Table of Contents

Key Takeaways ......................................................................................................................... 1
Application Security and 2017 Reported Vulnerability Trends ................................................. 2
Revised OWASP Top 10: A Place to Start the Journey ................................................................. 10
Security and Privacy with GDPR .............................................................................................. 13
Fortify Software Security Analysis Results: Insight for
  New Trends and the Usual Suspects ......................................................................................... 21
Top Vulnerabilities in Mobile Applications .............................................................................. 26
Conclusion & Next Steps ........................................................................................................... 40
Key Takeaways

The number of reported flaws in 2017 was more than double the number found in 2016, according to data from the National Vulnerability Database. Disclosed vulnerabilities that led to denial of service, code execution, memory overflows, information exposure, cross-site scripting, and flaws in web applications accounted for much of the growth.

Attackers are going deep as well as broad, penetrating further into the hardware and software stack. As such, developers, application-security specialists, and information-security managers need to look more broadly across their information-technology stack to guard against software weaknesses. This means IT security professionals need to focus across the entire software ecosystem.

Connected devices from Internet of Things (IoT) remains a high risk.

There is an increased reliance on open source software by developers in 2017 that has resulted in attackers’ increased exploitation of vulnerabilities in open source libraries and components.

The Micro Focus analysis of 2017 Fortify on Demand data observed an increase of over 3x the number of unique vulnerability instances in analyzed web applications compared to 2016 data. Furthermore, Micro Focus’ analysis of Fortify on Demand data in this report identifies the most commonly occurring vulnerabilities, and critical vulnerabilities, in web and mobile applications.

Encryption is a great tool when used properly. With the General Data Protection Regulation that came into effect in May of 2018, data privacy concerns come to the forefront and enterprises need to assess how data security and application security are impacted by GDPR. GDPR necessitates a complete rethinking of data handling processes, forcing a privacy-by-design methodology wherever personal data is collected and stored—regardless of the technology used.

A staggering 90% of applications have at least one issue outside of the OWASP Top 10. 49% of tested applications contained a critical or high-severity weakness that is not covered by the OWASP Top 10. While it’s a good idea to focus on the refreshed OWASP Top 10 list of web vulnerabilities, do not stop there. The supporting data for that list can be a rich source for understanding other important flaws, as can industry-specific sources such as PCI, MISRA, and HIPAA.

1 https://github.com/OWASP/Top10/blob/master/2017/datacall/analysis/OWASP%20Top%202017%20GM%20Data%20Analysis.xlsx
Application Security and 2017 Reported Vulnerability Trends

It has long been understood\(^2\) that a successful attack requires the exploitation of reachable vulnerabilities, which result from inherent weaknesses (design, implementation, or configuration flaws) (Figure 1).

These underlying flaws need to be continually investigated and remediated in order to decrease the risk of a potential attack. Several continually changing factors must be considered:

1. Source code and system architecture evolve
2. Identified vulnerabilities in system dependencies are disclosed
3. Fundamentally new weakness types are defined
4. People maintaining source code and systems change

The attack surface of enterprise systems continues to evolve as well (Figure 2), resulting in additional complexity that must be assessed for risk. While a cyber-attack can involve physical security and social engineering, we must be able to assess potential weaknesses related to both traditional and less-traditional architectures. Client/server, web, and peer-to-peer models have existed for a long time. In the last decade, however, we have seen these models expand to include smartphones, cloud computing, big-data analytics, and large networks of interconnected devices which can also include supervisory control and data acquisition systems/devices (SCADA), embedded systems/devices, and IoT.

\(^2\) Richard Bisbey II and Dennis Hollingworth, Protection Analysis: Final Report, ISI/SR-78-13, University of Southern California/Information Sciences Institute, Marina Del Rey, CA 96291 (May 1978).
These newer models bring with them additional communication protocols (e.g., there are over 30 competing wireless communication protocols for IoT alone), hardware components (e.g., smart phones and IoT devices), and software stacks (e.g., Linux on an embedded system connected to SCADA or the CAN bus of a vehicle). Correspondingly, we must continue to implement strong security and privacy countermeasures to address new risks. The classic enterprise perimeter protection model is no longer feasible. Applications such as web apps, mobile apps, embedded software behind the surface of an IoT device, and SaaS are the new perimeter. Weaknesses in software can exist at any layer of a system stack. With the ever-expanding perimeter for our enterprise and personal systems, we have to carefully consider where risks and countermeasures are to be identified and applied.

This past year brought with it many interesting patterns in the cyber-security industry. In the following sections we review major events and trends, discuss the latest developments related to standards and best practices, present data security challenges for the coming year, and provide quantitative analysis of vulnerability trends from 2017 data.

**Researchers Find Vulnerabilities Deeper in the System Stack**

Several significant changes occurred in last year’s vulnerability landscape.

