effects of a successful attack. Not much detail is known about individuals and organizations that resort to the last desperate step of paying attackers the money, nor whether the required data (or the private key required for decryption) is delivered to victims after the ransom money is paid. Judging by the prevalence of

ransomware threats, however, this cyber-criminal business model appears to be quite successful.

High-complexity malware

The best-known example of high-complexity malware is Stuxnet,⁸⁰ which was designed to attack industrial systems, particularly centrifuges used for the enrichment of uranium in the Iranian Natanz fuel enrichment plant. Malware such as Stuxnet poses a lot of questions for a malware researcher. Some questions are never answered, and some are answered only after lengthy, iterative research, where even the smallest clues are followed. In the case of Stuxnet, new details were revealed⁸¹ in 2014, almost four years after the malware was first discovered by the malware research community.

The year 2014 marked the discovery of another highly complex malware suspected to be developed by an organized and well-funded group of developers—Regin.⁸² Regin is a multi-component malware designed as a framework that allows for the creation of multiple plugins. Regin employs sophisticated hiding methods and encrypted virtual file systems, and was designed for the purpose of security intelligence gathering by continuously monitoring individuals and organizations. It may have been in use since 2008, but the first samples were discovered by Symantec and Kaspersky researchers as recently as 2013. The research shows that many components of Regin are not yet discovered and additional functionality and versions may exist in the wild.

Once again, this supports our conclusion that a skilled attacker will be able to penetrate all traditional levels of defense and maintain access to victim systems by choosing attack tools that will not show up on the radar of anti-malware and other protections.

⁷⁶ fbi.gov/news/pressrel/press-releases/u.s.-leads-multi-national-action-against-gameover-zeus-botnet-and-cryptolocker-ransomware-charges-botnet-administrator.

⁷⁷ symantec.com/security_response/writeup.jsp?docid=2014-061923-2824-99.

⁷⁸ krebsonsecurity.com/2012/08/inside-a-reveton-ransomware-operation/.

⁷⁹ microsoft.com/security/portal/threat/Encyclopedia/Entry.aspx?Name=Ransom.JS/Krypterade.A#tab=2.

⁸⁰ langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf.

⁸¹ securelist.com/analysis/publications/67483/stuxnet-zero-victims/.

⁸² symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf.

Other trends

It is worth mentioning a mini-comeback⁸³ for Visual Basic for Applications (VBA) as a platform for delivering malicious content through email attachments. VBA malware was particularly popular in the last years of the 20th century, with macro viruses accounting for a significant proportion of all malicious samples. In the past, the malicious code would use OLE automation techniques to access the Microsoft Outlook® automation interface, sending the infected document as an attachment or simply propagating to all opened documents by inserting malicious code into the standard Normal.dot template.

For a long time it was thought that malicious VBA code was extinct thanks to Microsoft's introduction of additional security features

that prevented automatic startup of code when a document was opened. However, this year we have observed VBA, embedded in Microsoft Office XML format documents, acting as the first stage of infection and downloading or dropping additional malware components. This, however, had to be achieved by using social engineering tricks to convince users to open the document and explicitly allow the VBA macro embedded in the file to run.

Another trend is the reappearance of Visual Basic Script (VBS) malware, with the most common family being the Jenxcus⁸⁴ worm. Jenxcus is a relatively simple worm, which owes its success to inventive techniques used for spreading and launching itself.

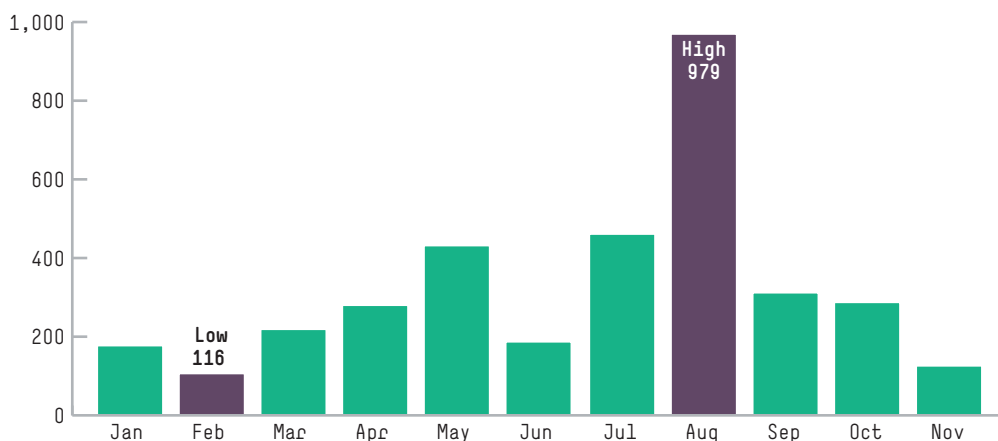


Figure 9. Top malware samples discovered by ReversingLabs in 2014, by volume per month

The worm is often delivered through a fake Adobe Flash updater setup file whose download is triggered when the user visits a maliciously crafted website—for example, a spoofed YouTube site. Once opened, Jenxcus enumerates mounted network drives and copies itself to them. In addition to that, Jenxcus creates a link with a base file name identical to the base name of a file that already exists on the drive. Users may unknowingly click on the malicious link instead of the file and launch the malware on their systems. Jenxcus also provides a backdoor to the infected computer by connecting to a website and allowing the attacker to send commands to control it.

by a malicious TOR exit node. It works by intercepting downloads of Windows executable files and modifying downloaded files on the fly to include additional malicious components designed to gather intelligence and steal user's data that is uploaded to the malware's command and control servers.

The key take away from the Onionduke story is that using TOR may help users stay anonymous, but it will not make them secure. TOR users must remember that the Internet traffic is routed through TOR exit nodes, and not all participants in the TOR network can be considered benevolent. Furthermore, users should not download executable files via TOR (or anything else) without using some sort of network encryption mechanism such as VPN.

We finish this brief overview of notable malware discovered in 2014 with Onionduke.⁸⁵ Onionduke is malware delivered

⁸³ virusbtn.com/virusbulletin/archive/2014/07/vb201407-VBA.

⁸⁴ microsoft.com/security/portal/threat/Encyclopedia/Entry.aspx?Name=Worm:VBS/Jenxcus#tab=2.

⁸⁵ f-secure.com/weblog/archives/00002764.html.

Proliferation of .NET malware in 2014

There was a marked increase in .NET malware in 2014, and there are multiple reasons why malware authors found this platform so attractive. The ease of the development platform, the availability of extensive libraries, the promise of multiplatform support, and the somewhat rudimentary state of the instrumentation and emulation by AV engines—as well as the lack

of advanced automated malware analysis that could target .NET applications—fueled an ongoing interest in .NET malware development by various actors.

While MSIL platform malware initially lacked the obfuscation and complexities of Win32 malware, actors have become increasingly inventive in using MSIL code injections, MSIL obfuscations and encryption. In 2014 we observed the following .NET malware and adware in the wild.

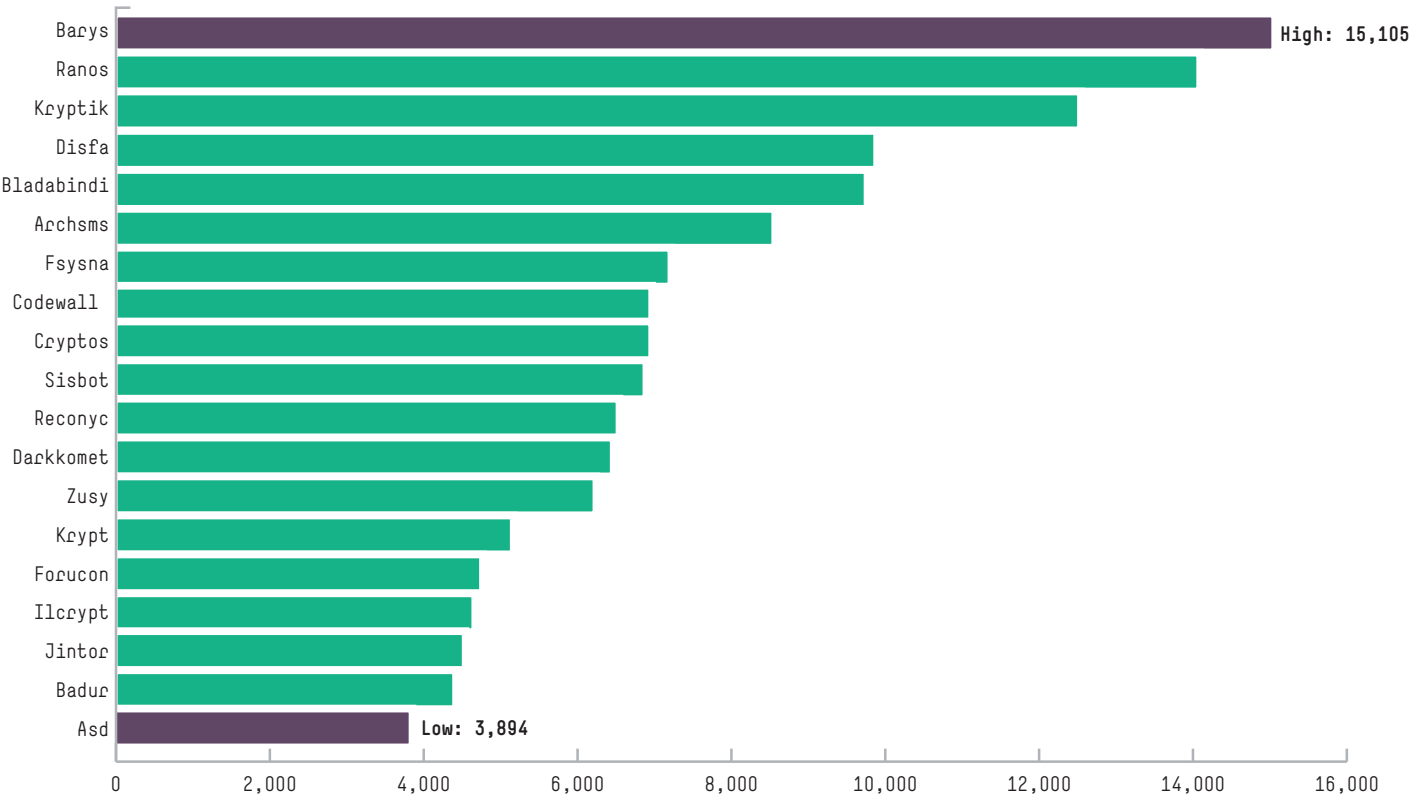


Figure 10. Top 20 .NET malware prevalence and distribution by family in 2014, from ReversingLabs data

On the top of the list of most prevalent .NET malware families we see Barys, Ranos, Kryptik, Disfa, and Bladabindi. Fsysna, Codewall, and Sisbot appear at about the same level of prevalence. Beginning from Fsysna the prevalence distribution

becomes considerably more even, without major spikes. This shows that the vectors of propagation remain consistent and are mostly associated with spam and social engineering downloads and not with major product-vulnerability exploits.

Ranos

This is a .NET Trojan developed to download and execute various malware families. There appears to be some naming confusion between Kazy and Ranos, however; whereas Kazy is classified as adware, Ranos is malware and is classified as a Trojan downloader. Due to its association as the downloader of other malware families, some variants of Ranos are misidentified as Bladabindi and Strictor. Ranos displays the following behavioral and functional characteristics:

- The majority of Ranos is written in .NET.
- Most variants of Ranos carry a compressed portion of their code in the resource area and are partially compressed and encrypted using the DNGuard .NET obfuscator and code protector. The protected code is stored in the resource section and is handled by the ZYDNGuard method.
- The developer metadata tends to contain information that aims to entice or socially engineer a user into opening a file (the Trojan has been distributed with filenames such as pictures.exe, Flash Player.exe, and so forth).
- Most of Ranos' functionality is for downloading and dropping arbitrary files from remote locations.
- Ranos has been observed to modify Windows hosts files in order to thwart antivirus and Windows update activities.
- Ranos tends to be distributed in files sized between 200 kb to 400 kb.

Bladabindi

Bladabindi is a very broadly reported malware family, often misidentified in practice. It is often confused with Zusy, Kazy, Disfa, and other MSIL malware families. Such name confusion and misidentification make it difficult to accurately judge prevalence and identify succinct behavioral characteristics with automated analysis. However,

Bladabindi displays the following behavioral and functional characteristics:

- Bladabindi .NET malware variants often have backdoor functionality that allows unauthorized access and control of a victim's computer.
- Some variants use a pluggable architecture that may allow the Trojan to be updated with new functionality after it is installed on a victim's computer.
- The malware adds a run key to the registry to execute when the system starts.
- Bladabindi allows access to private information such as volume information, computer name, OS version, user name, and so forth.
- The malware's backdoor functionality allows an attacker to manipulate files and folders as well as registry and firewall settings on the victim's computer.
- Some variants of Bladabindi are also known to dynamically inject MSIL code for obfuscation purposes, which further hinders analysis.

Barys

Barys is a .NET malware family, some variants of which are often misidentified as Bladabindi. Barys displays the following behavioral and functional characteristics:

- The Trojan drops itself into the Start Menu's Startup folder, thus ensuring that it starts at every system start (unless user manually deletes it).
- It modifies firewall rules.
- The Trojan uses strings obfuscation, as well as injecting itself in memory, thus further complicating its analysis.
- The Trojan has a proxy functionality and may be used to relay Internet traffic.



In 2014 we saw an increase in .NET malware related to **ATM attacks**. Such malware is installed on the machine and controlled through the keypad.

Sysbot

Sysbot is a .NET IRC bot malware family. Sysbot displays the following behavioral and functional characteristics:

- It drops itself as svchost.exe in the start menu, ensuring it executes at system start.
- It connects to IRC servers and uses predefined usernames to signal its activity.
- The worm ultimately gives unauthorized access to the infected computer and sends sensitive information to a remote attacker including IP address, processor, OS, enumerated USB devices, and logical drives information.
- The bot has the ability to update itself, which essentially allows it to download and install arbitrary files from remote locations.
- The worm is moderately string-obfuscated, which further complicates its analysis.
- The worm has multiple vectors it can use for propagation, including via IRC, MSN messenger, P2P networks, FTP servers, Facebook, and YouTube accounts, as well as removable USB drives.
- When spreading through IRC channels it entices users through social engineering to download and execute content.
- The worm can also use MSN Messenger to send files and copies of itself to various P2P client download locations with enticing names.
- In the case of spreading via FTP servers, it attempts to copy itself to ftp://<host>/index.exe, possibly hoping to be mistakenly executed in an attempt to get the host's index.

ATM malware attacks

In 2014 we saw an increase in .NET malware related to ATM attacks. Such malware is installed on the ATM and controlled through the ATM keypad. The majority of ATM host environments run a Windows OS and allow seamless execution of .NET applications. Because these systems are rarely examined by anti-malware software, such malware may persist for long periods of time without being

detected. At the same time the reports of such malware from the wild are low enough that they tend to remain under the general public's radar. The most notable .NET malware families in this category are Tyupkin and Padpin. The malware runs in place of ATM-controlled programs and intercepts coded keypad requests to dispense the cash to interested actors.

Attribution of .NET malware to the Syrian conflict

2014 was a year of much political and civil unrest around the world, including civil unrest and ongoing armed conflict in Syria. The conflict coincides with a notable spike in the development and propagation of malware attributed to the region, and .NET malware is a feature. Samples were spread through a number of compromised social media accounts associated with both sides of the conflict. There are cases in which Facebook pages were used to distribute malware, and others where users were enticed into downloading and running executables linked from political videos posted on YouTube. The samples were not heavily obfuscated and ranged from RATs to Trojan droppers and downloaders. Some malware associated with Syrian malware families are not strictly defined and may be detected as Variant.Kazy, Injector, MSIL.UL, MSIL.Agent, or Ransomlock. These generic name detections suggest the wide reuse of malware components and sources, and a lack of heavy obfuscation and encryption techniques.

Overall, we see an increasing number of .NET malware appearing on the scene. Actors are enticed by the promise of multi-platform execution, the availability of numerous third-party libraries and tools, and ease of development, coupled with ever-increasing levels of anti-debugging and code obfuscation techniques.

Linux malware

Usually, when we discuss malware, the focus is on Windows malware. This is due to the popularity of the platform for desktop users and the use of Windows as the main platform in many corporate environments. However, if we consider server infrastructure—especially the servers that serve the majority of Internet content—we see that the picture is somewhat different. Apache server is the most common⁸⁶ HTTP server application today and serves content on almost 60 percent of all websites, usually running on a Linux server distribution. The second most common HTTP server application, nginx, accounts for more than 22 percent of HTTP servers and usually also runs on a non-Windows platform. It is safe to assume that over 80 percent of Internet HTTP content

is served by servers that are running some flavor of Linux, which makes Linux-based systems, and especially server applications, an attractive target for attackers.

In the last couple of years there has been a renewed interest in malware written for Linux, and we felt that it was important to examine this in our report.

Top Linux malware

We tend to hear more about Linux when a major vulnerability in software is connected with it, such as was the case with CVE-2014-6271 (Shellshock). The majority of Linux malware belongs to categories that allow attackers to use affected systems as platforms for launching Distributed Denial of Service (DDoS) attacks.

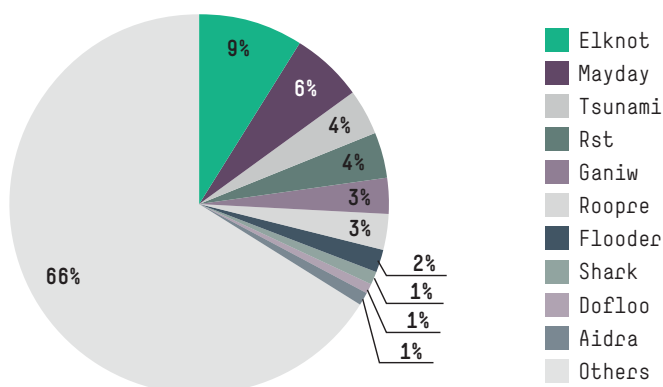


Figure 11. Top Linux ELF (Executable and Linking Format) malware discovered by ReversingLabs in 2014, by family

However, in the top 10 Linux ELF-based malware discovered in 2014, there are a couple of exceptions to the DDoS theme. The first of the exceptions is RST, a parasitic virus that's been around for more than 10 years. It infects other ELF executable files; this is an older style of malware and its inclusion in the list of top Linux malware is an indication that there is little awareness of the need to run anti-malware software on Linux systems.

The second exception is the Aidra⁸⁷ worm, accounting for more than 1 percent of all discovered ELF samples. Aidra, together with its counterpart Darloz, is designed to scan

the Internet for small office or home Internet routers and other devices and to spread by exploiting vulnerabilities or using default credentials if the telnet service on the target is running.

Aidra and Darloz belong to a relatively new class of malware that exposes some of the risks posed by different Linux implementations on small devices. With the development of the Internet of Things, many non-obvious computing devices will run Linux and connect to the Internet so we can expect more attacks like this in the next few years.

⁸⁶ w3techs.com/technologies/overview/web_server/all.

⁸⁷ now.avg.com/war-of-the-worms/.

Other interesting Linux malware

Linux, just like Windows, runs not just the kernel and standard OS tools but a number of other third-party applications. This holds especially true for Web servers, which are used to run many Web applications and open source content management frameworks such as Wordpress, Joomla, and Drupal.

In addition to that, the developers of Web applications often use these open-source frameworks as base platforms for their own applications. Modifying standard frameworks can prevent easy updating of the underlying framework when a vulnerability is discovered. This makes many applications on the Internet vulnerable to attacks, allowing attackers to modify applications so that they serve malicious content to website visitors.

The malicious content usually takes the form of additional JavaScript routines generated by server-side code uploaded by attackers. The server-side code can be implemented

in a scripting language, such as PHP or Perl, or by modifying the server binary code, if access to the Web server user account has been obtained through a system exploit or by using stolen user credentials.