The number of published flaws jumped 127%, to 14,646 (more than double the 6,447 vulnerabilities published in 2016) according to data from the National Vulnerability Database (NVD). Cyber-attacks cost businesses billions in damages from breaches and costs due to remediation. From data breaches due to missed patches, to finding critical and widespread vulnerabilities in the processors that underpin much of the world’s information technology, coding and design errors in software had a dramatic impact on both businesses and individual consumers.
Not only were the number of reported vulnerabilities on the rise, but specific categories of vulnerabilities appeared to be responsible for a large portion of the change. As depicted in Figure 3—based on text patterns in the CVE descriptions provided by analysis at CVE Details—logical groupings of vulnerabilities that led to denial of service, code execution, memory overflows, information exposure, and cross-site scripting accounted for the vast majority of increases in the last year. (The data from CVE Details allows for a single CVE to be classified as different classes of attacks.)

From our own analysis of the NVD data, Buffer Overflow (CWE-119) accounted for 2,500 CVEs documented in the 2017 dataset, 115% more than the previous year (Figure 4). The top 10 occurring Common Weakness Enumerations (CWEs) in 2017 included Buffer Errors (CWE-119), Cross-Site Scripting (CWE-79), Information Exposure (CWE-200), Improper Access Control (CWE-284), and the category of Weaknesses in Permissions and Privileges (CWE-264).

These classes of weaknesses are illustrative, but not necessarily definitive, since many disclosed vulnerabilities are not classified while others are grouped in a category, such as Weaknesses in Permissions and Privileges (CWE-264) or Resource Management Errors (CWE-399), which should not be assigned to individual vulnerabilities (more specific underlying CWE Base or Variants types should be used whenever possible—a more generic CWE Class type should be used otherwise).

Data for selected years and top vulnerability categories.

Figure 3. Trends in vulnerability disclosures in 2017.
Web applications accounted for much of the growth in disclosed vulnerabilities, representing 51 percent of all issues in 2017, according to data from security-information firm Risk Based Security. Yet, other software domains are also fertile ground for researchers and attackers. Operating systems, for example, continued to be a major focus of flaw finding: nine of the top-ten products with the highest count of reported vulnerabilities were operating systems in 2017 (see Figure 5, “Vulnerability Counts for Top-10 Operating Systems in 2017”). Android had the most reported software flaws of any single product, with almost double the number of reported flaws compared to the number-two product, the Linux kernel, according to a Micro Focus analysis of NVD data.
**Figure 5.** Operating systems with the highest volume of vulnerabilities reported in 2017 (NVD).

**Vulnerability Counts for Top-10 Operating Systems in 2017**

- Android: 842
- Linux Kernel: 447
- iPhone OS: 386
- Mac OS X: 297
- Windows 10: 268
- Windows Server 2016: 252
- Windows Server 2018: 243
- Windows Server 2012: 235
- Debian Linux: 230
- Windows 7: 229

**Figure 6.** Applications with the highest volume of reported CVEs in 2017 (NVD).

**Vulnerability Counts for Top-10 Applications in 2017**

- ImageMagick: 357
- Acrobat: 208
- Edge: 203
- Safari: 186
- Chrome: 154
- Tcpdump: 133
- XnView: 114
- IrfanView: 113
- MySQL: 85
- Internet Explorer: 79
Deeper into the System Stack

Inspecting the flaws that led to the most significant attacks and exploits in 2017 underscores that vulnerability researchers and attackers are penetrating deeper into the stack. While a Micro Focus analysis of NVD data shows that operating systems have the most reported vulnerabilities, other aspects of the software ecosystem are being targeted by vulnerability researchers and attackers as well.

Open source database platform MySQL, for example, had 85 vulnerabilities reported in 2017, while the open source Java Runtime Environment and Java Development Kit both had 69 vulnerabilities. QEMU, an open source virtualization platform, had 65 vulnerabilities reported, while rival Xen had 62 flaws found in 2017.

The massive damages caused by the ransomware attacks WannaCry\(^8\) and NotPetya\(^9\) used vulnerabilities underlying the EternalBlue exploit targeting Microsoft Windows implementation of the Server Message Block protocol (SMBv1) for Microsoft Windows operating systems to infect systems. The combined attacks caused more than a billion dollars\(^{10}\) in business losses\(^{11}\), impacting production, shipping, and services.

Hardware-related flaws such as the Meltdown and Spectre\(^{12}\) vulnerabilities discovered at the end of the year, related to speculative execution, allow passwords and other sensitive data to be stolen from software running on system architectures based upon processors affected by the disclosure.

These incidents show that vulnerabilities are no longer limited to the operating system, the applications, or networking code. IT security professionals need to focus across the entire software ecosystem.

Some of the key areas that became a greater focus of vulnerability for researchers and attackers in 2017 are the following:

1. Entire System Stack under Microscope

The basic processors and hardware underlying many technology products have had design flaws in the past, but security vulnerabilities have been rare (e.g. Rowhammer in 2015\(^{13}\)). Last year’s announcement of two flaws—Meltdown and Spectre—in major processor families showed that even the most widespread systems can benefit from additional security scrutiny.

The problems in Intel, AMD, and ARM processors were not the only flaws discovered in hardware systems in 2017. Researchers found a critical flaw in a specific Broadcom Wi-Fi chipset that allowed remote code execution on the target. The flaw allowed “for code execution on the main application processor in both Android and iOS,” according to Exodus Intelligence, the organization that found the flaw.\(^{14}\)

While finding remotely exploitable attacks against mainstream chips and architectures is rare—despite the discovery of several major issues this year—the Internet of Things is significantly more vulnerable\(^{15}\). A significant number of flaws are found every year in IoT\(^{16}\). The Satori botnet, for example, used a zero-day

12. [https://meltdownattack.com/](https://meltdownattack.com/)
15. [www.iotsecurityfoundation.org/blog/](http://www.iotsecurityfoundation.org/blog/)
16. [www.dhs.gov/securingtheIoT](http://www.dhs.gov/securingtheIoT)
flaw in Huawei home routers\(^\text{17}\) to compromise the devices and control them. Another botnet, Hajime, used a much simpler vulnerability—weak and hardcoded passwords—to compromise home routers and create its own botnet.

Finally, researchers have continued to successfully find vulnerabilities in operational and consumer technology. Academic and industry investigators, for example, discovered a way to attack automobile Controller Area Networks (CANs)—given physical access to a car—to silently disable features of the vehicle.\(^\text{18}\)

### 2. Software Dependencies under Attack

Developers increasing reliance on open source software components have resulted in attackers increasing their focus on vulnerabilities in those libraries and components. Because many software maintainers—43 percent, according to software-security firm Snyk\(^\text{19}\)—do not audit their programs for vulnerable dependencies, flawed components frequently lead to vulnerable software.

A flaw (CVE-2017-12615) in Apache Tomcat, for example, could allow an attacker to upload a Java Server Page to the server and then execute it, if the Tomcat server allows read-write privileges. More seriously, a critical vulnerability in Apache Struts 2 (CVE-2017-5638) could allow attackers to execute arbitrary commands remotely, resulting in various forms of attacks.

Vulnerable components are widespread. Approximately 6% of all Java components in the Central Repository and JavaScript components in the Node Package Master have a known vulnerability, according to component management firm Sonatype.\(^\text{20}\) The open source software library, binutils, had 106 vulnerabilities in 2017, while the Java Development Kit had 69, and the graphics library LibTiff had 60.

### 3. Software Creators Become Targets

Developers and development tools have become a focus of attackers as well. The NotPetya ransomware attack initially spread from a Ukrainian tax-software company whose update server had been compromised. The attack underscored that attacks can infiltrate the software supply chain and infect otherwise legitimate software.

A reminder came in mid-summer when Piriform, a system utility company that had just been bought by security firm Avast, allowed more than 2.3 million customers to be infected by malware installed in an update when its development pipeline had been compromised.

Vulnerabilities affected a range of tools used by developers as well. APKTool\(^\text{21}\), a popular program for reverse-engineering Android programs, and popular integrated development environments had vulnerabilities that could allow an attacker to access developers’ files. In another case, vulnerabilities in Microsoft’s Application Verifier (CVE-2017-5565 to CVE-2017-5567) could be used to replace the verifier with a custom service provided by the attacker.

---

17 [www.sans.org/newsletters/newsbites/xix/101#2](https://www.sans.org/newsletters/newsbites/xix/101#2)
19 [https://snyk.io/stateofossecurity/](https://snyk.io/stateofossecurity/)
21 [https://ibotpeaches.github.io/Apktool/](https://ibotpeaches.github.io/Apktool/)
These types of attacks give attackers opportunities to compromise legitimate software sources and bypass security measures such as source based whitelisting.

4. Connected Devices Are Vulnerable

Connected devices have increasingly become a target as well. The Android operating system, Microsoft’s Windows RT 8.1, and Apple’s iOS, AppleTV, and WatchOS are all in the top-30 products with reported vulnerabilities. In 2017, more than 670 security issues affected a variety of firmware, according to a Micro Focus analysis of NVD data.

At least another 20 flaws have product names that mention SCADA which is used to monitor industrial devices and processes. Products from Schneider Electric, a company known for industrial automation, accounted for 41 vulnerabilities.

Many manufacturers do not create advisories for flaws in Internet-of-Things and industrial devices. The CAN bus flaws in automobiles reported by a team of academic and industry researchers, for example, appeared only in an alert issued by the Industrial Control Systems (ICS) Cyber Emergency Response Team.

5. Defending Against Deep-Stack Attacks Remains a Challenge

Because operating system and application vulnerabilities are widespread, security teams must become well-versed in blunting the impact of vulnerabilities in these well-known layers of the software stack.

While many companies need to improve their asset-management practice, as well as their software-patching processes, updates help companies deal only with known vulnerabilities. These “known-knowns” are a problem that can be handled by improved vulnerability management.

---

In 2017, more than 670 security issues affected a variety of firmware, according to Micro Focus analysis of NVD data.

---


---

Figure 7. Known-knowns, Known-unknowns, and Unknown-unknowns
However, the known unknowns which are often called zero-day vulnerabilities—unknown vulnerabilities of known weakness types—are more difficult to mitigate. Processes, and in some cases tools, continue to lack maturity for detecting attacks on vulnerabilities in IoT devices or in firmware. Part of the challenge for IoT is the wide variety of communication protocols (e.g., wireless network protocols include the following: Bluetooth, BLE, ZigBee, Z-Wave, 6LoWPAN, Thread, WiFi-ah, 2G, 3G, 4G, LTE Cat [0, 1, and 3], LTE-M1, NM-IoT, 5G, RFID, SigFox, LoRaWAN, Ingenu, Weightless [-N, -P, -W], ANT, ANT+, DigiMesh, MiWi, EnOcean, Dash7, and WirelessHART).