The most popular language for script malware on non-Windows platforms is PHP, and we have discovered thousands of new samples in 2014. The majority of samples are used for one of three main purposes. The first, as seen in the Redirector family for example, serves to redirect Web browser applications to other sites in the Web malware infection chain. The second purpose, as seen in C99shell and Webshell, is to allow backdoor access to the affected server through a Web user interface. The third category belongs to PHP scripts planted to recruit the host system into a botnet for spamming or launching DDoS attacks. Spambot, Pbot, and Ircbot belong to this category.

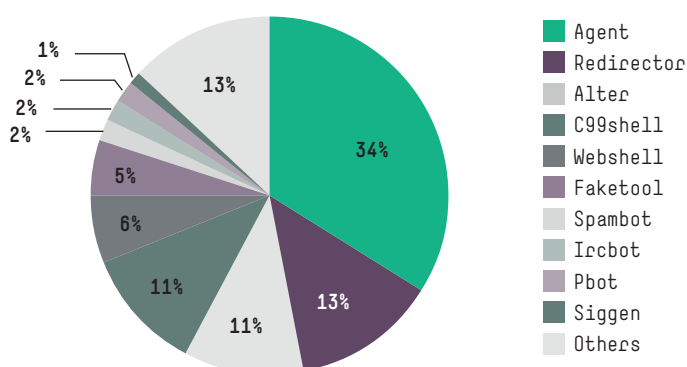


Figure 12. Top PHP malware noted by ReversingLabs in 2014, by family

The most interesting publicly known case of a wide server side compromise is known as Operation Windigo.⁸⁸ Windigo operators affected over 30 thousand servers using a combination of Ebury,⁸⁹ ssh credential-stealing malware, and Cdorked⁹⁰—modified Apache HTTP daemon binaries that redirected to malicious content and exploit toolkits designed to silently install and run several Windows malware families.

The operators behind Windigo made their money by sending spam from infected Linux servers and Windows systems. The Linux spamming side was implemented by a Perl-based malware called Calfbot.⁹¹

The security of Linux servers has a direct influence on the security of Windows desktops, as the Web is one of the main vectors for delivering malicious content. The greater the number of Linux servers compromised, the higher the probability that attackers will succeed in attacking users visiting compromised websites. In addition to this, if awareness of the need to protect Linux servers is not better developed, attackers will continue to use Linux as a major platform for launching spam campaigns and recruiting systems into DDoS botnets with great bandwidth potential.

⁸⁸ welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf.

⁸⁹ welivesecurity.com/2014/02/21/an-in-depth-analysis-of-linuxebury/.

⁹⁰ welivesecurity.com/2013/04/26/linuxcdorked-new-apache-backdoor-in-the-wild-serves-blackhole/.

⁹¹ virusradar.com/en/Perl_Calfbot.A/description.

Mobile malware

The year 2014 was a significant one for mobile malware. It was the year when mobile malware stopped being considered just a novelty. After all, in July we “celebrated” 10 years since the discovery of the first malware for mobile devices⁹²—Cabir or Caribe.

Cabir, which targeted the Nokia Series 60 Symbian platform, was a worm that used the Bluetooth OBEX protocol to spread from smartphone to smartphone. It was proof-of-concept malware created by the virus-writing group 29a (hexadecimal 666)—well-known for developing innovative pieces of malware at a time when malware developing was more of a hobby than a way to make money and attack organizations.

Fast-forward 10 years and we are seeing exponential growth in the number of discovered malicious apps, with the majority

of them targeting the Android platform. The actual reported numbers vary from company to company, but the general consensus inside the anti-malware industry is that there are over one million unique malicious apps known today, with several thousand more discovered on a daily basis.

Reasons for the popularity of the Android platform for the development of malicious apps are obvious. Android has become the most popular mobile platform, with over 1 billion⁹³ active users and over 1.5 million more registered every day.⁹⁴ Its reported market share is over 70 percent.⁹⁵ Google Play market, the most significant source of Android app distribution, currently hosts over 1.3 million apps but there is a large number of third-party app stores in the United States and worldwide, especially in China. These marketplaces, which often have less oversight, contribute to the growing malware problem on Android.

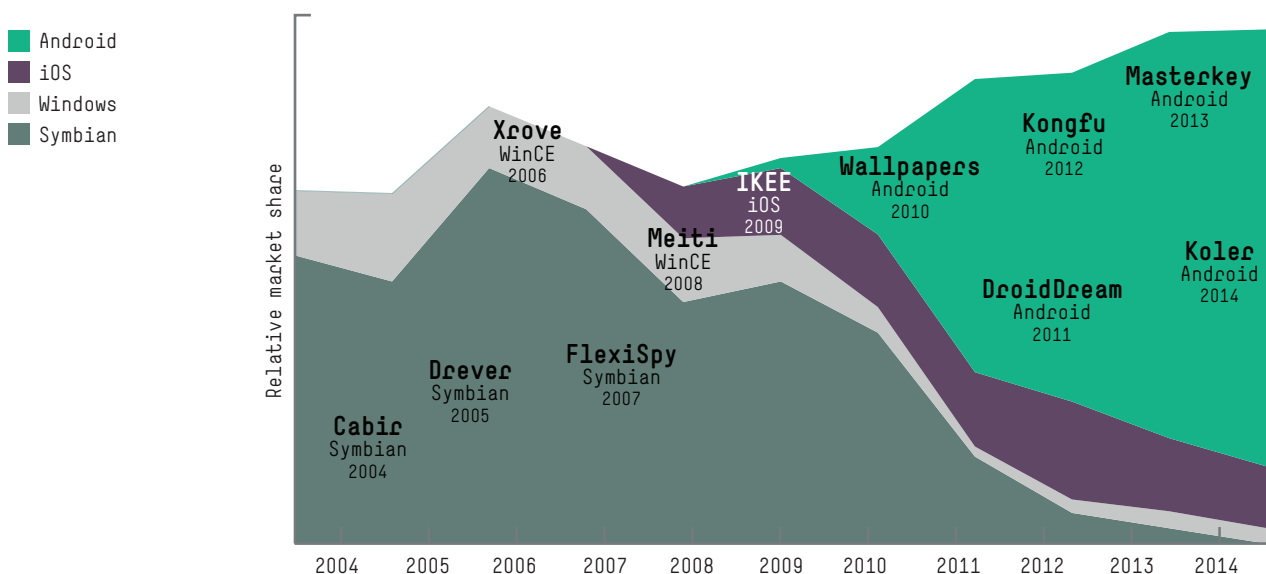


Figure 13. Ten years of mobile malware; note the rapid rise in Android popularity (and decline in Symbian popularity) in the last five years^{96, 97, 98, 99}

⁹² nakedsecurity.sophos.com/2014/06/01/from-cabir-to-koler-10-years-of-mobile-malware/.

⁹³ engadget.com/2014/06/25/google-io-2014-by-the-numbers/.

⁹⁴ androidcentral.com/larry-page-15-million-android-devices-activated-every-day.

⁹⁵ stats.unity3d.com/mobile/os.html.

⁹⁶ zdnet.com/article/mobile-os-market-shares-in-2005-symbian-51-linux-23-windows-17/.

⁹⁷ gartner.com/newsroom/id/910112.

⁹⁸ statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

⁹⁹ idc.com/proderv/smartphone-os-market-share.jsp.



Android anti-malware market

Since the discovery in August 2010 of the first malware for Android, Google's approach to allowing security apps for Android was very different from Apple's. While Apple exposed very little data that could be used for inspection and on-device malware classification and was actively discouraging development of security software for iOS, Google took a more hands-off approach and allowed anti-malware vendors to develop anti-malware software and publish it to Google Play.

However, a decision has been made that anti-malware apps on Android will not have any special privileges. As a result, unless the device is rooted, anti-malware apps are unable to prevent infection, but can only detect the installation of malicious apps onto the device and open a standard Android dialog that allows the user to remove the detected app.

APIs used by anti-malware solutions on Android are available to any apps allowed to listen to events triggered by the operating system when apps are installed. That may be one of the reasons for the large number of Android AV offerings—over 300 alleged anti-virus and anti-malware apps are currently hosted on Google—including all the traditional vendors with significant market share in the world of desktop anti-malware.

Nevertheless, awareness around the existence of security software for Android is relatively low. While we can expect more than 90 percent of Windows systems to be protected by security software, the overall protection level of Android devices is lower. We estimate that just below 40 percent of Android devices have some kind of anti-malware solution installed (based on the numbers displayed by Google Play market), which may be a bit low considering that the number of malicious apps for Android discovered daily is close to the number of Windows malware samples discovered around 10 years ago.

Meanwhile Google is not encouraging reports coming from anti-malware vendors, claiming that only 0.0001 percent of devices may ever encounter a malicious app.¹⁰⁰ That claimed number is backed by the data collected by the Google Play app. On the other side of the spectrum, vendors such as Kaspersky are reporting that the rates¹⁰¹ of malicious apps are several orders of magnitude higher on Android devices protected by their own software. The disparity between Google's and other vendors' data may arise from the fact that Google only measured downloads from Google Play market, while other vendor reports account for installs from all sources.

According to av-test.org, current anti-malware products for Android, although being rather rudimentary in terms of available technology and detection techniques compared to their Windows counterparts, are quite effective against known Android malware, with detection rates over 99 percent¹⁰² achievable by the majority of reputable vendors.

At the same time, with the release of Android 4.2 Google included its own Verify Apps anti-malware feature into the Google Play app. Verify Apps is an app scanner; it started as a simple feature that used SHA1 checksum calculation and cloud (Google Safe Browsing) API lookup to check apps for known-malicious samples during installation. With the release of Android Lollipop (5.0), Verify Apps has evolved more sophisticated protection mechanisms. These include re-scanning of apps after they are installed, as well as scanning of apps outside the Google Play market. We can expect that the Verify Apps functionality will develop into a fully featured anti-malware product, which will certainly be welcomed by Android users and the security industry.

¹⁰⁰ infosecurity-magazine.com/news/google-android-malware-threat-is-vastly/.

¹⁰¹ media.kaspersky.com/pdf/Kaspersky-Lab-KSN-Report-mobile-cyberthreats-web.pdf.

¹⁰² av-test.org/en/news/news-single-view/32-protection-apps-for-android-put-to-the-test/.

Top Android malware families in 2014

As expected, the majority of Android malware discovered in 2014 was found outside of the Google Play market, although there are instances in which malware was placed on Google Play by maliciously created developer accounts. The biggest family, similar to the situation with Windows platform, is Agent, which is a standard name reserved by anti-malware companies

for malware that cannot be classified with high certainty into any well-known family. Agent is followed by some of the usual suspects—for example, Opfake and Boxer, two related families originating and mostly targeting users in the Russian Federation and neighboring countries.

However, it is the families that are not seen in most top 10 Android malware family lists that have caught our attention, because they follow successful patterns previously seen and frequently used in Windows malware.

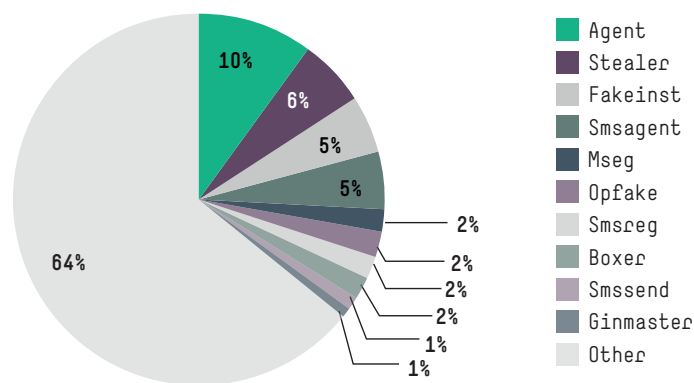


Figure 14. Top Android malware families in 2014, as detected by ReversingLabs

Notable Android malware in 2014

Although some of the methods featured here will not appear sophisticated compared to the methods used by Windows-based malware, most of them are new to Android, especially considering the fact that the first Android malware was only discovered a few years ago.

Ransomware

This year we have seen our first purposefully made ransomware samples, with functionality that prevented the users of infected devices from working. This was usually achieved by a combination of social engineering techniques used to convince the user to allow a malicious app to obtain device admin privilege (which is different from the user privileges granted to apps by the underlying Linux kernel) and interception

of various system events that prevented the user from launching any other app or terminating the malware.

The first ransomware, which belongs to the Android.Trojan.Tlock family, was discovered in April and contained basic functionality for locking the screen and preventing the user from navigating away from it. The app purported to be a free anti-malware app by Norton, but after presenting a fake anti-malware user interface, it displayed a warning that appeared to come from the FBI.

This technique is consistent with techniques used by some of the desktop-based malware families, such as Reveton or BrowserLock. The Tlock family evolved over time and reached full ransomware functionality that included allowing the user to unlock the device by entering a valid MoneyPak voucher number into the application UI.



Tlock was not the only Android ransomware family in 2014. New families also included Koler and Simplelocker. Simplelocker was written for the Russian and Ukrainian markets and attempted to encrypt some file types on the external memory card using the AES encryption algorithm with a hard-coded phrase to initialize the algorithm, which allowed for easy recovery of encrypted data. In addition to that, Simplelocker uses the TOR .onion domain for communications to its command-and-control (C&C) center, which makes it the first Android malware to actively use TOR.

One of the mitigation factors for apps attempting to encrypt documents is Google's introduction of constraints for

accessing apps on the external memory card in Android 4.4 (KitKat). The new access permissions prevent third-party apps from writing outside their own directory, similar to how a sandbox prevents apps from accessing another application's data on the main internal memory card.

However, even without the ability to encrypt documents, Android ransomware has the potential to prevent users from accessing their devices, simply by employing screen-locking techniques.

This chart shows an increase in the number of Android ransomware samples discovered monthly from the moment in April when the first Android ransomware was discovered.

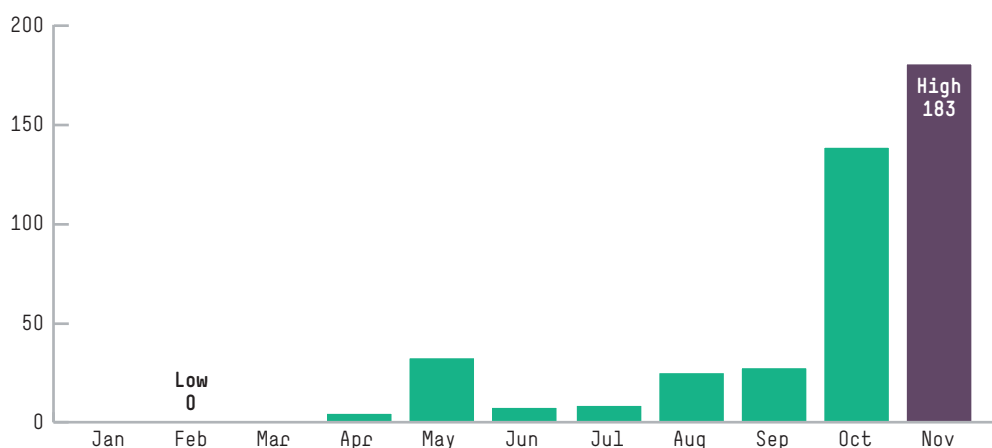



Figure 15. Number of Android ransomware samples discovered per month by ReversingLabs ; note January-March absence



One attack used Web injections and social engineering to install fake **banking apps** onto smartphones.

SMS malware

Sending SMS messages to premium-line numbers was the first payload used by Android malware writers. This technique was particularly useful in Europe, where SMS still remains one of the more popular services allowing vendors to monetize their services (it allows users to pay for small items, such as ringtones, by sending SMS messages).

The popularity of premium SMS services and instant payments to service providers in Europe provided malware writers with a platform that they successfully used to their financial benefit for several years. SMS malware usually pretended to be a game or a similarly interesting app, but soon after the user granted the app permission to send SMS messages, it would start sending them to the premium lines, making users pay an unexpected price premium for what should have been free.

Opfake, Boxer, and Fakeinst are the most common families in Android SMS malware.

It is worth noting that most of the SMS Trojans target Russians and citizens of other countries of the former Soviet Union. However, the time of SMS Trojans may be coming to an end. Kaspersky Labs researchers have attributed an increase in the number of discovered Android SMS Trojans shortly before July to new rules introduced by the Russian telecom regulator for services paid by SMS. These new rules mean that the providers of SMS services now need to send a confirmation code to users, which must be confirmed by the user before continuing on to use a premium line service.

However, it is more likely that the writers of SMS malware are simply regrouping and working on addressing those new rules. It is likely that we will continue to see SMS Trojans, especially in Europe, in the foreseeable future.

Banking Trojans

Banking Trojans became more prolific this year, with several families designed to attack the transaction authorization system that uses mTANs sent to users' smartphones over SMS. The Android.Trojan.Faketoken family intercepts SMS messages from banks in order to forward them to locations controlled by the attacker.

This attack was first employed in a mobile component of the Zeus (Zitmo) family, which coordinated attacks on users of Internet banking by using Web injections and social engineering to install fake banking apps onto smartphones. Once a user logged into Internet banking through the desktop browser, the desktop component of Zeus (Zbot) would conduct a fraudulent transaction, often using an automated transaction system (ATS) built from the JavaScript code injected by Zbot.

The mobile component was essential for intercepting and forwarding the mTAN required to conclude the fraudulent transaction. Stealing and forwarding mTANs is the most commonly seen function for mobile banking malware.

A second common attack pattern, mostly targeting Korean banks, is delivered as a fake mobile banking app that attempts to replace existing mobile banking apps with malicious copies. One of the more interesting examples was Android.Trojan.Gepaw, discovered in January. Gepaw was one of the first Android malware samples to be purposefully installed by its Windows component if an Android device was connected to a desktop.¹⁰³

Overall, 2014 was a significant year for mobile banking malware, but we have not yet seen it reaching the potential we have observed with Windows banking malware. We will continue to carefully monitor the development of mobile banking malware in 2015.

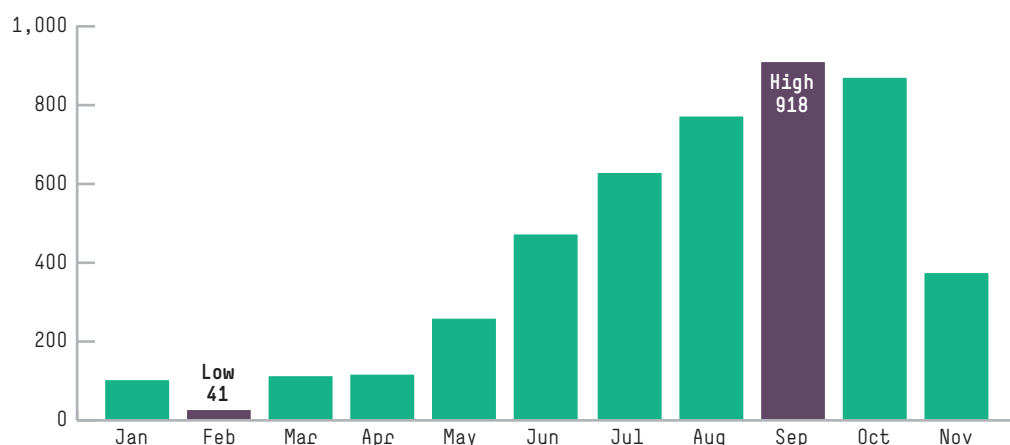


Figure 16. Number of Android banking Trojan samples discovered monthly during 2014 by ReversingLabs

Conclusion

The targets rise and fall, but the overall trend in volume is unambiguous: Malware threats are more widespread and pernicious than ever. The creators of malware have become smarter and more aggressive and will continue to do so, both developing new forms of attack and repurposing older approaches and known vulnerabilities. It continues to be critical for enterprises to

ensure all systems are protected, regardless of platform; though not a cure-all, anti-malware protections remain an effective defense, particularly against attacks aimed at long-known vulnerabilities. Those ongoing efforts are the proper and measured response to those in the industry who seem to dismiss the usefulness of anti-malware protections. They won't catch every attack by modern malware, but they'll miss 100 percent of the threats they're not deployed against.