Hardening systems against the exploitation of unknown vulnerabilities means focusing on better overall security, rather than focusing on specific disclosed vulnerabilities. Detection of anomalous activity becomes much more important. Testing both software source code and software operation—two tenets of DevOps—can help companies detect vulnerabilities in their own code. Do not forget to also test the source code dependencies within the software stack for previously unknown weaknesses.

Revised OWASP Top 10: A Place to Start the Journey

As mentioned earlier, web applications were a significant attack vector last year, according to data from security-information firm Risk Based Security. But now security professionals have a refreshed list of top 10 critical web application security risks—known as the OWASP Top 10—to help them start making their online apps more secure. That does not mean, however, that web application security should begin and end with the OWASP Top 10 list.

Since the first OWASP Top 10 in 2003, the published list has increased in stature as a security measure. The officially released 2017 report, which included input from more than 70 individual contributors and data from more than 40 organizations, raises awareness amongst developers and managers by encouraging best practices.

As important as the OWASP Top 10 has become, it must be only a starting point for security practitioners. There are hundreds of issues that can affect software security—for instance, there are more than 700 common software security weaknesses identified by the Common Weakness Enumeration (CWE) community and nearly 900 defined in the Fortify Taxonomy of Software Security Errors—so organizations need to look beyond the Top 10 to forge an effective security program.

The 2017 OWASP Top 10
The 2017 edition of the OWASP Top 10 contained several web application risks from the 2013 list:

- Injection flaws, when untrusted data is sent to an interpreter as part of a command or query.
- Broken authentication, where faulty implementation of authentication or session management allows an attacker to compromise passwords, keys, or session tokens and assume a user’s identity.
- Sensitive data exposure, which usually occurs when data is not encrypted.
- Security misconfiguration, commonly a result of insecure default configurations, incomplete or ad
  hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages
  containing sensitive information.
- Cross-site scripting, which results when an application includes untrusted data in a new web page
  without proper validation or escaping, or updates an existing web page with user-supplied data using
  a browser API that can be used to create scripts.
- Using components with known vulnerabilities, such as libraries, frameworks, and other
  software modules.

Two items from the 2013 list were merged. Insecure direct object references and missing function-level
access control became broken access control. This results when restrictions on what a user can do on a
network are not properly enforced.

Three new items were added to the 2017 list:
- XML external entities, which, when evaluated within an XML document, can be used to disclose
  internal files.
- Insecure deserialization, which can be used for remote code execution and replay, and for injection
  and privilege escalation attacks.
- Insufficient monitoring and logging, which contribute to the time required to discover a data breach.

**Rejected, but Not Forgotten**

Two of the controversial additions to the OWASP Top 10 in RC1—“Unprotected APIs” and “Insufficient
Attack Protection”—were removed from the final version of the 2017 OWASP Top 10. In addition, two items
from the 2013 list were scrapped—Cross-Site Request Forgery (CWE-352) and Unvalidated Redirects and
Forwards (CWE-601).

Does exclusion from the Top 10 mean those items should be ignored? Certainly not. In fact, the 2017
OWASP Top 10 team cautions security practitioners: “The Top 10 covers a lot of ground, but there are
many other risks you should consider and evaluate in your organization. Some of these have appeared
in previous versions of the Top 10, and others have not, including new attack techniques that are being
identified all the time.”

The final OWASP 2017 Top 10 report combined analysis of survey data with data from application se-
curity vendors to identify many categories that did not make it into the Top 10. These include Cross-Site
Request Forgery (CWE-352) and Unvalidated Forward and Redirects (CWE-601), as well as Uncontrolled
Resource Consumption (CWE-400), Unrestricted Upload of File with Dangerous Type (CWE-434), User
Interface Misrepresentation of Critical Information (CWE-451), Improper Control of Interaction Frequency
(CWE-799), Inclusion of Functionality from Untrusted Control Sphere (CWE-829), and Server-Side Request
Forgery (CWE-918).
Ultimately, however, we should be aiming to remove the underlying weaknesses that make the vulnerabilities possible, rather than looking for ways to mitigate. If a mitigation fails, or can be bypassed, the vulnerability could still be exploited.

Beyond the Baseline

The supporting data for the list can be a rich source for understanding additional vulnerabilities. In creating the list, OWASP established incident rates for more than 30 vulnerabilities based on data from 23 companies covering 114,000 applications. Some of the most-cited vulnerabilities didn’t make it into the top 10. They include Information Leakage/Disclosure (CWE-200), Unvalidated Forwards and Redirects (CWE-601) and Cross-Site Request Forgery (CWE-352).

It is also important to note that the OWASP Top 10 is a list of web application security risks. While many attacks on organizations do indeed involve the web, there are other attack surfaces that need to be considered when formulating a security policy. For example, social engineering, which does not require breaking an application, is frequently used to gain unauthorized access to systems.

Email, too, is a popular attack vector. In 2017, for example, a report from security provider AppRiver noted that its email protection solution quarantined 14.5 billion emails containing malware.