Risks: Spotlight on privacy

In the enterprise, risk is a broad security concept—it is evaluated and then managed, remediated, reassigned, or simply accepted. But to many consumers and enterprises in 2014, risk could be narrowed down to one question and one response: *Is my private information in danger of being exposed? Fix it!*

Though 2013 put high-level privacy issues in the spotlight with Edward Snowden's surveillance disclosures, it was 2014—The Year of the Data Breach,^{104, 105} with shops from Neiman Marcus to Goodwill in the spotlight—that truly drew the attention, and ire, of business and consumer alike. A landmark decision further ensured that commercial enterprises must consider the confidentiality, integrity, and availability of user information—the three core tenets of information privacy—in their security equations.

We'll look more closely at the mechanics of data breaches, specifically point-of-sale breaches, [later in this report](#). For now, it's worth noting that attackers tallied considerable data-breach totals in 2014, with at least a dozen reported compromises affecting more than 10 million records apiece and three—Home Depot, eBay, and a group of South Korean financial entities—reporting over 100 million affected records apiece.¹⁰⁶ These numbers ignore both Target's 110 million-record breach, which may be said to have kicked off the mayhem in December 2013, and the highly publicized "CyberVor" incident, which likely overstated its claims to holding 1.2 billion records but raised privacy and data security yet again as a topic of wide concern.

Of course, a larger breach may not affect all or even most of the users whose records it touches, and a breach of just a few records can be disastrous if the right individuals' information is obtained. The Wyndham

Hotels and Resorts lodging chain, for instance, was breached three times in 2008 and 2009, with under a million records affected in total—hardly a blip on the radar in 2014. However, a 2012 suit brought by the Federal Trade Commission against the company alleged that the chain made multiple false claims in its privacy policy about security measures in place to protect consumer data, and that its failure to actually safeguard consumers' information caused substantial injury in violation of Section 5 of the FTC Act, which governs unfair or deceptive practices.¹⁰⁷ *That case became Federal Trade Commission v. Wyndham Worldwide Corp.*¹⁰⁸

Wyndham argued that the FTC didn't have the authority to set unfairness standards for cybersecurity—and that it was the businesses (specifically banks and credit-card issuers) who pay to cover breach costs, not the consumers whose private information is affected, who are actually harmed by such breaches.¹⁰⁹ It also argued that the FTC must issue regulations about cyber security before making any claims about its fairness or lack thereof.¹¹⁰

The ruling came back at the District level in April, and the FTC won on all three counts—a remarkable judicial affirmation of the FTC's role in enforcing data-security standards in support of consumer privacy.¹¹¹ (The ruling is under appeal at the Circuit level.) As other government agencies and legislative bodies struggle to address cyber security—whether through frameworks (NIST)¹¹² or procurement rules-making (Department of Defense)¹¹³ or proposed legislation (HR 5793)¹¹⁴—the FTC's case puts privacy firmly in the cyber security conversation. In the words of legal scholars Woodrow Hartzog and Daniel Solove, "the implications of this case could not be more important for data security as well as for privacy."¹¹⁵

¹⁰⁴ techrepublic.com/article/prevent-2015-from-becoming-another-year-of-the-data-breach/.

¹⁰⁵ darkreading.com/attacks-breaches/the-year-of-the-retailer-data-breach/d/d-id/1317462.

¹⁰⁶ breachlevelindex.com/#!breach-database.

¹⁰⁷ ftc.gov/news-events/press-releases/2012/06/ftc-files-complaint-against-wyndham-hotels-failure-protect.

¹⁰⁸ ashkansoltani.files.wordpress.com/2014/04/ftc-v-wyndham-opinion.pdf.

¹⁰⁹ privacyassociation.org/news/a/ftc-v-wyndham-round-one/.

¹¹⁰ ashkansoltani.files.wordpress.com/2014/04/ftc-v-wyndham-opinion.pdf.

¹¹¹ privacyassociation.org/news/a/the-wyndham-decision-where-does-cybersecurity-legislation-go-from-here/.

¹¹² nist.gov/cyberframework/upload/cybersecurity-framework-021214-final.pdf.

¹¹³ executivegov.com/2014/01/report-govcon-procurement-cybersecurity-could-see-changes-in-new-defense-policy-bill/.

¹¹⁴ [congress.gov/bill/113th-congress/house-bills/5793?q={%22search%22%3A\[%22cyber+security+chain+management%22}\]](http://congress.gov/bill/113th-congress/house-bills/5793?q={%22search%22%3A[%22cyber+security+chain+management%22}]).

¹¹⁵ docs.law.gwu.edu/facweb/dsolove/files/BNA%20FTC%20v%20Wyndham%20FINAL.pdf.

Americans seem to be highly aware of privacy and data-security issues, but they are deeply **pessimistic** that they can do anything about them.

Other types of privacy issues seemed to be of less concern in 2014, but regulatory and business activity continued. In Europe, privacy advocates successfully advanced the “right to be forgotten” by online search sites such as Google. That ruling, in *Google Spain SL, Google Inc. v. Agencia Espanola de Proteccion de Datos* (Mario Costeja González), sounded somewhat unusual to Americans accustomed to less comprehensive privacy legislation, but well in line with European understanding of the responsibilities of data processors.¹¹⁶ As Google and other search sites figure out what compliance may entail, some observers suggest that “the right to be forgotten everywhere”¹¹⁷—and the European Court of Justice’s rejection of the separate corporate entity doctrine (which allows non-European companies to operate in Europe through subsidiaries while denying that European law applies to the entire company)—may extend far beyond the original case.

The Snowden revelations concerning data surveillance seemed to lose their impact on privacy-concerned consumers,¹¹⁸ and a few other areas of traditional interests to privacy mavens also found little purchase among civilians. An ineradicable “supercookie” being deployed by Verizon drew remarkably little concern beyond traditional privacy circles.¹¹⁹ A survey in November by Truven Health Analytics and NPR indicated that consumers have few concerns about privacy and electronic medical records, at least as practiced by their own physicians.¹²⁰ In fact it’s possible, according to a Pew Internet survey that promises to be the first in a yearlong series surveying consumer attitudes to privacy that the Snowden revelations have backfired on Americans in a very particular way: They’re highly aware of privacy and data-security issues, but they are deeply pessimistic that they can do anything about them.¹²¹

For enterprises—some feeling as helpless as any consumer as they watch competitive data, confidential business plans, and executive emails make headlines—the current state of privacy presents the proverbial challenges and opportunities. The level of political and legislative activity will inevitably continue to rise, with some efforts more appropriate than others.¹²² The international standard community is working to move standards that will cover fields such as cloud computing.¹²³ If the FTC does not ultimately prevail in its attempts to establish its jurisdiction over security and privacy, it’s reasonable to expect that the current patchwork of federal and state regulations will expand. So will the need for enterprises to competitively track and manage changes¹²⁵ and to find a way to cooperatively address threats. Some observers see opportunity afoot, with privacy-proactive businesses offering transparency and safety as a differentiator¹²⁶ if not an outright customer requirement.¹²⁷

Still, some issues can elicit an engaged response. In particular, there appears to be growing consumer awareness about privacy issues at the Internet of Things level, whether that’s concern that one’s TV¹²⁸ or thermostat¹²⁹ is a security and privacy risk or something more systemic. The mass theft and online posting in summer 2014 of private photos from hundreds of celebrity-owned iPhones blurred the line between security and privacy, as what first appeared to be an intrusion shaped up to be a bad combination of poor password choice and inadvertent saving of images to the cloud by users who didn’t quite understand the implications of automatically doing so. In the long run these privacy breaches incrementally raise user and enterprise awareness of good security practices, but the combination of big breaches and angry governmental cries to “fix it” will likely continue to be the privacy story in 2015.

¹¹⁶ natlawreview.com/article/article-29-working-party-agrees-right-to-be-forgotten-guidance-following-may-2014-cj.

¹¹⁷ privacyassociation.org/news/a/the-right-to-be-forgotten-everywhere/.

¹¹⁸ pewinternet.org/2014/11/12/what-americans-think-about-privacy/.

¹¹⁹ eff.org/deeplinks/2014/11/verizon-x-uidh.

¹²⁰ truvenhealth.com/Portals/0/NPR-Truven-Health-Poll/NPRPulseDataPrivacy_Nov2014.pdf.

¹²¹ pewinternet.org/2014/11/12/public-privacy-perceptions/.

¹²² techdirt.com/articles/20141003/17382028725/politicians-cynically-using-jp-morgan-hack-to-try-to-pass-laws-to-diminish-your-privacy.shtml.

¹²³ privacyassociation.org/news/a/one-stop-cloud-compliance-how-the-iso-new-cloud-security-standard-could-change-cloud-computing/.

¹²⁴ privacyassociation.org/resources/article/full-report-benchmarking-privacy-management-and-investments-of-the-fortune-1000/.

¹²⁵ associationsnow.com/2014/09/how-can-retail-and-financial-groups-prevent-another-target-breach/.

¹²⁶ fortune.com/2014/11/18/data-privacy-competitive-differentiator/.

¹²⁷ wired.com/2014/11/arab-spring-of-privacy/.

¹²⁸ bgr.com/2014/10/31/smart-tv-privacy-and-security/.

¹²⁹ usatoday.com/story/tech/2014/01/16/google-acquires-nest-privacy/4518317/.

Exposures

Emerging avenues for compromise: POS and IoT

A number of different factors may arise and lead to different types of security exposures. Novel technology can lead to exposures, as the implications of new technology can sometimes be difficult to guess and avenues of attack can be unexpected until observed in practice (reminding us that sometimes, we should think like an attacker). The following section addresses and critically examines some of the recent and emerging fronts in the security wars.

First, let us look in detail at technical aspects of one of the biggest security stories of 2014—the point of sale (POS) system breaches at a number of major retailers in the United States, most notably Target and Home Depot. Both breaches resulted in the compromise of millions of customers' credit card and account details and were perpetrated using malware specifically created to target POS systems. This type of malware has been seen in the wild for some years now, and in this section we track its ongoing development by examining the evolution of three notable examples—Dexter, BlackPOS, and Mozart. (Information on additional POS-focused malware, Decebal and Back-off, can be found in the [glossary](#).)

Considering the sensitive financial data that is processed and used by POS systems, they are an obvious target for attackers. As protections have increased at different points in the retail transaction process, particularly in the transportation and storage of financial data, attackers have naturally looked to other intersections for possible points of compromise—in this case, at the actual point of sale. In the second part of this section, however, we will look at technologies where an attacker's path to monetization is less obvious.

The second part of this section looks at emerging technology and security implications for the Internet of Things (IoT). To a large degree, the security aspects of the increasingly ubiquitous networked computing we see in IoT technologies are untested, but questions regarding the implications of increasing convergence abound. Key areas of concern are sensitive data leakage, confidentiality (such as those reported in a 2014 HPE wearable-devices blog post¹³⁰), and data integrity. We look at some of the issues around these pervasive technologies and consider possible future trends.

The evolution of POS malware

In recent years, we've seen multiple POS infiltration incidents, with particularly high-profile breaches in 2013 and 2014 at Target and Home Depot. In the Target breach, the details of over 40 million credit and debit cards and the information of 70 million customers were stolen. In the case of Home Depot, 56 million credit and debit card account details were taken. And these are only the biggest incidents—there are undoubtedly more—both publically reported on and otherwise, but even these are two too many. The modus operandi used by the attackers to capture financial information in these breaches was POS malware. POS malware is not new, but 2014 saw considerable development in these malicious programs. They have evolved rapidly from being primitive and basic to advanced and complex. In this section, we look at three recent POS malware types in order to better understand the threat landscape in this space and to examine how they are evolving. All three are somewhat notorious. The first is Dexter, which was discovered in the wild in 2012. The second is BlackPOS, the malware that was used for the Target breach, and the last is Mozart, which was used in the Home Depot breach.

¹³⁰ h30507.www3.hp.com/t5/Applications-Services-Blog/Wearable-Technology-and-the-challenge-of-Fragmentation/ba-p/160704#VfDmE0cQ5s.



Capturing financial data: Process enumeration and memory acquisition

In order to capture financial data, most POS malware uses a technique called RAM scraping. RAM scraping occurs when the malware enumerates the processes and virtual memory space of the target machine looking for track 1 and track 2 data. (Track 1 and track 2 data is stored on the magnetic strip on the back of the card. It includes the account number associated with the card, its expiration date, and additional details that determine how and which transactions will be processed.) Process enumeration is usually performed by the EnumProcesses API or the CreateToolhelp32Snapshot and related APIs. Regardless of this difference, the two methods function in the same way.

In order to limit the amount of enumeration that is required to capture data, POS malware may use different techniques for selecting which processes to target. Looking at our case studies,¹³¹ BlackPOS, for example, only scans a process named “pos.exe.” This is notable, as it implies that the authors of the malware had previous knowledge of the environment in which the malware would be run. The malware authors also save the “pos.exe” string in encoded form in the binary, so that the process being targeted isn’t obvious upon casual inspection of the executable. In contrast, Mozart and other POS malware types use a blacklist approach. Blacklisting excludes common process names from the scanning process. This approach is more generic, as the malware can then be used for different POS systems using different process names. Many POS systems still use Windows XP machines, which have limited CPU and memory capacity, so saving resources by more effectively targeting relevant processes is an essential feature of modern POS malware.

After locating the processes to scan, the POS malware queries the virtual memory list allocated on the target process using the

VirtualQueryEx API. This routine is common to many different POS malware types. Then, after enumerating processes and virtual memory on the target machine, the POS malware uses the ReadProcessMemory API to read the process memory.

Enumerating processes and virtual memory is a very common feature of POS malware. While special customized code is not required for these tasks, for malware analysts, the process and exclusion lists can provide clues as to which product line the malware may be targeting.

Getting on the right track: Track data location methods

At this stage in the process, our example malware types diverge, and each uses a different approach to find the valuable track 1 and 2 data in the information that’s been scraped from memory. While older POS malware (such as vSkimmer) often used regular expressions to search for track 1 and track 2 patterns, this can be very CPU-intensive. Considering that POS machines have limited CPU and memory resources, better methods were required. Thus more recent POS malware tends to use custom track search routines.

Dexter

The search routine used in Dexter is simple and only recognizes track 2 records. The recognition of the track 2 pattern starts by locating the track 2 separator (denoted by “=”) and then checking for numeric characters before and after the separator character.

BlackPOS

BlackPOS searches for both track 1 and track 2 patterns. It checks if the current byte is either the track 1 separator (denoted by “^”) or the track 2 separator (denoted by “=”). If it matches the pattern, it checks the length of numeric characters that come before the separator in order to further validate that this is the correct data.

¹³¹ h30499.www3.hp.com/t5/HP-Security-Research-Blog/Hacking-POS-Terminal-for-Fun-and-Non-profit/ba-p/6540620.

Mozart

The Mozart track 1 and track 2 data recognition routine is more sophisticated than that used by the other POS malware. Of note, it does not buffer any data before attempting to match track separators. While malware such as BlackPOS buffers data and uses CPU cycles to find possible number patterns, Mozart checks the validity of track 1 and 2 data heuristically after it finds the track separators. Mozart is also able to recognize a broader range of character types in both data tracks than our other two examples, making it not only more efficient computationally, but giving it a broader application and possibly higher success rate.

Mozart also uses some additional verification mechanisms of interest. It checks the card issuer identification number after it gathers possible track records, then it utilizes the Luhn algorithm¹³² to check that the numbers it captured were potentially valid card numbers. Not only does this effort decrease the number of false positives, it decreases the size of the log file.

Track data management

While Mozart might be the best example of efficient targeting of track data, BlackPOS stands out for its track data management. Many POS malware types simply save the track records they capture direct to the file system. However, BlackPOS maintains a binary tree data structure of the track record in memory so that it can quickly check if the record has been collected previously. This has a few advantages. By removing duplicate records, it can reduce the size of the track record log file. With a smaller footprint on the network, it has a better chance to avoid detection. Storage is usually minimal on POS systems and by saving on local hard disk storage, BlackPOS can save more track records. In contrast, POS malware

like Dexter uses a simple function, StrStrA, to find duplicate track records: A binary tree data structure is a lot faster than just using a StrStrA string matching function when it comes to performance.


Interestingly, while Mozart was responsible for the theft of 56 million credit and debit card numbers, it doesn't contain any data management functionality; it simply saves every track entry it encounters while RAM scraping.

Data exfiltration

Once the financial data is captured, it needs to be transported from the affected system so it can be sold or otherwise used for fraud. Usually, this step of data exfiltration is not directly performed by the POS malware. As with other modern malware, POS malware is often modular and while it performs RAM scraping and saves the data to a log file on a local system or remote UNC location, different modules are responsible for uploading those log files to the attacker. That setup is common and was used in recently reported breaches, including those at Target and Home Depot.

Both Mozart and BlackPOS use very similar schemes for uploading log files. They both use a specific time frame to initiate a network operation to copy the files when the traffic will look least suspicious, and they both use Windows sharing to push the log files to a central location inside the compromised network. The most likely reason for this is that the infected POS machine itself might not have an Internet connection; therefore, the data needs to be transported to a location that does. From there, another component pushes the data outside of the network to machines under the attacker's control. BlackPOS is known to use an FTP program to upload its log files.

¹³² creditcards.com/credit-card-news/luhn-formula-credit-card-number-system-1273.php.



Using regular expressions is resource-intensive. It makes the scanning process slow and may affect the overall performance of the POS machine itself, making **detection** more likely.

Defenses against POS malware

The following lists a number of steps that businesses using POS systems can take to reduce the risk of being compromised by POS malware:

- Use multi-factor authentication; this will substantially increase the difficulty required to use compromised credentials
- Segment the network
- Limit allowed protocols
- Limit user privileges
- Initially monitor the addition of new users, particularly privileged users
- Monitor for excessive and abnormal LDAP (Lightweight Directory Access Protocol) queries
- Use application whitelisting on sensitive servers (those that are used for critical functions)
- Heavily secure and monitor Active Directory
- Migrate to an EMV Chip-and-Pin point of sale system, which makes cloning credit cards nearly impossible

By looking at the POS malware responsible for some of the recent big breaches, we've discovered that while they share a number of similar features, they do not appear to share the same code base. Simply put, the functionality is similar but the code is different, which suggests that they were developed independently by different

groups. Of course, some features, such as track record management are also different for each case. The most notable difference between them, however, is the track record recognition routines. Each malware uses its own method for locating track records and regardless of differences, the custom code they contain is much more advanced than that of older POS malware families that use regular expressions for this purpose. Using regular expressions is resource-intensive. It makes the scanning process slow and may affect the overall performance of the POS machine itself, making detection more likely as administrators investigate the source of the slowdown. Also, as RAM scraping is all about timing—when a customer swipes the credit card it is passed to the POS process, but there is no guarantee on how long the data will be intact in process memory. Finally, the faster you finish one loop of scanning, the greater the chances of catching new track records.

Our final takeaway though, regarding BlackPOS and Mozart in particular, is that they look customized—that is, they look as though they were developed for these specific breaches. BlackPOS contains an encoded form of a targeted POS process name, which might be specific to Target's POS environment. BlackPOS and Mozart also have hardcoded server IP addresses to which they push the log files of captured data. This tells us that these POS malware programs were built by people who knew the targeted environments.

The Internet of Things

While the “Internet of Things” (IoT) might just seem like one of the latest catchphrases, it is an important paradigm in computing and communications that is likely to have profound effects for security. In the IoT, the objects we use in the physical world are identified, labeled, and interconnected. While the technology for object connectivity has been around for some time, a confluence of additional factors has contributed to the current development of IoT, including significantly cheaper storage, advances in Big Data processing, and connectivity chipsets. The IoT enables these objects to not only communicate intelligently together, but also to monitor, measure, and act in response to changing environmental conditions, leading to greater efficiencies in all manner of applications, from remote healthcare, to industrial systems, to avoiding having your fridge run out of milk. The IoT is where the physical world meets the virtual in practice.

Understandably, the diverse and ubiquitous nature of the technology and devices that comprise the IoT gives rise to concerns regarding security and privacy in particular. The coming level of interconnectivity, combined with exponentially more data capture and storage, is unprecedented; in order for systems to respond appropriately to changing conditions, they must take accurate and ongoing measurements of their environment and the system or endpoint they are monitoring (even if that endpoint is a person). What is thought to be appropriate access to and use of this data is yet to be determined, but there are multiple dimensions that need to be considered. Also, as the IoT brings the physical and the virtual world together, the consequences of unethical or malicious use become increasingly “real”—what if an attacker hacked your car, or even your pacemaker?



Figure 17. Evans Data Corporation global survey of developers, conducted in spring 2014¹³³

A spring 2014 Evans Data survey showed that 17 percent of developers globally were working to develop applications targeting connected devices for the IoT and that 23 percent are planning to do the same in the next six months or so.¹³⁴ Even though Internet-connected devices have existed for some time, it seems that the acquisition of Nest Labs, the connected-devices company (and creator of the Nest learning thermostat), by Google for \$3.2 billion marked a turning point in the market, firmly focusing attention on IoT.¹³⁵ Several other important acquisitions soon followed, such as SmartThings by Samsung¹³⁶ and Dropcam by Google,¹³⁷ showing that momentum is continuing to build in the IoT industry and that growth is accelerating. (Our researchers spent some time in 2014 hacking a Samsung Smart IoT

TV with surprising results. Read the blog¹³⁸ for details.)

The explosive growth of IoT technologies is not accidental and is the result of the convergence of several technologies, including broadband and Big Data processing and acquisition. The burgeoning field of data collection and processing is tightly intertwined with that of IoT and the two areas otherwise fuel each other's growth.

The IoT normally implies interconnected hardware nodes. These hardware endpoints collect information about the environment and respond to the control functions of a user. These nodes could be part of a larger electrical device or could act as standalone units.

¹³³ evansdata.com/press/viewRelease.php?reportID=38.

¹³⁴ evansdata.com/press/viewRelease.php?pressID=212.

¹³⁵ investor.google.com/releases/2014/0113.html.

¹³⁶ samsung.com/us/news/23607.

¹³⁷ reuters.com/article/2014/06/22/us-google-nest-dropcam-idUSKBN0EW02820140622.

¹³⁸ h30499.www3.hp.com/t5/HP-Security-Research-Blog/How-I-learned-to-hack-my-TV-and-started-worrying-about-the/ba-p/6383829.



The endpoint wireless infrastructure is still in its infancy, and unfortunately a lack of collaboration in the industry during its development failed to create an open ecosystem that would accommodate heterogeneous devices and communication protocols. The lack of common interface solutions and security standards has led to proprietary implementations of protocol stacks and firmware updates, and as a result, has significantly expanded the surface for malicious attacks and vulnerabilities.

At its core, an end-node device normally consists of a CPU (central processing unit, which acts as a control unit), sensors, input and output modules, and a network processor together with either a wired or wireless network front end. Many functions are often combined within a single integrated chip (IC) solution.

The majority of current market solutions tend to employ MIPS- or ARM-based chipsets and to run flavors of Linux. This feature was most probably dictated by the availability of ready-made development solutions and tool chains (groups of programming tools that are used in series to create a product), which were adopted from set-top boxes, routers, and NAS (network-attached storage) devices.

For example, a first-generation Nest thermostat uses an AM3703 Sitara processor¹³⁹ from Texas Instruments. The thermostat is based on the ARM Cortex™-A8 architecture. The development tools include the Linux EZ Software development kit and the Android Development Kit for Sitara Microprocessors. Both packages are available for download from the TI website and are free of charge. These freely available development tool chains based on popular OSes, combined with relatively lower rates of power consumption, made this a lucrative processor for the IoT market. The first-generation Nest OS is based on Linux 2.6.37 and uses other free software components. The firmware image is locked so it only accepts signed firmware updates. Nest also provides unlocked firmware so it can accept unsigned firmware images. This allowed a third party to re-implement the basic logic of the thermostat as an open source project called FreeAbode.

Big appliance manufacturers, such as Samsung, GE, Whirlpool, and Bosch, are likely to follow this trend. Many also use

proprietary systems based on Linux or other popular OSes; Bosch is the driving force behind the idea of developing a universal and open-source smart model.

The IoT continues to grow. It continues to capitalize on new opportunities in areas such as sensor monitoring in traffic, railways, car-parks, the home, the local power grid, embedded medical devices (including wearable sensors), and computing. The areas for these applications include industrial robotics, automotive, factory automation, home security, agriculture, and more. According to a Gartner projection made at the end of 2013, the number of connected devices (excluding PCs, smartphones, and tablets) is estimated to be around 26 billion in 2020. This, according to their projections, will far exceed the number of PCs, tablets, and smartphones (which is estimated to be around 7.3 billion units in 2020).¹⁴⁰ Over the next few years more devices will incorporate multiple types of wireless connectivity, including Wi-Fi, ZigBee, Z-Wave, MiWi, and other proprietary protocols. Also, to satisfy the demand for micro power and always-on connected sensors, and to simplify non-interactive automation nodes, there are many low-power, low-cost Wi-Fi and processing modules fast coming to market.

There seem to be two large groups of hardware platforms targeting the various fields of IoT. The first group can run any sort of Linux or Windows embedded OS. The second group consists of less computationally intensive and power-hungry processors that are aimed at proprietary or open-source real-time operating systems with a small memory footprint (for example RTOS, Micrium uC/OS-II, uC/OS-III, or TI-RTOS-KERNEL). This second group is even more fragmented and tends to rely on proprietary software stacks and integrated development environments. There is an effort to bring homogeneity to the IoT ecosystem that is being driven by the ARM consortium and its partners with the development of an event-driven MBed OS specifically targeting low-power devices within the IoT realm. The solution rests on three pillars: as the MBed OS itself, the MBed device server (which acts as an MBed-powered IoT devices cloud aggregator and a portal for Internet applications), and a suite of MBed tools aimed at simplifying the development effort from the ground up to fully designed solutions. Most importantly,

¹³⁹ ti.com/product/am3703.

¹⁴⁰ gartner.com/newsroom/id/2636073.

these solutions are open source and attractive to a large proportion of makers and the developer community.

There's an ongoing effort to engage developer communities to participate in creating hardware, software, and firmware solutions covering network-connected, embedded, and deeply embedded devices. This of course is not limited only to hardware and firmware. The software includes developing services, cloud solutions, and Big Data analytical and aggregating platforms. The industry, taught by the experiences of companies like Apple and Google, understands that a competitive advantage lies with ease of development and an abundance of applications. This will drive the development of frameworks and standards even further.

On the hardware end of things the same methodology persists. Success lies in targeting a broad community of makers, including those who are not entirely skilled in electronic design or engineering. This implies modular hardware design with most of the functionality embedded into ICs. It gives rise to a prevalence of system on integrated chip solutions (SoIC), which simplifies hardware design and connectivity but unfortunately, due to proprietary firmware solutions and a lack of unified interface protocols, increases solution fragmentation even further.

Major chip manufacturers, realizing the need for SoIC that target the lower and middle end of IoT devices, are bringing comprehensive firmware and integrated development environment solutions to market. One popular approach is to separate the logic of the TCP/IP stack and Wi-Fi radios from a connected device. There are many manufacturers racing to compete in the fast growing IoT chipset and Wi-Fi modules field. Manufacturers to note include Microchip, STMicroelectronics, Texas Instruments, Freescale Semiconductor, Broadcom, Qualcomm Atheros, Nordic Semiconductor, Atmel, and others.

The current lineup of developing frameworks shows that possible solutions are likely to congregate around integrated and self-contained Wi-Fi modules connected to a host processor. Such modules will either offload networking services from a host controller, or will be capable of carrying application software on the network processor itself. These Wi-Fi modules are also likely to run a flavor of the Linux OS and its TCP/IP stack or another third-party OS, and to rely on lwIP (lightweight IP) or their own TCP/IP stack (often royalty-free with the use of proprietary chipsets such as Microchip and TI). The Wi-Fi module network processors, as well as host device processors, will gravitate toward a balance of computational power and the ability to sustain lower power consumption. This is especially important for "always-on" devices with lower connectivity bandwidth such as ambient light sensors, temperature and humidity sensors, and so on.

With the sector rapidly expanding and so many alternative solutions rushed to the market by different manufacturers, we should expect to experience further market fragmentation and a bigger variety of frameworks and available solutions. The initiative is there to introduce a common open-source framework, including connectivity layers as well as development tool chains and software (for example, MBed, OpenIoT, or IoT-SyS). This shows that popular IoT devices are most likely to run an open source OS such as AllJoyn, which was initially developed by Qualcomm but is currently sponsored by the AllSeen Alliance. This is the most prominent OS and thus likely to be accepted as an industry standard because it is cross-platform and has APIs for Android, iOS, OS X, Linux, and Windows. It also includes a framework and a set of services that will allow manufacturers to create compatible devices. Other operating systems worth mentioning are Contiki, RaspbianPi (based on a Debian distribution of Linux), RIOT, and Spark.

The availability of open-source solutions might act as a double-edged sword for discovering and **mitigating** vulnerabilities.

Conclusion

Despite many efforts to abstract and universalize the IoT framework there remain multiple actors and competing solutions in the market today, and the fragmentation of solutions and devices discussed in this section is likely to continue. Addressing security concerns such as attack vectors on such devices is likely to require individual solutions for each type and family of device—an extremely difficult situation. The limit on the number of possible “over the air” or “in service” upgrade-capable devices, especially in the lower-end cheap sensor segment, is likely to make mitigating such attacks even more difficult.

Attacks could involve various layers of the device infrastructure. This could include applications running on smartphones or tablets, cloud services—including firmware and network service stacks on Wi-Fi modules—as well as the firmware and application layer on the host processor. Various vectors of propagation could also be used, including compromising update files or exploiting network and host processor communication layer vulnerabilities, as well as possible vulnerabilities in cloud service infrastructures and smart device applications. The availability of open-source solutions might act as a double-edged sword for discovering and mitigating vulnerabilities. On one hand, common source is likely to be susceptible to the same vulnerability across a span of many devices, but on the other hand it will allow the broader community to discover, test, and mitigate vulnerabilities earlier, provided the firmware and software can be updated in a timely manner. Unfortunately in the case of many of the simpler IoT devices, this just might not be possible—at least initially. There are still a number of unknowns when it comes to the security of the IoT in practice.

The final section of our report brings us full circle and to one of the most important parts of this publication—controls. It answers the important question of why vulnerabilities arise and shows a very different view of the threat landscape. Being aware of the specific circumstances that give rise to vulnerabilities lets security practitioners address their root causes and make enterprises, not to mention your coding practices and software selection, considerably more secure.

Controls

In the past year we have seen the manifestation of several vulnerabilities that gathered a storm of media attention. The uproar around Heartbleed, Shellshock, and Poodle brought renewed scrutiny to software dependencies as they relate to software architecture. We noted new trends in where vulnerabilities are detected, as well as continued shifts in the types of weaknesses leading to them. To gain insight into the current state of software security, we analyzed a sample set of security audits performed by HPE Fortify on Demand on 378 mobile apps, 6504 Web apps, and 138 Sonatype reports from 113 projects. These audits include results from static, dynamic, and manual analysis.

In order to have a consistent view of the data analyzed for this Report, we’ve ensured that all identified issues were classified according to the HPE Software Security Taxonomy (Originally the “Seven Pernicious Kingdoms”), which was substantially updated and refined¹⁴¹ in mid-2014. These updates and refinements are reflected in HPE WebInspect, our dynamic analyzer. Our work to extend the taxonomy to other assessment techniques (such as manual analysis) and HPE Fortify products (such as HPE Fortify on Demand) continues in 2015.

¹⁴¹ gartner.com/newsroom/id/2636073.

Distribution by kingdom

The HPE Software Security Taxonomy is organized into kingdoms, which are collections of vulnerability categories that share a common theme. One of the metrics that gives us more insight into vulnerability trends is the distribution of

kingdoms discovered in applications, and especially how those change over time. The following graph compares the distribution of vulnerabilities discovered in Web applications across kingdoms in 2014 (using a sample size of over 6,500 apps) against those spotted in 2013 (using a sample size of over 2,200 apps).

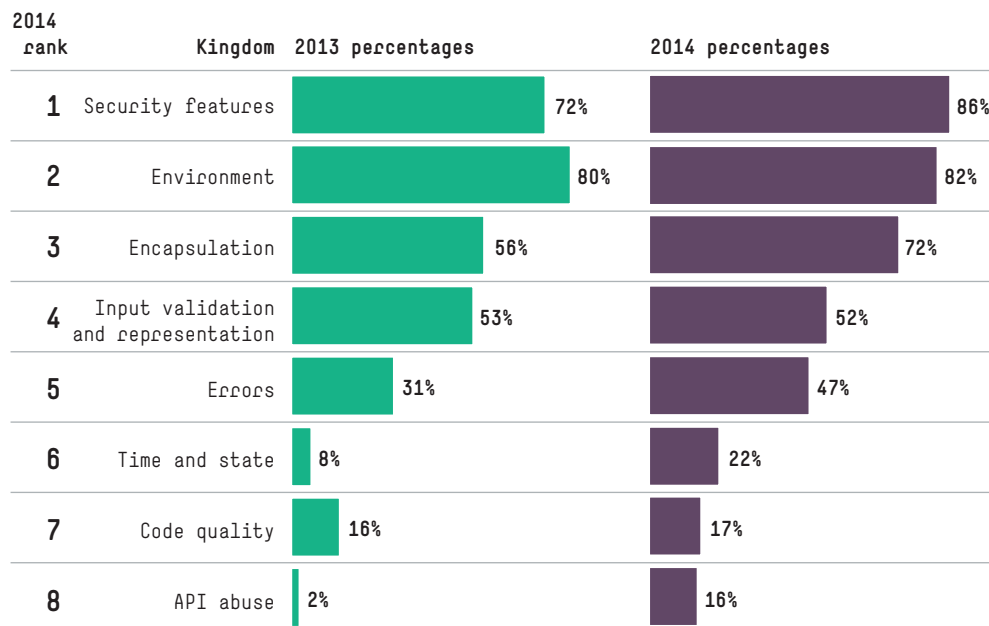


Figure 18. Web app vulnerability category distributions by kingdom in 2013 and 2014

In general, in 2014 more Web applications contained vulnerabilities in each kingdom than they did last year, with the exception of Input Validation and Representation kingdom (which decreased by just 1 percent).

In 2013, Environment was the kingdom represented in the greatest percentage of applications (80 percent), while Security Features took second place with 72 percent. This year the two kingdoms switched places. Even though the percentage of applications that contained Environment issues stayed relatively similar between 2013 and 2014 (80 percent vs. 82 percent), the percentage of applications that contain problems related to Security Features—including access control, privacy violation, password management, insecure transport, and security of cryptographic primitives—jumped from 72 percent in 2013 to 86 percent in 2014. Unfortunately, this statistic is once again consistent with the recent rash of privacy and confidentiality breaches, ranging from stolen personal data (Snapchat,¹⁴² University of Maryland¹⁴³), to credit card numbers (Neiman Marcus,¹⁴⁴ Home Depot¹⁴⁵), to

personal health information (Los Angeles Department of Health Services¹⁴⁶).

The same rise in breach reports explains jumps in other kingdoms. The percentage of applications that have API Abuse problems, many of which are related to misuse of SSL certificates, jumped from 2 percent to 16 percent. The percentage for the Encapsulation kingdom, whose categories delineate ways in which an application might leak system data to a potential attacker who can use this information to mount a bigger attack on the system, went up by 16 percent (from 56 percent to 72 percent). The Errors kingdom, whose categories are very much related to leaking system information in the form of error messages resulting from improperly handled exceptions, exhibited a similar jump from 32 percent to 47 percent. And many of the categories in Time and State kingdom that relate to improper session and account management contributed to the increase in the percentage of applications susceptible to such problems, from 8 percent to 22 percent.

¹⁴² gibsonsec.org/snapchat/fulldisclosure/.

¹⁴³ umd.edu/datasecurity/.

¹⁴⁴ neimanmarcus.com/NM/Security-Info/cat49570732/c.cat.

¹⁴⁵ bits.blogs.nytimes.com/2014/09/18/home-depot-says-data-from-56-million-cards-taken-in-breach/?_r=0.

¹⁴⁶ healthitsecurity.com/2014/03/07/los-angeles-county-dhs-reveals-168000-patient-data-breach/.

The jump in the percentage for the Time and State kingdom can also be attributed to the manual assessment findings, which were included in our dataset this year for the first time. Session and account management defects are more similar to design flaws, and therefore are hard to detect using automated techniques. However, they are still important to find and fix, and thus they inspire security practitioners to explore ways other than manual assessment—a cumbersome and error-prone process—for spotting design-level security concerns.

In addition to including the results of manual assessments in our analysis of Web application vulnerabilities, this year we also performed the same analysis on mobile applications. The following graph represents the distribution of kingdoms discovered in both mobile and Web application datasets that were assessed using static, dynamic, and manual techniques.

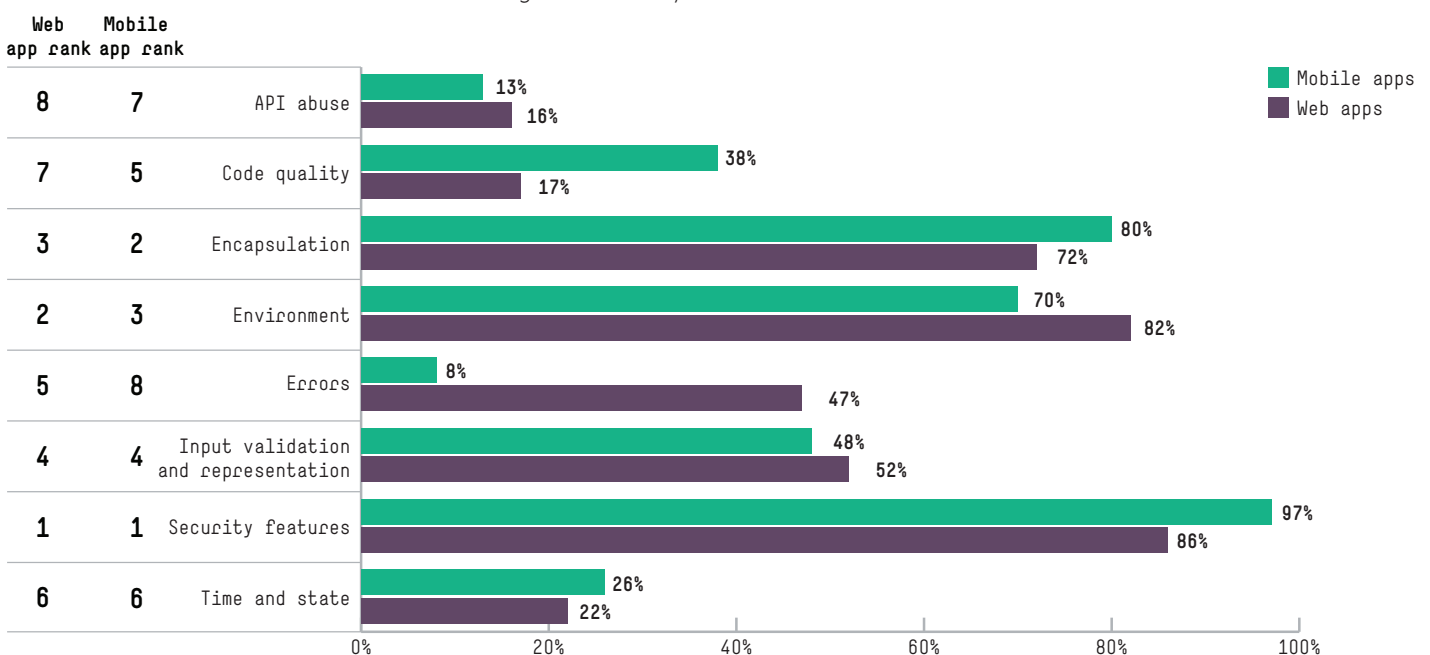


Figure 19. Web app and mobile app vulnerability category distribution by kingdom in 2014

It is clear from Figure 19 that the Security Features kingdom takes first place not just for Web applications, but also for mobile applications. Furthermore, the percentage of mobile applications that contain Security Features issues (97 percent) is higher than that of Web applications (86 percent). There are at least two factors which contribute to this finding. First, there are far fewer options for securely storing sensitive information on mobile devices, fewer choices of encrypted databases, and questionable mechanisms for storing cryptographic keys used for protecting this data. Second, the very nature of mobile application development requires developers to use security features, such as Android permissions or iOS data protection that they may not fully comprehend. This lack of clarity eventually leads to misuse.