When determining if your organization’s policy covers all the bases that need to be considered, it is also a good idea to check sources of standards and best practices.

In healthcare, for instance, an organization needs to comply with HIPAA guidelines. In the credit card industry, adherence to PCI DSS is necessary. For the automotive industry, MISRA guidelines for C and C++ are added to the list of relevant background literature. If business is being done with citizens from the European Union, your policy needs to meet the General Data Protection Regulation (GDPR), which took effect in May 2018.

There are also numerous government and professional resources that can be helpful for formulating a security policy, such as the Defense Information Systems Agency’s Security Technical Implementation Guides (STIGs), FIPS 200, NIST Special Publication 800-53 R4, and the ISO 27000 series.

By using the OWASP Top 10 as a starting point and supplementing it with additional resources tailored to a business’s operations, it is possible to craft a solid security policy that reduces risk and optimizes protection.

---

32 https://github.com/OWASP/Top10/blob/master/2017/datacall/analysis/OWASP%20Top%2010%202017%20GM%20Data%20Analysis.xlsx
34 https://misra.org.uk/
35 www.eugdpr.org
36 https://iase.disa.mil/stigs/Pages/index.aspx
38 www.himss.org/nist-special-publication-800-53-r4-security-and-privacy-controls-federal-information-systems-and
39 http://www.27000.org
Security and Privacy with GDPR

No security policy is complete, especially these days, without strong protection for your customers’ data. Europe has long been at the forefront of consumer protection laws, and the EU’s General Data Protection Regulation (GDPR) is a best-practice privacy framework for how to approach encryption and other recommended measures.

First, it is not possible for businesses to know for sure whether any individual customer is an EU citizen. Second, regardless of citizenship status, it is always better to do more rather than less to protect individuals’ privacy. Protecting the privacy of individuals can be a true competitive edge in this market. To achieve the protection, there is good, practical advice to be drawn from GDPR guidance for organizations of all sizes.

The GDPR is a refresh of Europe’s data protection laws, harmonizing statutes across the 28 EU member states. Effective on May 25 this year, GDPR replaces legislation that dates to 1995, before the dot-com boom and prior to the birth of Facebook, Google, Twitter, and cloud computing.

GDPR accounts for recent technological innovations and plans for future developments that will create and use personal data. An additional benefit of GDPR is that it applies a consistent framework to both EU and non-EU based controllers/processors that sell goods and services, or those that monitor EU citizens’ behavior. This replaces the mishmash of differing laws and regulations that exist in the EU today.

Just as PCI DSS is a generally accepted framework of security controls for the protection of credit card data, the GDPR is a structure for handling personal data privacy issues generally, but in the form of a law for any organization doing business with EU citizens.

A major driver behind the GDPR is the principle that data related specifically to a person belongs to that person—not to the organization creating, possessing, or processing it. As a custodian of EU citizens’ personal data, the organization is responsible for protecting the data under the framework laid out in the GDPR.

Personal data is essentially any information that relates to an identified, or identifiable, living individual and may include:

- Name
- Home address
- Email address such as name.surname@company.com
- Identification card number
- Location data (for example the location data function on a mobile phone)
- Internet Protocol (IP) address
- Cookie identification(s)
Advertising identifier of your phone

Data held by a person or company that could be used to uniquely identify a person

GDPR is a law written using the concept of articles. Article 25, Data protection by Design and by Default, requires that business processes for products and services have data protection designed-in during development. A report by the European Union Agency for Network and Information Security elaborates on what needs to be done to achieve privacy and data protection by default. Privacy settings are required to be set at a high level by default, and technical and procedural measures should be used to ensure that the entire processing lifecycle complies with the regulation.

Applications Impacted by GDPR

Even core business processes, such as order taking, can and should be undertaken without the identity of the customer being revealed to unauthorized entities. But beyond traditional back-office processes and applications supporting them, consider the impact to the data lakes being created for analytics by various applications:

- Telcos use their global networks to collect call records from subscribers and consumers, as well as track the location of individuals’ mobile phones. The telcos’ applications process data for fault detection, roaming data, and network optimization.
- Car manufacturers collect sensor data from millions of cars globally to find defects to get ahead of recalls and service issues. With today’s connected cars, however, manufacturers can also have applications that could merge different data sets and track what you are doing and where you are going.
- Retailers look at customer data to determine buying patterns and brand loyalty, detect potential credit card fraud, and create new customer services.
- Health insurers analyze sensitive customer data to detect potential prescription medication fraud, insurance overpayments, and to customize access to this information for customers.

All these industries, and others, cast a wide net to collect data for analytics and could include EU citizen personal data that would require protection under the GDPR. However, sensitive data managed by organizations is often not protected as well as it should be. This has resulted in many large-scale data breaches that have undermined confidence. The Information Beautiful site helps show the impact and increasing escalation of these attacks.

This consistent inability to demonstrate adequate protection of personal data was a driver behind the GDPR’s creation. GDPR Article 32 requires businesses to protect their systems and applications “from accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to personal data” by taking into account “appropriate technical and organizational measures.”