While the numbers for the API Abuse, Input Validation and Representation, and Time and State kingdoms are similar between mobile and Web applications, the Code Quality and Errors kingdoms both exhibit drastic differences, and the numbers for the Encapsulation and Environment kingdoms are basically reversed.

Let's start by looking at similarities. As mentioned above, many of the API Abuse problems are about misuse of certificates, which equally applies to both mobile and Web applications. The same goes for session and account management flaws in the Time and State kingdom. Categories of vulnerabilities stemming from inadequate input validation and encoding are similar for both Web and mobile applications, and the

fact that the percentages of applications that contain such issues is nearly the same is indicative of the fact that developers with similar skillsets write code for both mobile and Web applications, and thus make the same kinds of mistakes.

It is interesting that the numbers for the Environment and Encapsulation kingdoms are reversed, but this can be easily explained as well. One of the biggest differentiators of the mobile ecosystem, compared to Web applications, is the fact that different mobile applications run side by side on the same device within the same environment and are able to communicate with each other. Therefore, there are more opportunities for data to cross trust boundaries in mobile applications, which explains the higher percentage of mobile applications that contain Encapsulation issues (80 percent vs. 72 percent). On the other hand, there are a lot more ways to deploy Web applications, a lot more Web servers to run the applications on, and a lot more opportunity for misconfiguration, which is why Web applications (82 percent) are more susceptible to Environment issues than mobile applications are (70 percent). This also explains why the number of Web applications that contain defects in the Errors kingdom is much higher than that of mobile applications (47 percent vs. 8 percent), because many of the categories in the Errors kingdom are about misconfiguring an application or a Web server in terms of handling exceptions or providing a custom error page.

As for the differences in Code Quality numbers, our data indicates that a much higher percentage of mobile applications—compared to Web applications—contains instances of null dereferences. The ability of mobile applications to communicate with each other is one of the reasons behind this. In the mobile world, developers do not always check the data coming from another application against null, because a lot of the time they are expecting the data to be coming from another component of the same application (rather than from an entirely different app) and thus assume they can trust it to be in the expected format. Second, our mobile application dataset includes both Android and iOS applications. Many of the Code Quality issues are related to C- (and therefore iOS-) specific problems, such as type mismatches, which simply don't exist in Web applications. This explains why the percentage of mobile applications (38 percent) that contain Code Quality issues is higher than that of Web applications (17 percent).

Breakdown of top five Web application vulnerabilities

At a high level, all vulnerabilities can be represented within a category of the HPE Software Security Taxonomy. Some categories can be solely represented by a single vulnerability (e.g., cross-frame scripting), while others could be a grouping of multiple vulnerabilities (e.g., Web server misconfiguration). Below is a quick look at five categories that affected the greatest number of applications.

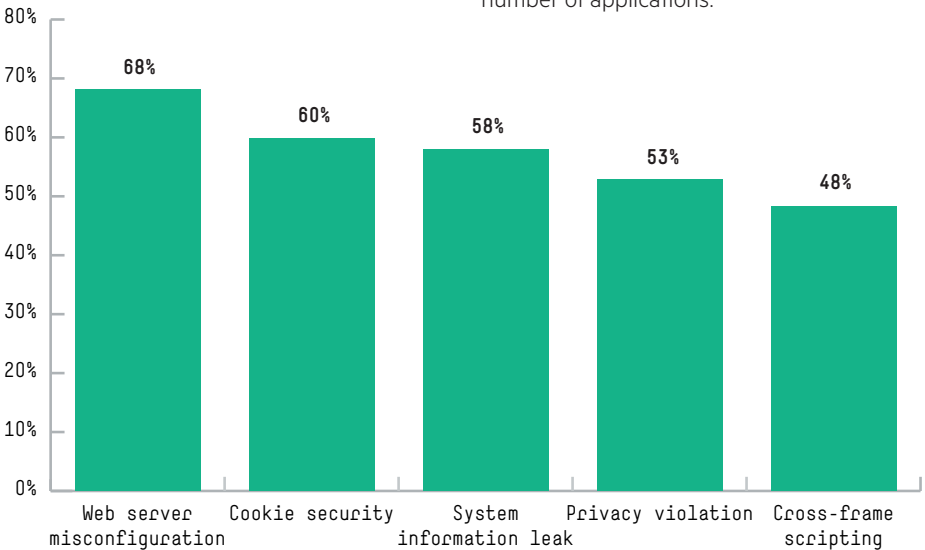


Figure 20. Top five vulnerability categories across applications evaluated in 2014

Cookie security and system information leakage continue to be prominent issues when compared to 2013's trends. Due to recent changes to the HPE Software Security Taxonomy, access control issues have now been merged into the greater Web Server Misconfiguration group. With this in mind, misconfiguration issues gained the top spot based on 2014 data. Interestingly, Transport Layer Protection (now called Insecure Transport) and Cross-Site Scripting moved down to seventh and eighth places respectively. These were replaced by Privacy Violation and Cross-Frame Scripting. The HPE Cyber Risk Report released in 2013

performed a detailed analysis of defenses against cross-frame scripting. Two years later, HPSR continues to find this to be a prevalent issue across Web applications.

Top 10 Web application vulnerabilities

Each category within the taxonomy may be further refined based on specific characteristics of the vulnerability. The chart below represents the most prevalent Web application vulnerabilities seen in 2014.

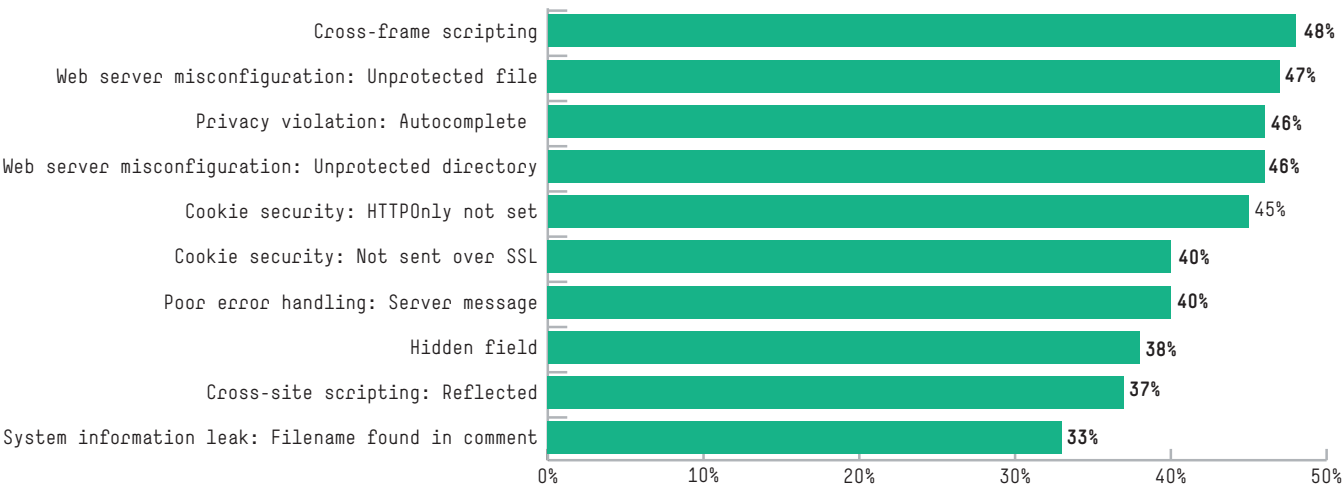


Figure 21. Top 10 common vulnerabilities discovered in Web applications in 2014

It is interesting to note that nine of the top 10 vulnerabilities are represented within the top five categories shown in Figure 20. Based on

the above chart, we can see that cross-frame scripting affects the most applications.



Because more attention is paid to critical issues, we decided to create a view of top 10 critical issues that plague Web applications.

Below is the representation of the subset of the vulnerabilities of critical risk found in Web applications and their prevalence.

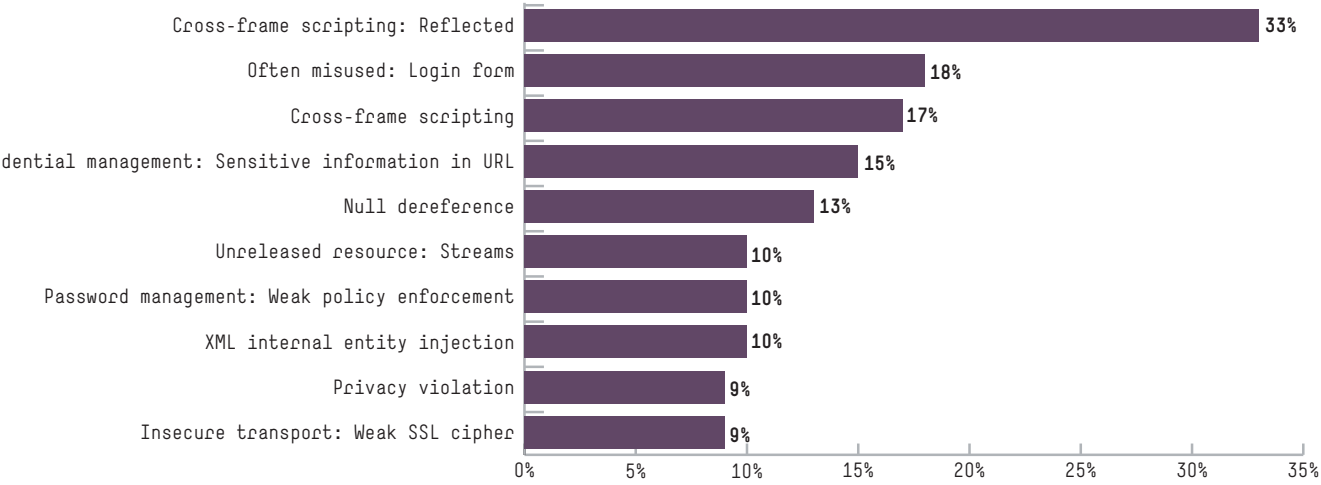


Figure 22. Top 10 critical vulnerabilities noted in Web applications in 2014

While there is no surprise that Cross-Site scripting: Reflected is the top critical concern affecting most Web applications, it is notable to see various authentication-related vulnerabilities in the top 10. Erroneous implementation or configuration of login forms, credentials, passwords, and secure transport may provide the right ingredients for privilege escalation.

The trend with server misconfigurations seems to be very similar to last year, and it

is the number-one issue across all analyzed applications within this category.

Looking deeper into the breakdown, access to unnecessary files and directories seems to dominate the misconfiguration-related issues. In comparison to the top 10 common vulnerabilities shown in Figure 21, Unprotected File and Directory are the second and fourth most pervasive, affecting 46.86 percent and 45.63 percent of applications respectively.

Breakdown of the top five mobile application vulnerabilities

Let's discuss the more general or aggregated trends in vulnerabilities observed this year before we review more specific trends.

Mobile application vulnerabilities continue to evolve as the platforms become attractive targets for application developers. Due to the nature of mobile devices, their vulnerability surfaces share some attributes with traditional client/server applications and Web applications. However, the type

of information that is trusted on mobile devices creates some unique attack vectors as well. Mobile devices contain sensors and actuators of types not historically common in personal computers or servers, which collect and transmit private information about the user of the device. The list of sensors that can reveal sensitive information include cameras, microphones, accelerometers, gravity sensors, rotational vector sensors, gyroscopes, magnetometer, global positioning system (GPS) sensors, near-field communication (NFC), light sensors, M7 tracking chips, barometers, thermometers, pedometers, heart-rate monitors, and fingerprint sensors.

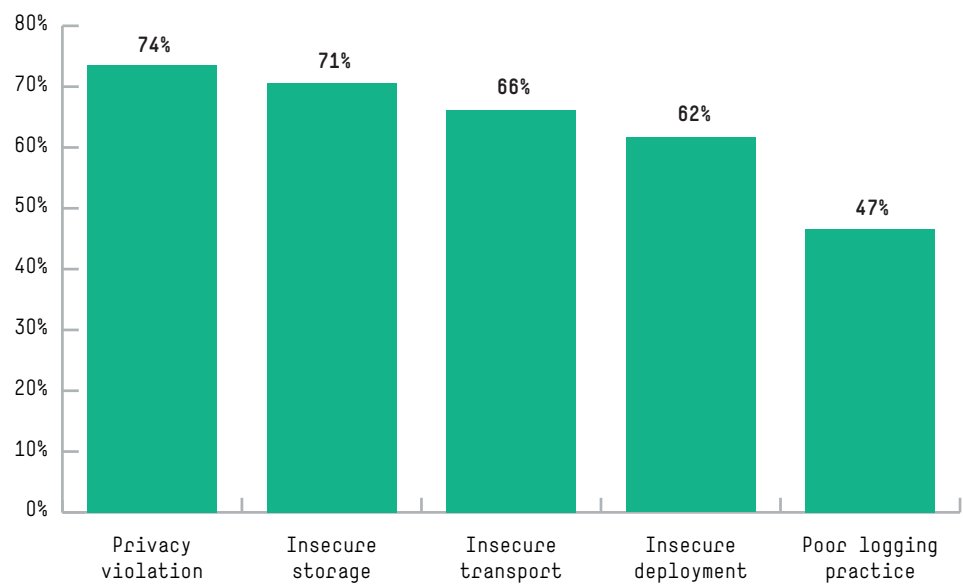


Figure 23. Top five mobile vulnerabilities noted in 2014, aggregated by category

In contrast to the top five aggregated vulnerabilities found in Web applications, mobile apps have four completely different categories, with only Privacy Violation in common. While Privacy Violation was the fourth most common vulnerability observed in the Web applications we analyzed (52.91 percent), it takes the top spot in mobile apps (73.54 percent). Mobile applications are unique in that they have access to a wealth of personal information from the array of sensors built into modern mobile devices. Furthermore, privacy violation weaknesses occurring on mobile devices can lead to the disclosure of location, sensitive images, data entered from the keyboard or displayed on the screen, and other personal information. In total, there were 15 specific privacy

violation categories observed in the sample population. The potential for disclosing geolocation information was the most common, found in 31 percent of all mobile applications in the study.

Part of securing sensitive information on mobile devices involves ensuring that the data is stored in such a way that it is adequately protected. Insecure Storage weaknesses, which take the second spot (70.63 percent), are introduced into mobile applications through either the misuse of APIs, which are provided to help ensure the protection of data using encryption schemes, or through their lack of use. On Android devices, this can take the form of data being written to external storage (including backup

storage) without the use of encryption, or of making data stored on internal storage world readable or writeable. For iOS devices, a data protection API is provided that uses hardware encryption to protect data; however, without proper understanding of the various levels of data protection and what they apply to, developers easily can make mistakes that leave the data vulnerable.

While smartphones can be used for viewing, manipulating, and storing local data, these devices also free users to interact with a world of interconnected resources from the convenience of their hands. Through communication protocols, both sensitive and benign data is shared between remote services and our devices. The third most common aggregate vulnerability category is Insecure Transport (66.14 percent), which identifies weaknesses caused by the use of communication protocols that do not protect sensitive data through encryption (such as HTTP instead of HTTPS) or that make use of less secure protocol options.

Taking the fourth spot, Insecure Deployment weaknesses were identified in 61.64 percent of the mobile applications studied. Insecure Deployment combines various deployment configurations, settings, and states that result

in unnecessary weaknesses. For mobile apps this may include not using features such as PlayReady DRM, not checking to determine if the app is running on a jailbroken device, or exhibiting properties that may indicate malicious intent.

Rounding out our top five, Poor Logging Practice weaknesses were found in 46.56 percent of the mobile applications. The use of standard output often indicates that the application is using standard output for debugging and logging rather than structured logging facilities such as android.util.Log for Android, or the use of NSLog in iOS. Because standard outputs, and logs, can be read by third parties, developers should always be cognizant of the risks associated with any sensitive information being written.

Top 10 mobile application vulnerabilities

As shown for Web applications, each category within the taxonomy can be further refined into more specific vulnerabilities. The following chart depicts the vulnerabilities observed with the highest frequency in the sample population of mobile applications.

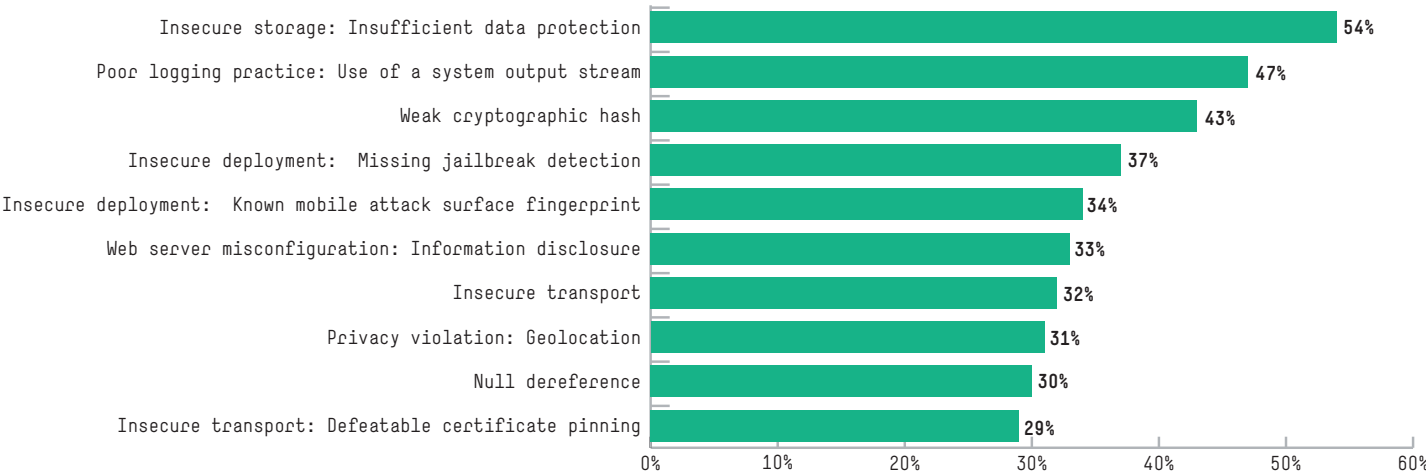


Figure 24. The 10 most common vulnerabilities noted in mobile apps in 2014

While Privacy Violation was the most prevalent aggregated category, Insecure Storage: Insufficient Data Protection (54 percent) was the most commonly observed vulnerability overall. In addition, all of the top 10 common vulnerabilities above are captured by the top five, except for Weak Cryptographic Hash (43 percent),

Web Server Misconfiguration: Information Disclosure (33 percent), and Null Dereference (30 percent). Most developers tend to pay attention to the most critical vulnerabilities. The following chart represents a subset of the vulnerabilities considered to pose considerable risk that were found in the mobile applications.

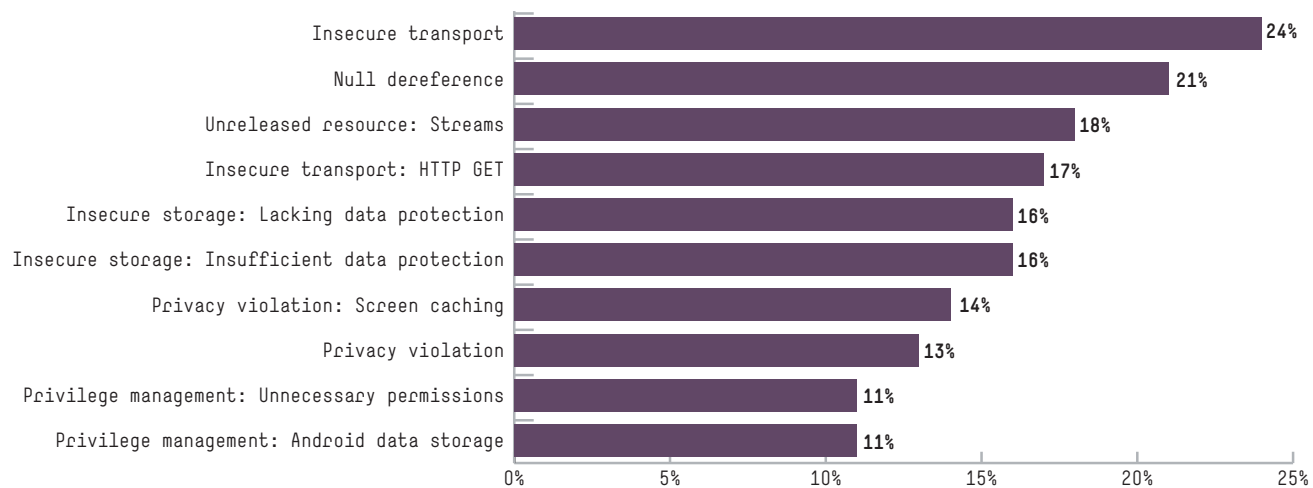


Figure 25. The 10 most critical vulnerabilities in mobile apps noted in 2014

Focusing on the vulnerabilities considered critical to mobile applications, the list becomes very different from the 10 most common vulnerabilities. Insecure Transport takes the top spot with 24 percent of applications exhibiting critical issues (compared to the 32 percent overall that had insecure transport issues). Interestingly, the second most commonly observed critical

vulnerability was Null Dereference (21 percent), which can lead to an application crash when executed. Furthermore, outside of the top five aggregated categories above, unreleased streams (18 percent) were common along with excessive (Unnecessary Permission, 11 percent) and risky (Android Data Storage, 11 percent) privilege settings.

Open source software dependencies

Organizations have relied on open-source components to build their software since the beginning, and more so since the late 90s when the Open Source Initiative (OSI) was formally established. However, little or no attention was paid to the increase in attack surfaces due to integrated open-source components. As mentioned earlier, manifestation of several critical vulnerabilities (such as Heartbleed, Shellshock, and Poodle) in prevalent libraries (such as OpenSSL and Bash) has brought renewed attention to software dependencies. There is an increased interest in accounting for these components in overall risk assessment for an

organization. The information in this section is intended to help organizations to really understand how and if they are likely to be affected by integrated open source components.

Approach

HPE Security Research analyzed Sonatype third-party library security data for over 100 randomly selected real-world Java applications provided by the HPE Fortify on Demand managed service. The reported data contains name, version, and a list of CVEs associated with open-source components referenced in each application. Furthermore, the data also contains the fraction of open source components referenced by each application.

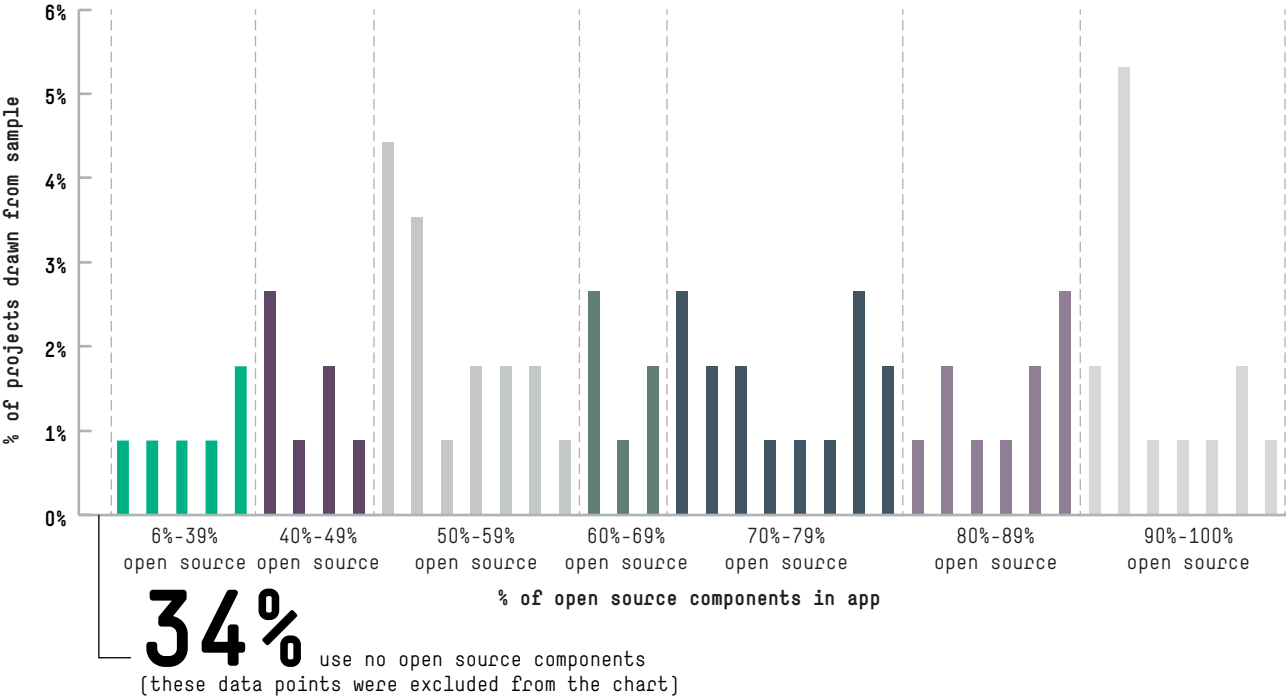


Figure 26. Distribution of open-source components referenced by application

The graph above shows the percentage of open-source components referenced in the projects we examined. We excluded the 34 percent of projects we examined that contained no open-source components, and ranged the remaining projects along the X axis, from just a bit of open-source presence on the left to nearly 100 percent open source at the far right. We noted that over 55 percent of the projects we examined were over 50 percent open source—that is, over 50 percent of their components are distributed as open source. Over 10 percent of the projects we examined were 90 percent open source or greater.

Issues inherited from open-source libraries

Next we investigated the type of issues that are inherited from open-source libraries. To better understand the type of risks associated with these CVEs from the data, we mapped each CVE to the HPE Software Security Taxonomy.

One hundred thirty-two unique CVEs were reported across 88 unique components; 228 unique components when different versions are considered. These were distributed across six kingdoms of categories related to problems in the code and one kingdom related to problems outside the code, such as environment. It is interesting to note the absence of code quality issues in the publicly disclosed list of CVEs. One perceived reason could be that most critical code-quality issues such as double-free or memory leak are related to C, as opposed to the Java-based libraries referenced in the projects we evaluated. Although such issues need to be addressed to ensure the desired functionality from the application, apparently they are not often reported to the National Vulnerability Database.

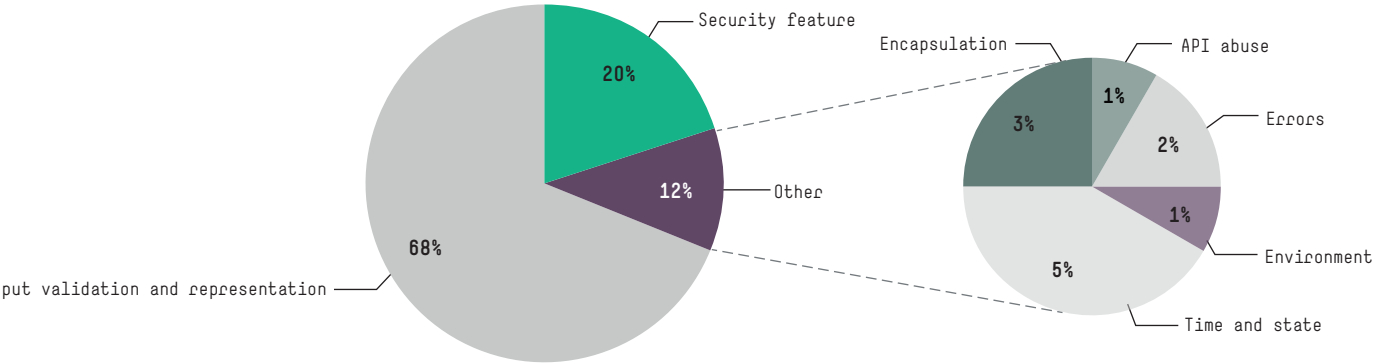


Figure 27. Unique CVE distribution across kingdoms

Sixty-eight percent of vulnerabilities are due to inadequate input validation. Within the Input Validation and Representation kingdom, Denial of Service and XML External

Entity Injection are the most frequently reported categories. All issues reported were spread across 33 categories and most were concentrated in the top 10.

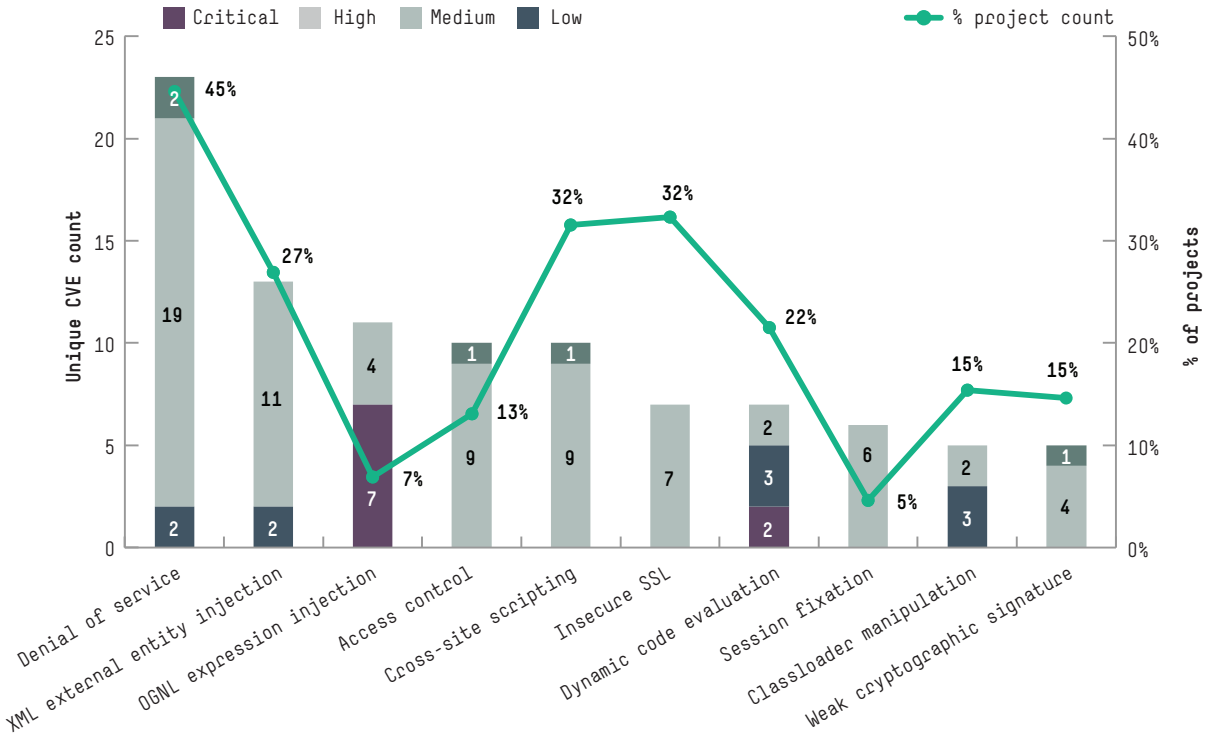


Figure 28. Top 10 vulnerability categories by unique CVE and severity distribution

Denial of service is the most commonly reported category; however, it is not the most critical. OGNL expression injection is the most commonly reported critical category; however, at 7 percent it affects only

a small fraction of applications reviewed. The majority of projects inherit denial of service, cross-site scripting, and insecure SSL issues from third-party libraries.

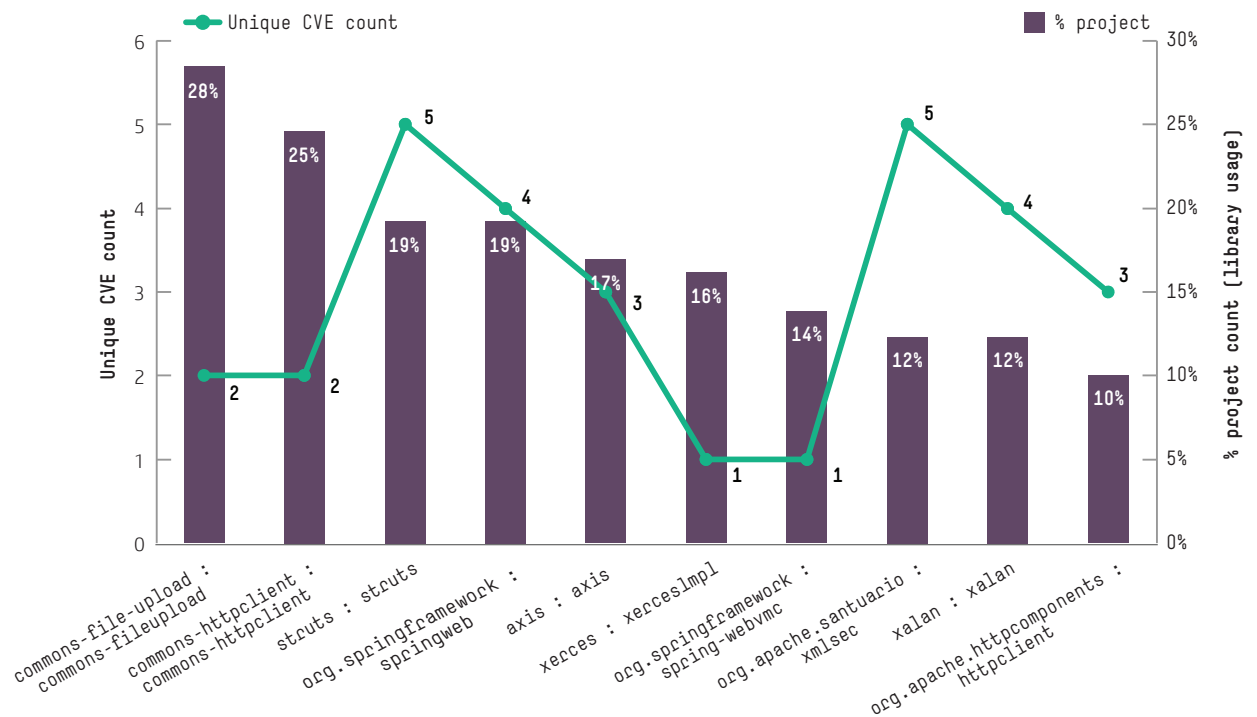


Figure 29. Top 10 popular libraries and unique CVE distribution

Popular libraries have on average three issues, while most have fewer than five issues.

Libraries with the most vulnerabilities are less popular. They also produce fewer releases.

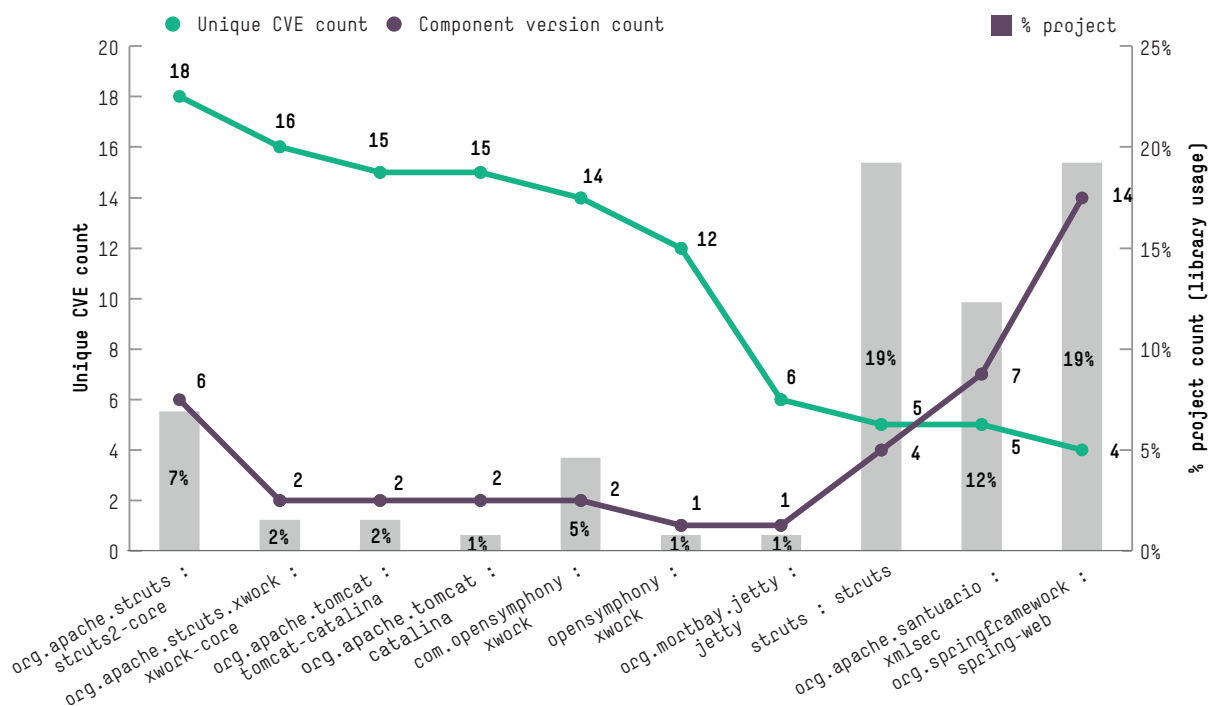


Figure 30. Top 10 vulnerable libraries by unique CVE and severity distribution

The Heartbleed effect

Our analysis in the section above shows that the use of open-source libraries is extensive. Two-thirds of all applications built today rely on open-source components. Once a component is integrated, vulnerabilities in those components are inherited in the software. The new software can in turn be referenced to further, creating a multiplicity factor for pre-existing security issues. The Heartbleed bug is an example of such a scenario.

To analyze this situation, we selected a group of components that integrate the OpenSSL library and could be present in the server environment in some form. Researchers at the University of Michigan performed a detailed analysis¹⁴⁷ of the Heartbleed bug, which is CVE-2014-0160. As part of their cyber security research, they surveyed components that integrated the OpenSSL library and were affected by Heartbleed bug. The list of components we have selected is an extension to the list provided in their research; we added three

additional categories (appliance, operating system, and programming language/dev environment) that are an essential part of an IT environment. We then examined data on vendor response and remediation efforts around the Heartbleed vulnerability. We compared vendors' response to Heartbleed with response to an OpenSSL vulnerability (CVE-2013-4353) that was reported before Heartbleed, and another one (CVE-2014-0224) that was reported after Heartbleed.

The objective of this comparison was to analyze the response to various OpenSSL issues by dependent vendors in 2014.

Our research looked at response time for various vendors based on publicly available information. It should be noted that some vendors might have notified their users privately. In addition, "No information" could mean that either the vendor ignored the issue and thus failed to notify users or that the vendor did not depend on a vulnerable version of OpenSSL and hence was not affected.