If an organization retains personal data about EU citizens, the GDPR requires reducing the risk of holding that data. Reducing disclosure risk is where the concepts of pseudonymization and encryption come into play. Think of this as a way of de-identifying, or neutralizing, the data so businesses can still get meaning from it, but without the costs associated with any breaches.
Encryption and Pseudonymization: What the GDPR Says

In practical terms, the GDPR necessitates a complete rethinking of data handling processes, forcing a privacy-by-design methodology whenever personal data is collected and stored—regardless of the technology used. Although the GDPR does not specify the technologies required to enable compliance, it strongly hints at the use of encryption and pseudonymization as approaches to protect personal data.

Encryption is referenced four times in the text of GDPR (Recital 83, Articles 6, 32, and 34). In each case it is advocated as an approach to mitigate the risks of processing personal data. Encryption is positioned by the GDPR as a mechanism that renders personal data unintelligible to unauthorized individuals, which is a mitigating action against both a data breach and the requirement to make a public notification of that breach (Article 34).

Pseudonymization is cited more often than encryption within the GDPR (Recitals 26, 28, 29, 75, 85, and 156, and Articles 4, 6, 25, 32, 40, and 89). Pseudonymization protects the identity of individuals through the process of removing personal identifiers from data. Although the GDPR encourages the use of pseudonymization to "reduce risks to the data subjects," (Recital 28), pseudonymized data is still considered personal data (Recital 26) and therefore remains covered by the GDPR.

Applying Encryption

While neither encryption nor pseudonymization are mandated by the GDPR, the EU is giving organizations a strong hint that it’s a best practice. However, few industry segments, let alone specific companies, have a long history of operating encryption systems and managing encryption keys. Beyond financial services, telecommunications, and the defense sector, most industries have limited experience with using encryption in their business processes. Encryption and pseudonymization are very generic categories, and there are many variants with differing levels of coverage, capability, flexibility, and manageability.

To understand the limitations of the approach typically used, as well as the best way to encrypt sensitive data, it is helpful to understand a notional “encryption stack” that is conceptually very similar to the familiar internet protocol stack that provides the basis for today’s internet.

The Encryption Stack

The internet protocol suite that is known as “TCP/IP” (for two of its components: the Transport Control Protocol and the Internet Protocol) defines the communications protocols used in most of today’s computer networks. One notable feature of TCP/IP is how it abstracts the functionality of a computer network into four layers that we think of as comprising a “stack.” In this notional stack, we have multiple logical layers that process information.

The TCP/IP stack comprises four layers: Application, Transport, IP and Network Access (sometimes called the Link layer), as shown in Figure 8. As the arrows in Figure 8 suggest, information is passed only between adjacent layers of the TCP/IP stack. So a task conducted at the Transport layer can pass information to a task running one layer away at the IP layer, but not to a task conducted two layers away at the Network Access layer.
System architectures can be represented by many heterogeneous models. As such, the Open Systems Interconnection (OSI), TCP/IP, Linux kernel, and Intel CPU privilege ring models are all depicted side by side in Figure 8. These four abstract representations provide different ways to think about where encryption could occur in a system’s architecture with varying levels of complexity. In each of the four represented models, data transfers from one layer into another through well-defined APIs. Through color coding, we also observe that there are privilege boundaries between these layers (purple: physical hardware access; yellow: processor-enforced high-privilege instructions (e.g. kernel-mode); blue: processor-enforced low-privilege instructions (e.g. user-mode)). By conducting encryption activities at different layers of the system stack different risk factors are eliminated or introduced.

It is useful to think of encryption as an input/output process where data must be encrypted or decrypted as the information is retrieved or sent between layers. Restricted instructions that can manipulate memory directly are callable within protection ring 0 of the Intel ring model, as such code executing in ring 3 (i.e. instructions invoked from user-mode memory for processes executing in a Linux OS) cannot directly manipulate the contents of physical memory. Processes running the code have to ask the Kernel to do that for them.

Thus, for example, data retrieved from a network interface adapter could be brought into the Linux kernel model and placed in memory in an encrypted form. The encrypted data would need to be decrypted before use at some layer above the network layer before it is used by the application. The decision about which layer would be appropriate, and what permissions for the instruction set would be needed, depends on what you are trying to achieve.

Figure 8. Conceptual relationship between OSI, TCP/IP, Linux, and Intel Rings depicting architectural layers and access permissions
Similarly, it can be useful to think of encryption as taking place either relative to, or at different levels of, the TCP/IP stack, thus creating a notional “encryption stack” that closely parallels the TCP/IP stack. Transport Layer Security encryption that is used by secure websites, for example, operates between the Application layer and the Transport layer. IPSec encryption—used to create virtual private networks (VPNs)—operates at the IP layer. Link encryptors encrypt at the Network Access layer. Full-disk encryption (FDE) operates below the Network Access layer, as does transparent database encryption (TDE).

There are good reasons to encrypt at different places relative to the TCP/IP stack, but it is important to understand that when you encrypt at a particular place in the stack, the encryption protects only against threats that target layers at or below where the encryption takes place (data is in clear text above the layer where encryption takes place.)