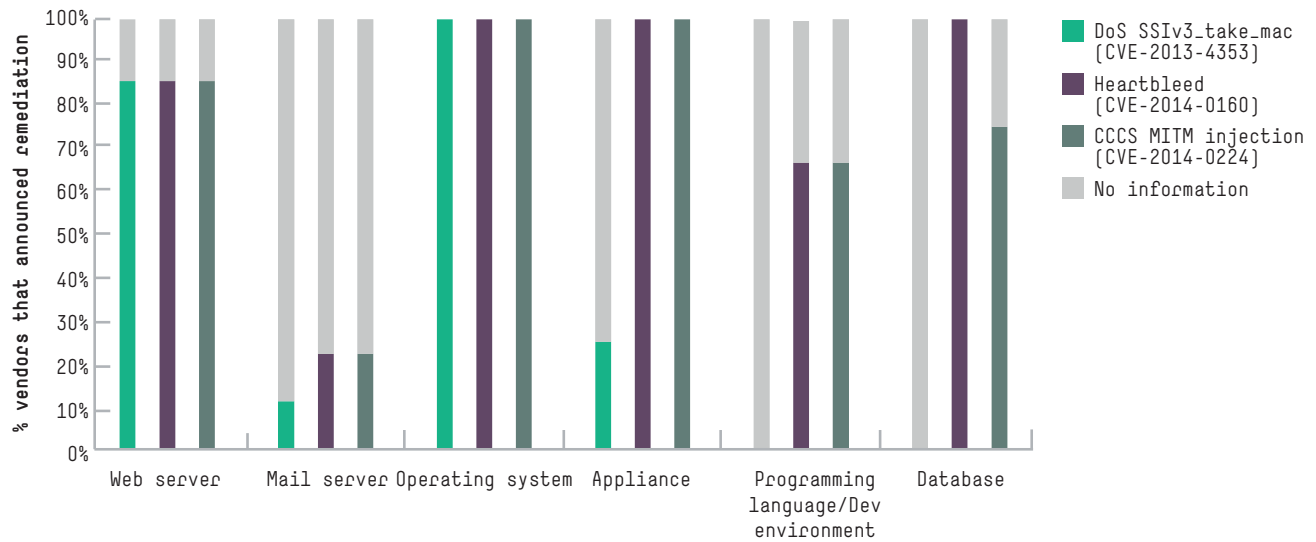


Figure 31. Dependent vendor remediation trend for OpenSSL security issues

A significant number of vendors did not openly acknowledge and remediate weaknesses inherited from OpenSSL before Heartbleed, whereas the trend seemed to change post-Heartbleed.

Fewer disclosures were noted for mail server libraries using OpenSSL. Most of these libraries rely on dynamic binding of OpenSSL, which shifted the onus to users to keep their installed OpenSSL packages updated.

The “Heartbleed effect” has resulted in not only more vendors opting to remediate OpenSSL issues, but also in improved response time to remediate these issues. The figure below shows the maximum number of days taken to remediate OpenSSL issues, as measured by notifications made public. We have averaged across all three OpenSSL issues, adjusting for the corresponding type of vendor.

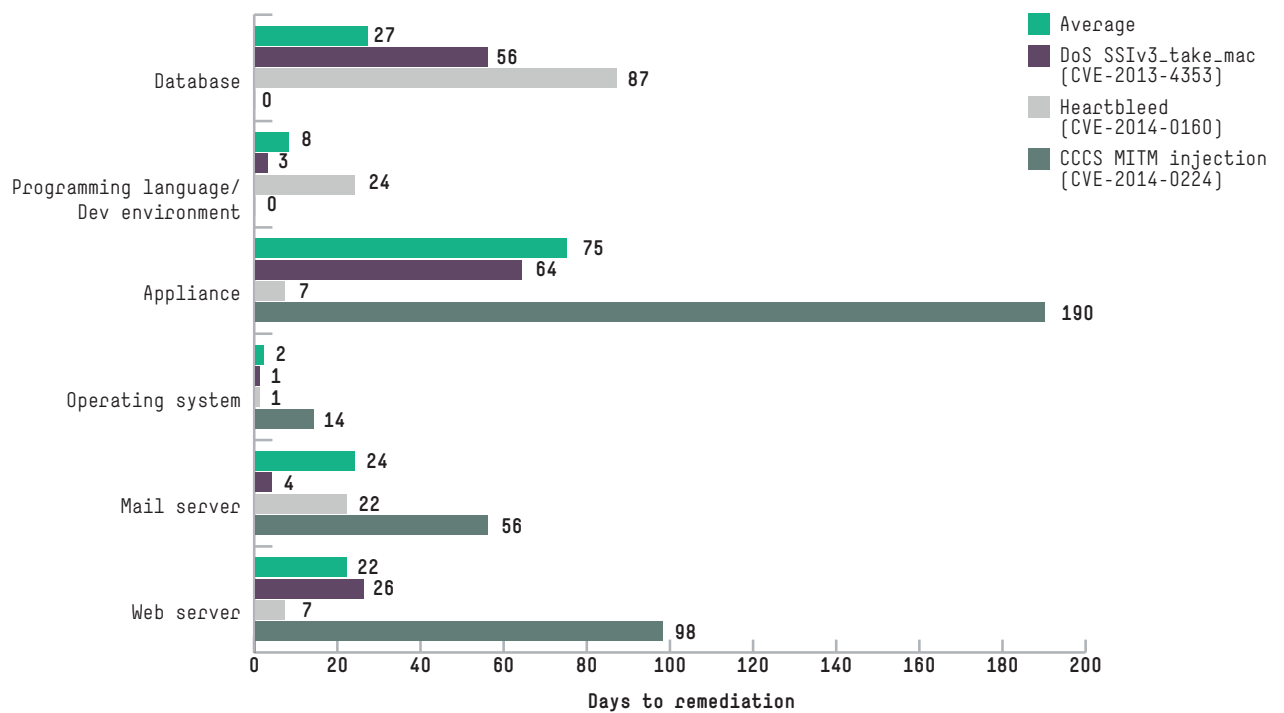


Figure 32. Maximum days to announce remediation by category

Based on the data above, operating systems have stellar performance when it comes to remediating dependent vulnerabilities. This could be because they are most critical to providing indirect dependence to installed applications via dynamic binding. A slow response from appliances can be attributed to various reasons:

- Often users do not have information about software components that run on their appliances.
- Appliances may not be embedded further to create a multi-level dependency, reducing user awareness of the vulnerability and demanding a fix from appliance manufacturer.
- Appliances may not be updated easily.
- While the worst maximum fix times show up in the Appliance category, in that category that was the only appliance that actually pushed a fix for a pre-Heartbleed OpenSSL issue.

Most vendors do not maintain and provide a list of all third-party components that their software depends on, resulting in an incomplete risk profile for users of the components. Furthermore, most licenses, while requiring embedding software to disclose usage of the library, don't impose version number disclosure on the vendor. We as the software industry need to make including a list of third-party components along with version number in software specification an essential part of the software disclosure for adequate risk assessment for organizations.

Remediation of static issues

For the purpose of analyzing remediation patterns of vulnerabilities in Web applications, a subset of the overall sample set in previous analyses was taken. This selection was done based on the following criteria:

- Only automated scans that performed static code analysis were considered.
- All vulnerabilities considered were detected and fixed within the past year.

The typical process between the first scan on an application and the following remediation scan is as follows.

1. User requests a scan of an application.
2. Automated static code analysis of the application is performed.
3. HPE auditors audit the results of the scan.
4. HPE operators publish the audited results.
5. User reviews the results. The user may request a re-audit of certain issues based on the business criticality of certain applications.
6. User triages the results and assigns to developers. In some cases, developers may directly triage the issues.
7. Developers fix the issues.
8. QA validates the fix. This may involve several iterations depending on the quality of the fix.
9. The organization may submit the new version immediately or batch many fixes together before sending a newer version for assessment.
10. HPE performs the remediation scan and validates the fix. The entire cycle may be repeated for issues that weren't fixed in this round.

In order to better understand when most of the vulnerabilities in a given kingdom or severity were fixed, the statistical median value was considered to plot the graph below:

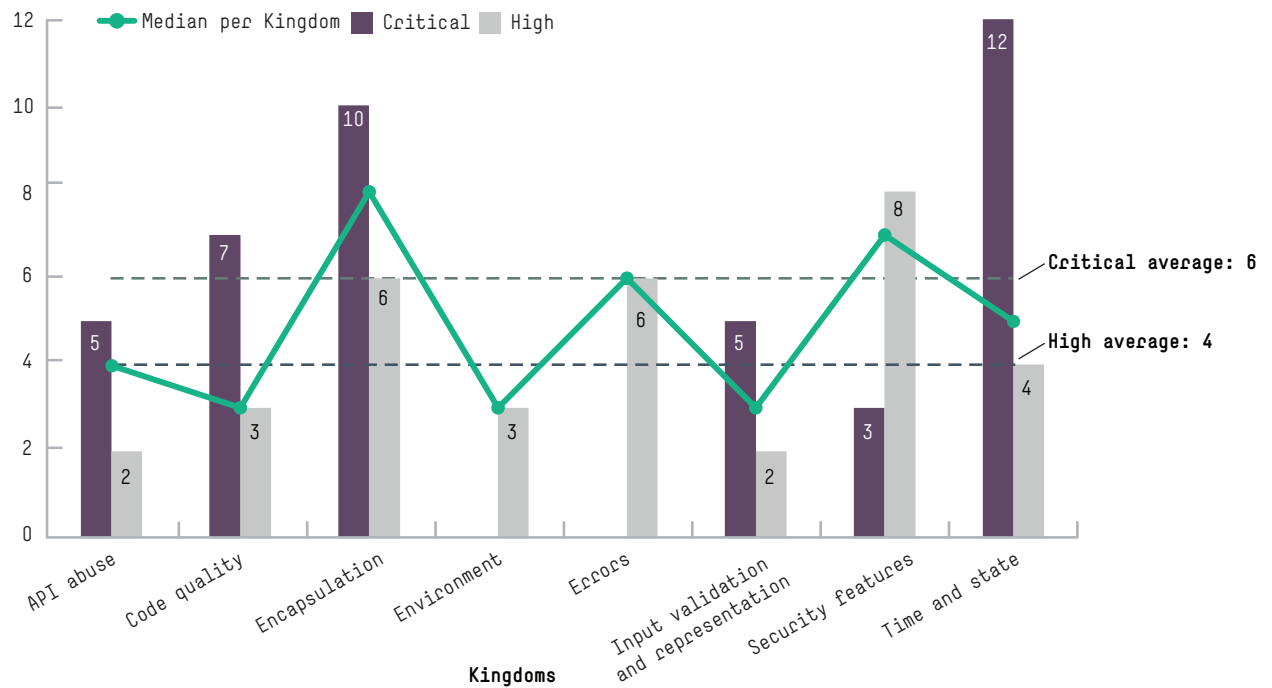


Figure 33. Median number of scans to remediate critical vulnerabilities

The graph represents the median number of scans to fix a critical or high vulnerability in a given kingdom. As an organization, it would be interesting to verify the frequency of static scans done on an application during its lifecycle, and in turn to calculate the number of days it takes to verify a fix for that application. This would also give an idea of the risk assumed by the organization during that period (this only applies to applications in a production environment).

We were hoping that critical vulnerabilities would be the fastest to fix. Interestingly, this was not always the case. One possible reason could be that most organizations tend to fix and verify all critical and high vulnerabilities first. Hence, the developers could be prioritizing their tasks from a single bucket based on the ease of completing the task, rather than the severity of the issue.

The overall data (not represented in the graph above) also showed that environment-related issues with lower severities were usually verified after a very long time—as much as 240 scans later. This might be acceptable, because typical fixes to environment issues involve tweaking the server or application configuration toward the end of a release to meet the security standards of a production environment (while the application is developed in a test environment). It could also indicate that some organizations may perform multiple scans within short intervals, or have long development periods between releases.

While issues generally get fixed after anywhere between two to 12 scans, not all of them get fixed. This could depend on the sensitivity of the application and the risk appetite of each organization. HPSR was interested to see if there were any patterns that stood out while looking at the whole picture, agnostic of the application and the organization. Below is the chart that provides this data.

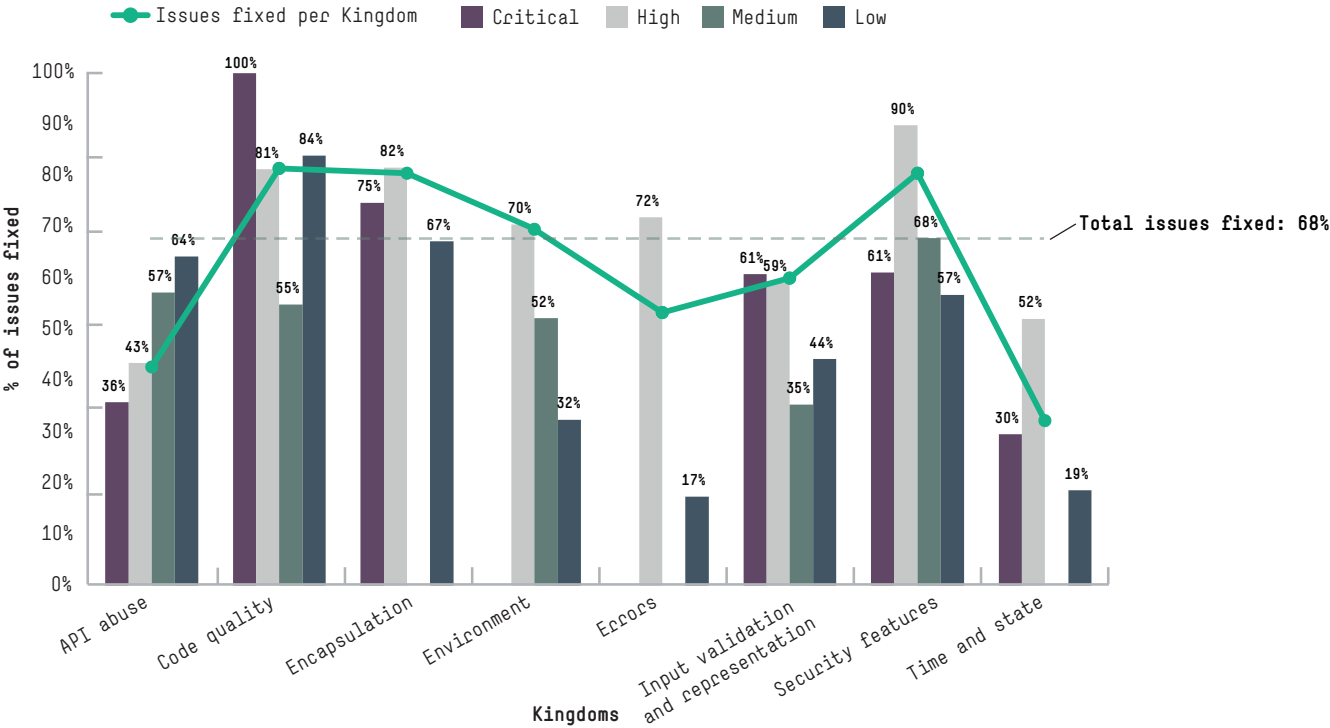



Figure 34. Issues fixed by kingdom; higher numbers are better



Ironically, misused **security** features continue to be the trouble area for both Web and mobile apps.

In a typical security development lifecycle, it is recommended that corporations have practices in place for identifying weaknesses in their applications. As part of that lifecycle, it is the responsibility of the team to verify that known vulnerabilities and any close variants have been mitigated. In studying the data from the past year, we have observed that for those applications that are following a security development lifecycle, where applications are analyzed more than once, over 68 percent of vulnerabilities are being resolved.

Overall, the percentage of issues fixed in the Security Features, Encapsulation, and Code Quality kingdoms are relatively high. It is also interesting to note that in the Input Validation and Representation kingdom, where the more well-known vulnerabilities such as cross-site scripting and SQL injection reside, the percentage of critical and high issues fixed are similar. This could imply that such injection issues are prioritized and fixed together during implementation.

Conclusion

Knowledge is power; being aware of the specific circumstances that give rise to vulnerabilities lets security practitioners address their root causes—even, if healthy secure development practices are followed, before they are committed to code and released to an unsuspecting world.

For a snapshot of the state of application security in 2014, we analyzed a sample set of security audits performed by HPE Fortify on Demand on 378 mobile apps, 6,504 Web apps, and 138 Sonatype reports from 113 projects. These audits include results from static, dynamic, and manual analysis. All identified issues were classified according to the HPE Software Security Taxonomy (originally the “Seven Pernicious Kingdoms”), updated and refined in mid-2014 and under continued expansion in 2015.

The good news, if there is good news to be had, is that the uproar around such high-profile incidents as Heartbleed and Shellshock may yet lead software developers and architects to tackle security issues, particularly those in foundational or legacy code, more effectively. We saw that repeated scans lead to better software, and that the open-source development community seems to be selecting for healthier, better componentry—components with more security issues simply aren’t used as often by other developers.

That said, vulnerabilities are still pervasive. Sorting the vulnerabilities we discovered into the categories of our taxonomy, we found every indication that more Web and mobile apps contained discoverable vulnerabilities. Ironically, misused security features continue to be the trouble area for both Web and mobile apps. There were a few differences endemic to each type of application; the nature of mobile apps means they’re particularly prone to code quality issues rarely found in Web apps, while Web apps suffer from the types of problems covered in the errors section of our taxonomy.

Progress is possible. Despite the rise in detected vulnerabilities, the very fact that they are daylighted means that they can be analyzed and addressed. Our research indicates that critical-class vulnerabilities are taken seriously and given patch-development priority—not a surprising find, when one remembers that post-release patching remains part of a healthy development lifecycle, but perhaps a sign that smart practices truly do take hold and crowd out lesser habits in the security ecosystem.



Summary

In a world where more and more people and devices connect to the Internet, greater focus must be placed on security and privacy. The past year has seen the manifestation of several vulnerabilities that gathered a storm of media attention. Network defenders should use the information in this report to better understand the threat landscape, and best deploy resources to minimize security risk. While the Internet has brought global connectivity to millions, the darker side of the Internet is pervasive and influential. Our report shows the machinations and maneuvers of criminals and state-sponsored operators in the cyber underground have significant and lasting effects on the security of the greater Internet and our greater societies. Cyber crime comes in many flavors, but it remains vastly driven by financial interests. Looking into nation-state-sponsored cyber activity highlights the many levels at which cyber operations and state-sanctioned activity can occur, and demonstrates how malware and the tools and techniques of cyber criminals can be utilized in different ways to accomplish different goals—such as stifling protest or targeting opposing state interests, as well as perpetrating fraud or stealing intellectual property.

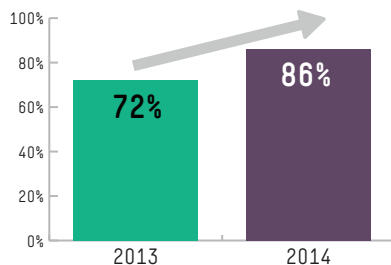
Responses to this long-recognized threat and international cooperation to address these attacks are improving and continue to gain momentum. Other nations also pose significant threats in our globally connected world. Iran continues to develop its cyber capabilities and views hacker groups as a force multiplier to be used to target Western entities, particularly corporations and government entities. North Korea has continued its tradition of asymmetric warfare in the age of the Internet, with a remarkable commitment to developing cyber warfare capabilities even as it copes with aging infrastructure. As far as financially motivated attacks go, while systems such as “chip and pin” are likely to prove useful, as particular points in financial processes get hardened, other points become more attractive to attackers, and as technology develops to improve the security of systems, it also conversely develops to make particular targets increasingly accessible. We expect escalations in this area to continue.

In the face of increasing threats, software vendors continue to make it more difficult for attackers with the implementation of security mitigations. However, these mitigations are not enough when they are built on inherently vulnerable legacy code. On multiple occasions in 2014, high-profile vulnerabilities were discovered that left enterprises scrambling to deploy patches and clean up compromised machines. Watching the industry respond to the Heartbleed vulnerability highlighted how unprepared we were for this type of disclosure. Due to the severity and active exploitation of the vulnerability, corporations were forced to respond quickly, and to patch servers that were not routinely patched. The issue existed in an application library that did not have a clear update path; enterprises did not have a solid understanding of which applications were using this library and where it was located inside of their networks, further complicating efforts.

Discovery of information disclosure vulnerabilities such as Heartbleed shows why information disclosure vulnerabilities are highly valued by the exploitation community. These issues can also be used in conjunction with remote code execution vulnerabilities to bypass modern exploit mitigations. Heartbleed was a nice demonstration of a highly controllable information disclosure vulnerability due to a buffer over-read. Vulnerabilities found in legacy code were also a significant factor in 2014, with flaws in Adobe Flash Player and RTF parsing in Microsoft Office being prime examples. In each case, either legacy or deprecated code was at fault. As the quality of exploits continue to improve, they reveal a deep understanding of the nature of the vulnerability and the internals of the target applications.

The year 2014 saw the growth of malware continue to increase, with the number of newly created malware samples doubling year over year. It was also a significant year for mobile malware, as 2014 was the year when mobile malware finally stopped being seen as just a novelty. While the majority of Android malware discovered in 2014 was found outside of the Google Play market, there have been instances when malware was placed there by maliciously created developer accounts. Ransomware was also a key theme throughout this past year as attackers continued to exploit a

Jump in issues concerning the realm of security features from 2013 to 2014



business model where users' data is held for ransom by malware often using asymmetric encryption algorithms. Perhaps the most notable ransomware is CryptoLocker, which appeared at the end of 2013 and caused significant damage prior to an FBI-led takedown. CryptoLocker has spawned a number of copycats, with CryptoWall the most well-known.

The threat from malware continues to rise as the attacks on Target and Home Depot highlighted the risk from point-of-sale (POS) devices. By looking at the POS malware responsible for these breaches, we have discovered that this type of malware is being actively developed by multiple groups. Our investigation uncovered ongoing development, increasing sophistication, and a divergent code base in current POS malware. Significantly, these programs were built by people who knew the targeted environments. The attackers using them had gathered initial intelligence from the targeted systems before creating custom malware to exploit them. This highlights the planned nature of these attacks and reminds us that attackers are increasingly playing the long game. Enterprises must be able to monitor their networks and systems in a manner that allows them to discover malicious intelligence gathering and reconnaissance activities that may herald an approaching attack.

There appears to be growing consumer awareness about privacy issues at the Internet of Things (IoT) level, whether that's concern that one's television or thermostat represents a security and privacy risk or something else. The mass theft and online posting of private photos from hundreds of celebrities blurred the line between security and privacy, as what first appeared to be an intrusion shaped up to be a bad combination of poor password choice and inadvertent saving of data to the cloud. IoT is much more than a buzzword—it's a new paradigm that brings ubiquitous computing and its security implications closer to the average person. Attacks could involve various layers of the device infrastructure. This could include applications running on smartphones or tablets, cloud services (including firmware and network service stacks on Wi-Fi modules) as well as the firmware and application layers on the host processor. Various vectors of propagation could also be used, including compromising update files and exploiting network and host processor

communication layer vulnerabilities, as well as possible vulnerabilities in cloud service infrastructures and smart device applications.

In general, 2014 saw more Web applications containing vulnerabilities in each security kingdom than they did last year. Issues concerning the realm of Security Features—including access control, privacy violation, password management, insecure transport, and security of cryptographic primitives—jumped from 72 percent in 2013 to 86 percent in 2014. This statistic is consistent with the recent rash of privacy and confidentiality breaches, ranging from stolen personal data to credit card numbers to personal health information. The Security Features kingdom also greatly affects mobile platforms, as the percentage of mobile applications that contain security features issues (97 percent) is higher than that of Web applications (86 percent). Looking at data analysis of over 6,500 applications, increases in vulnerability distribution by security kingdom grew in all but one category (and that category decreased by just 1 percent).

While the threat from the Internet itself can take on an increasingly global scale, a worldwide network of security researchers stands ready to help the software industry secure its code. The HPE Zero Day Initiative is the world's largest vendor-agnostic bug bounty program, with almost 10 years' experience coordinating vulnerability disclosure. Over its history, it has grown to a network of over 3,000 independent researchers working to expose and remediate weaknesses in the world's most popular software. Over the past two years, researchers representing several new regions (including Germany, South Korea, China, and the Russian Federation) popped up with high submission rates and quality technical analysis. Researchers in these countries are not only focusing on vulnerability discovery but also on innovative exploitation techniques.

Looking ahead, technology continues to enhance our world in numerous ways, and with that comes the challenge of maintaining security and privacy throughout our digital lives. However, with increased cooperation and a thorough understanding of the imminent threats, we can continue to increase both physical and intellectual costs an attacker must spend to successfully exploit a system.

Authors and contributors

The HPE Cyber Risk Report 2015 is an annual collaboration.

Authors	Contributors
Dustin Childs	Joe Gura
Art Gilliland	Ted Ross
Brian Gorenc	Jewel Timpe
Heather Goudey	ReversingLabs
Angela Gunn	Sonatype Inc.
Alexander Hoole	
Jason Lancaster	
Sasi Siddharth Muthurajan	
Jeong Wook Oh	
Yekaterina Tsipenyuk O'Neil	
John Park	
Oleg Petrovsky	
Joe Sechman	
Nidhi Shah	
Tim Sotack	
Vanja Svajcer	

Glossary

29A

A notorious virus writing group established in the mid-1990s that published an e-zine on virus writing of the same name. “29A” is the hexadecimal representation of the number 666.

AES (Advanced Encryption Standard)

A cryptographic standard, based on the Rijndael cipher, specified by the National Institute of Standards and Technology (NIST) in the United States for the encryption of electronic data. AES is a symmetric key algorithm and replaced the previous standard, DES (Data Encryption Standard).

ASLR (Address Space Layout Randomization)

A security mechanism where the locations of important elements of a program in memory are randomized in order to make them harder for an attacker to find and utilize. This increases the difficulty for the attacker to perform particular types of exploit that rely on jumping to particular address areas of memory.

Anonymous

A loosely associated and informal group of hackers that participate cooperatively in various forms of protest online. Types of protest have included Denial of Service (DoS) attacks and website defacements against various entities, including government and commercial organizations, and protesting against a broad range of different political and social issues from digital rights management and anti-piracy to revenge porn.

API (application programming interface)

A set of tools and resources that provide various functions developers can utilize when creating software.

APT1 (Advanced Persistent Threat 1)

A Chinese cyber espionage group tracked

and reported on by Mandiant. Mandiant reports that this group has been in operation at least since 2006 and has been involved in a number of operations to steal terabytes of sensitive and confidential data, including valuable intellectual property from dozens of organizations in various industries.

Blackshades RAT

A remote access Trojan (RAT) that allows unauthorized access and control of computers on the Windows platform. This particular RAT came to be quite notorious and was the subject of raids by the FBI on over 100 people in 2014 that led to dozens of arrests.

BrowserLock

A type of malware known as ransomware. Ransomware is malicious software that locks a user's computer in some way and then demands a ransom in order for service to be restored. As the name suggests, this malware locks the affected user's Web browser and holds it to ransom.

Buffer overrun/overflow

A buffer overflow is a type of vulnerability that arises when a program writes an excessive amount of data to the buffer, exceeding the capacity of the buffer and then overwriting adjacent memory. This type of vulnerability may be exploited to crash programs or, with the correct manipulation by a skilled attacker, used to execute arbitrary code on a targeted computer. Buffer vulnerabilities can be avoided by the use of bounds checking, which checks the capacity for inputs before they are written.

Bytecode

A form of instruction set designed for efficient execution by a software interpreter. Bytecodes are compact numeric codes, constants, and references (such as numeric addresses), which encode the result of parsing and semantic analysis of things like type, scope, and nesting depths of program objects.

C&C: See Command and Control

Cabir

The first malware written for mobile devices, first detected in 2004. Cabir was a worm that targeted the Nokia Series 60 Symbian platform and spread via the Bluetooth® OBEX protocol to other phones. It was proof-of-concept malware created by the virus writing group 29A.

Clickjacking

A technique used to make a user take an action of an attacker's choice by clicking on part of a webpage. While the user may believe they are clicking on something innocuous, in effect they are performing an action that is required by the attacker in order to achieve their goal—for example, by taking the action of clicking on a particular object on a page, a user may inadvertently execute a script or comply with a request to grant a particular type of risky activity.

Command and control (C&C)

As with many terms used in computer security, this term has been borrowed from the military. Similar to the military use of the term it means a method of exercising authority over resources, for example, a commanding officer commanding his troops. This term is often used in the context of malware and botnets in particular, where a structure is set up to command and control many compromised computers from either a centralized, or in some cases, decentralized position. A centralized command and control structure might be a single server that compromised computers connect to in order to receive commands. A decentralized command and control structure could be where compromised computers connect to a peer-to-peer network, where commands are spread through the network from many possible nodes. Command and control may also be known as C2.

Command injection

Command injection occurs when an attacker is able to pass unsafe data to a system shell via a vulnerable application so that the unsafe data is then executed on the targeted system. The result therefore of a successful command injection attack is the

execution of arbitrary attacker-supplied code on a targeted system. The risk of command injection attacks can be mitigated by appropriate input checking and validation.

Cross-frame scripting

A form of cross-site scripting attack, in which an attacker exploits a vulnerability in a Web browser in order to load malicious third-party content that they control in the frame of a webpage on another site. This attack may allow an attacker to steal sensitive information, such as login details, that may be input into the frame because the targeted user believes the request for login details came from the legitimate site.

Cross-site scripting

An attack that occurs when an attacker exploits a vulnerability in Web applications in order to inject malicious code into client-side code that is delivered from a compromised website to an unsuspecting user. This code that is delivered to the user is trusted, and hence executed, as it appears to come from a legitimate source. These types of attack occur due to insufficient checking and validation of user-supplier input. Attackers may use this type of attack in order to bypass access controls or steal sensitive data.

CryptoLocker

A type of malware known as ransomware. Ransomware is malicious software that locks a user's computer in some way and then demands a ransom in order for service to be restored. In the case of CryptoLocker, as the name suggests, users' files are encrypted using an asymmetric encryption algorithm. A ransom is then demanded from affected users in order to decrypt and therefore restore their files. CryptoLocker was first discovered in the wild by researchers in 2013. It was reported to have been propagated via the Gameover Zeus botnet. The "success" of ransomware such as CryptoLocker has spawned many copycat programs. The best way to avoid being the victim of ransomware is to ensure that regular backups of your files are created and maintained, and then stored in an unrelated system; thus, if files are encrypted, they can be restored from backup.

DEP (data execution prevention)

A security measure used by modern operating systems that is intended to prevent the running of malicious code on an affected system. It operates by marking areas of memory as either executable or non-executable and raises exceptions when code attempts to run from areas that are deemed non-executable.

Decebal

Decebal is named after the Romanian Emperor “Decebalus,” the last king of Dacia, and is a VBScript-based POS malware created by Romanian cyber criminals. Similar to Dexter, it also scrapes memory for track 2 credit card information and sends it back to C&C servers. Decebal was observed in the wild in early 2014, although various samples since captured appear to date it back to mid-2013. It is written in Microsoft Visual Basic, which is an event-driven programming language that follows the COM programming model. This makes it ideal for the malware to target point of sale systems that run OPOS systems.

Dexter

POS malware that was first discovered in the wild in 2012. Dexter uses RAM-scraping techniques in order to capture credit card details from track 2 data captured on POS systems.

Directory traversal

A directory traversal occurs when an attacker is able to access areas of a system that are not intended to be publically accessible.

This may occur due to inappropriate or insufficient checking of user-supplied data that contains characters that may be interpreted to traverse paths to a parent directory of the targeted system.

DoS (denial of service)

A condition that occurs when the resources of a system are exhausted in some manner that causes the system to stop responding.

Exfiltration

In the context of computer security, exfiltration refers to the removal of valuable captured data from a compromised system.

This is often performed as a separate step, by separate tools from those that may be used in order to first compromise the targeted system and then to capture and store data. In order for data to be exfiltrated, for example, it may need to be moved to a computer that has access to the Internet.

Exploit

Code written expressly to take advantage of the security gap created by a particular vulnerability in order to deliver a malicious payload. They may be targeted at specific organizations or used en masse in order to compromise as many hosts as possible. Delivery mechanisms utilize many different technologies and vehicles and often contain a social engineering element—effectively an exploit against vulnerabilities in human nature in order to make the victim take a particular action of the attacker’s choosing.

External leakage

An external information leak occurs when system data or debugging information leaves the program to a remote machine via a socket or network connection.

Fakeinst

Mobile malware that targets the Android platform. The Fakeinst family of malware masquerades as helpful installers that are used to install other useful applications. However, once executed they attempt to send SMS messages to premium-rate services, leaving affected users with unexpectedly high phone bills.

Foundational technologies

A technology that provides infrastructure for another technology. For example, a document management system could require a relational database as a foundational technology.

Frame-sniffing

These attacks occur when an attacker uses a hidden HTML frame to load a target website inside an attacker-controlled webpage. By doing this, attackers can access sensitive information about the content of framed pages.

Fuzzing

This is an automated vulnerability research technique that involves using a fuzzer (a fuzzing tool) to inject malformed data to an application in order to attempt to cause it to crash. This technique is used to uncover possible areas of weakness or vulnerability within the application for further research and testing.

Gameover Zeus

A notorious botnet consisting of peer-to-peer variants of the Zeus malware family (also known as Win32.Zbot). By using peer-to-peer, this botnet used a decentralized command-and-control infrastructure, thus avoiding the single point of failure of more centralized command and control structures and making the botnet more resilient to takedown. The Zeus malware family is associated with the sophisticated theft of online banking credentials; however, the botnet may have been used to carry out other malicious activities such as propagating additional malware, sending spam, and carrying out distributed denial of service (DDOS) attacks. In mid-2014, U.S. authorities brought down a large botnet of Gameover Zeus variants and caused significant disruption to the botnet's malicious activities.

Gepaw

Malware written for the Android platform that targets online banking users. One of the first Android malware samples to be purposefully installed by a Windows component if an Android device was connected to the desktop.

Heap spray

Heap spraying is a technique used by attackers to assist in use-after-free exploitation by decreasing entropy in the address space. It is not itself an exploit method; instead, it aids attackers by making freed memory space more orderly and predictable. Heap spraying consists of forcing repeated allocations in an attempt to reclaim the freed buffer and to introduce some usable order to the freed space.

Heartbleed

A flaw, discovered in 2014, that allowed for unauthenticated remote attackers to disclose the memory of applications that use

a vulnerable version of OpenSSL. Successful attacks could result in the disclosure of SSL private keys, usernames/passwords, and session tokens.

Information disclosure vulnerability

A vulnerability that results in the unauthorized disclosure of information from a system. The effect of this type of vulnerability varies according to the nature of the information that may be disclosed. Such information could be used in order to conduct reconnaissance before a targeted attack, or could be the end game of the attack. The Heartbleed vulnerability was an information disclosure vulnerability that allowed for unauthenticated remote attackers to disclose the memory of applications that use a vulnerable version of OpenSSL.

Juche

A North Korean political ideology that proposes that the future independence of the North Korean state is determined by the agency of its people. This principle (simplified here) is used to drive and promote policy in North Korea.

Keylogging

The act of capturing keyboard events, such as keystrokes, in a log file that can later be exfiltrated by an attacker. This type of function may be used to capture sensitive information such as login details for various systems.

Legacy code

Code within a software program inherited from a previous version of the program.

Middleware

The software that sits between layers of other software to make the layers below and on the side work with each other. It is software invisible to the user that takes two or more different applications and makes them work seamlessly together.

MoneyPak

An online payment system, often used by the perpetrators of ransomware and rogue antivirus software, to gather ransom and extortion payments from victims.

Mozart

RAM-scraping POS malware that was used in the Home Depot breach in order to steal the details of over 56 million debit and credit cards.

mTANs

A one-time password used by some online banking providers in order to authorize transactions.

Private keys

Within a public key cryptographic system, a private key is one only known by its owner. Also, a term used to represent symmetric key encryption, which is encryption based on a shared secret between two or more entities communicating.

RAM scraping

RAM scraping occurs when the malware enumerates the processes and virtual memory space of the target machine looking for track 1 and track 2 data.

Ransomware

Ransomware is malicious software that locks a user's computer in some way and then demands a ransom in order for service to be restored. This locking may include encrypting the user's files in some way and then demanding payment for the decryption key, or it could be more simplistic and only rely on giving the user the impression that their computer is locked (even though it may be easily recoverable).

Remote code execution (RCE) vulnerability

A vulnerability that allows attackers to execute their own code on a target system. Depending on the vulnerability used, the RCE may be executed with either user- or system-level permissions.

ROP (return oriented programming)

An exploit technique that allows an attacker to execute code while bypassing certain types of defense-in-depth measures, such as ASLR.

Segfault

A segmentation fault (segfault) occurs when a program attempts to access a memory location that the program is not allowed to access. Segfaults may also occur when a program attempts to access a memory location through a method that is not allowed. An example of this would be a program attempting to write to memory marked as read-only.

Shellcode

A small piece of code used as the payload during the exploitation of a vulnerability. While these types of payloads typically start from a command shell, any code that performs a similar function is generically referred to as shellcode.

Shellshock

A family of security vulnerabilities in the UNIX® Bash shell, first disclosed in September 2014.

Songun

A North Korean social ideology that prioritizes the needs and preferences of the Korean People's Army in affairs of state and in allocation of resources.

Spaghetti code

A pejorative term for software that has an overly complex and tangled control structure. Code of this type is named because the program flow is conceptually like a bowl of spaghetti (e.g., twisted and tangled).

TOR

Free software designed to allow users to enable online anonymity and resist censorship. By directing traffic through thousands of relays, TOR (The Onion Router) conceals a user's location and network usage from those attempting to conduct network monitoring or traffic analysis.

Track 1 and 2 data

Track 1 and track 2 data are stored on the magnetic strip on the back of a credit card. The strip includes the account number associated with the card, its expiration date, and additional details that determine how and which transactions will be processed.

Trojan

Malicious software that, unlike worms or viruses, is unable to spread of its own accord. There are many different types of Trojans that are used in conjunction with other types of malware in order to perpetrate computer crime. One of the most notorious types is a remote access Trojan (RAT) that can be used by a remote attacker to access and control a victim's computer.

Use-after-free

A use-after-free vulnerability can occur when memory is allocated to an object that is used after it is deleted (or deallocated). Good programming practice dictates that any reference pointing to an object should be modified when the memory is deallocated, to keep the pointer from continuing to make the area of memory where the object once resided available for use. (A pointer in this abandoned condition is broadly called a "dangling pointer.") If the pointer isn't modified and tries to access that area of memory, the system can become unstable or corrupt. Attackers can use a dereferenced pointer in a variety of ways, including execution of malicious code.

Vulnerability

Defects or bugs that allow for external influence on the availability, reliability,

confidentiality, or integrity of software or hardware. Vulnerabilities can be exploited to subvert the original function of the targeted technology.

Worm

A self-contained malicious program that is able to spread of its own accord. The classification "worm" is only used to describe the ability to spread without a host file (as may be the case with computer viruses) and worms contain many different and varied payloads beyond spreading from host system to host system.

WSDL

Web Services Description Language; an XML-based interface definition language used to describe the functionality offered by a Web service.

Zbot: See Zeus.

Zero day

A previously unknown vulnerability for which no patch from the vendor currently exists. It is referred to as a zero day because the vendor has had zero days to fix the issue.

Zeus

A family of malware that targets the Windows operating system. It is used primarily to steal banking information, but has also been used to install the CryptoLocker ransomware. Also see Gameover Zeus.

Learn more at
hp.com/go/hpsr



Sign up for updates

★ Rate this document



**Hewlett Packard
Enterprise**

© Copyright 2015 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for HPE products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

4AA5-0858ENN, November 2015, Rev. 1