If you protect data with FDE, for example, the encryption will protect the data while it is stored on encrypted disks. When the data leaves the disks and is handed off to the Network Access layer, FDE no longer protects it. So if a cybercriminal manages to steal a hard disk that is encrypted with FDE, they will probably be unable to access its contents. But if a cybercriminal intercepts information being transmitted across a network, the FDE provides absolutely no protection for the data.

Similarly, malware that reads data from a hard drive protected with FDE will be totally unaffected by the FDE—once the encrypted data is read from the hard disk, the FDE no longer protects it.

Similarly, TDE protects data only while it is stored in a database. When the data is read from the database, the protection provided by the encryption is lost, so attacks that operate at most levels of the encryption stack are totally unaffected by TDE.

And if you are using TLS to encrypt data between the Transport and Application layers, TLS will protect against attacks that target the Transport layer, the IP layer, and the Network Access layer, but it will not protect against attacks that target processes running at the Application layer.

The biggest and most severe data breaches that have affected both the public and private sectors all operate at the Application layer. This includes almost all versions of both malware and advanced persistent threat (APT) attacks. Because of this, encrypting at the application layer is the only form of encryption that will address these serious threats. TLS encryption does not protect against threats, nor does FDE, nor does TDE. But since these are the most common forms of encryption currently used by businesses, most of their use of encryption is ineffective at protecting against the most serious threats they face.

**Application Layer Vulnerabilities**

Encryption alone will not suffice to ensure data security. It must be supported by strong key management for the encryption to be useful at all. In a nutshell, key management is all aspects of encryption that do not directly relate to encrypting or decrypting data. It includes tasks such as generating keys, storing keys, and transporting keys.
Key management is so important to the business use of encryption that Micro Focus likes to echo the sentiment of General Robert H. Barrow, the former Commandant of the Marine Corps. Barrow once noted that “Amateurs talk about tactics, but professionals study logistics,” while Micro Focus is fond of saying that “Amateurs talk about encryption, but professionals study key management.”

It is not too hard to create secure encryption schemes, but devising secure ways to implement key management is much, much harder. Essentially anything that can go wrong with hardware or software can affect the security of key management. In particular, software vulnerabilities in the application layer can dramatically reduce, or even eliminate, the security that encryption can provide. Cracking an encryption key may take billions of years on powerful computers, but finding a weakness in software is much faster and easier.

The application layer itself consists of many layers and components. Defects in the implementation of encryption technology, the presence of vulnerabilities in the application layers above and below the layer where encryption occurs, vulnerabilities in any of the software components with which the application interfaces, and configuration and maintenance errors in the systems and networks that store and process sensitive data—all these will determine the effectiveness of data protection.

For a comprehensive assessment of data security, it is important that businesses take a holistic view and include application security and vulnerability assessment as part of their software development program.

Broadly, we believe that application vulnerabilities can fall into four groups related to data security: (1) privacy violation; (2) insufficient data protection; (3) access violation; and (4) indirect access to sensitive data.

PRIVACY VIOLATION
First, encryption or pseudonymization of data is required in data security to ensure the privacy of data. Data confidentiality and the privacy of the individual are compromised when applications simply fail to encrypt or pseudonymize data before writing to external devices such as the console, file system, or the network. Examples of this include:

- Transmitting sensitive data in plaintext over an HTTP connection leaves it vulnerable to a man-in-the-middle attack.
- Writing sensitive data in plaintext to log files cannot be assumed to be safe, as log files may fall prey to an insider threat.
- Printing or displaying sensitive data in plaintext to a console is visible not only to the intended audience but also to malicious onlookers.

Research\(^\text{48}\) shows that fewer than 5% of the data breaches in the first half of 2017 involved encrypted data, thus rendering the stolen data useless. In other words, 95% of the stolen data may have potentially been prevented from abuse had it been encrypted.

---

INSUFFICIENT DATA PROTECTION
Second, it is important to ensure that the best data protection mechanisms are not undermined through faulty execution or implementation flaws in the encryption method. While encrypting data at the application layer will ensure that data flows, encrypted, through the lower layers of the system stack, any flaws in the implementation or configuration of the encryption algorithm will defeat the purpose.

Examples of where these flaws could occur include the following:
- Use of a cryptographic algorithm that is known to be weak (e.g. SHA-1)
- Unprotected storage of cryptographic keys or credentials
- Incomplete or insecure configuration of encryption parameters

ACCESS VIOLATION
Third, even if encryption is implemented correctly at the application layer, access violation vulnerabilities in the systems and processes that store and process sensitive data may compromise security. The September 2017 Deloitte breach\(^49\) that compromised several thousand records of its client data resulted from attackers gaining unauthorized access to its email server and all of Deloitte’s administrator accounts.

Implementation and configuration errors in authentication, authorization, and access policies can include:
- Weak password requirements
- Missing authorization checks on critical resources
- Misconfigured access control policies

These will allow an attacker to bypass authorization checks to access sensitive data or gain elevated privileges to system resources and the ability to execute malicious arbitrary commands, leaving sensitive data vulnerable to misuse.

INDIRECT ACCESS TO SENSITIVE DATA
Finally, while errors in implementation of encryption technology or access procedures for restricted system resources, lead to direct violation of data security, there is a whole suite of other application vulnerabilities that if exploited successfully may result in indirect access to sensitive data. These vulnerabilities, which are often overlooked, can pose a serious threat to data security.

The recent Meltdown and Spectre\(^50\) vulnerabilities related to speculative execution demonstrate how architectural flaws in memory management can violate isolation boundaries between applications and operating system. These issues remind us that low-level vulnerabilities such as Memory Leak, Cache Management and Buffer Overflows can all compromise data by allowing unauthorized access to cleartext data that may be found in cache, registers, or on a system heap or stack during data processing. Similarly, Unhandled Exceptions in the application can reveal sensitive system or personal data that may be on the program stack trace.


\(^50\) [https://meltdownattack.com/](https://meltdownattack.com/)
Injection vulnerabilities such as SQL Injection and Command Injection are yet other examples of vulnerabilities that may allow attackers to gain unauthorized access to system resources and data by running malicious code on the server. The risk posed by these injection vulnerabilities to personal data security needs to be evaluated in the context of the application being assessed. While a SQL Injection on a company’s inventory database will not compromise personal data, the same SQL Injection vulnerability on a human resources database will.

**VULNERABILITIES IN CODE YOU ARE DEPENDENT UPON**

The above vulnerabilities need to be addressed not only with your in-house developed code, but also in third party software components that interface with your application. Vulnerabilities, such as S2-046 referenced by CVE-2017-5638, have resulted in large data breaches due to OGNL Expression Injection existing in a commonly used library dependency. This injection vulnerability allowed attackers to infiltrate back-end systems, including databases holding sensitive personal data of clients. The exploited vulnerability existed in the Apache Struts 2 component of the application, an open source framework for creating modern web applications.

It is therefore critical that vulnerabilities in third party and open source software components be addressed with the same urgency as vulnerabilities in your own code. This includes ensuring that:

- Applications are deployed in a secure manner
- Systems and applications are patched and updated in a timely fashion
- Servers and other system resources are configured with security in mind

Misconfiguration of River City Media backup servers led to the exposure of 1.4 billion records containing personal email addresses, physical addresses, and IP addresses. Organizations should test, assess, and remediate vulnerabilities in external components to ensure that these components do not compromise and undermine their data security effort.

By reflecting upon the mistakes of the past, organizations are better able to plan for data security in the future. As organizations garner their resources to address GDPR compliance requirements it would be a mistake to implement data security measures without holistic consideration for application security. It is therefore imperative for organizations to ensure that all systems, services, and applications that handle and process sensitive data are secure by themselves.

GDPR Article 32 requires businesses to protect its systems and applications “from accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to personal data” by taking into account “appropriate technical and organizational measures.” These technical measures must include a formal process for application security and vulnerability assessment if comprehensive data security is to be achieved.

---

51 [https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638)
Fortify Software Security Analysis Results: Insight for New Trends and the Usual Suspects

This section presents our analysis of anonymized and sanitized vulnerability data collected by the Micro Focus Fortify on Demand (FoD) platform. The service analyzes web and mobile applications, and applies various security testing methodologies including static source code analysis, dynamic black box penetration testing, mobile and manual analysis.

Results from these different approaches are normalized under the common vocabulary of the Fortify Software Security Taxonomy, also known as the Seven Pernicious Kingdoms (7PK). This taxonomy aims to provide an explanation of weaknesses from a developer-remediation point of view (i.e. what underlying change is required to fix the weakness). Since the original taxonomy was created in 2005, updates have been made to the original categories so the guiding principles are applicable to all weakness categories uniformly, regardless of the method used to detect the weakness.

The subset of FoD data analyzed in this report was collected between October 31, 2016 and October 30, 2017. Web application data was collected from more than 7,800 applications and the mobile dataset was collected from more than 700 applications.

53  https://vulncat.fortify.com
Web Application Results by Kingdom

The likelihood of web applications found to be vulnerable per kingdom

<table>
<thead>
<tr>
<th>Kingdom</th>
<th>% AppCount 2016</th>
<th>% AppCount 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>21%</td>
<td>20%</td>
</tr>
<tr>
<td>Time and State</td>
<td>24%</td>
<td>25%</td>
</tr>
<tr>
<td>Code Quality</td>
<td>30%</td>
<td>35%</td>
</tr>
<tr>
<td>API Abuse</td>
<td>34%</td>
<td>37%</td>
</tr>
<tr>
<td>Input Validation and Representation</td>
<td>50%</td>
<td>57%</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>60%</td>
<td>70%</td>
</tr>
<tr>
<td>Environment</td>
<td>76%</td>
<td>81%</td>
</tr>
<tr>
<td>Security Features</td>
<td>91%</td>
<td>93%</td>
</tr>
</tbody>
</table>

Figure 9. Web application vulnerability distribution across Kingdoms for 2016/2017

Figure 9 shows Security Features again topping the list, with 93% of the tested web applications having an issue of this type in 2017 vs. 91% in 2016. Except for the Errors kingdom, every other kingdom saw the number of vulnerabilities go up. The reduction in Errors, however, is only 1%, which can be attributed to the complete elimination of the “Poor Error Handling: Overly Broad Throws” and “Poor Error Handling: Throws Inside Finally” categories in the scan results.

While the distribution of vulnerable applications looks the same, there is a significant jump in the number of unique issues reported over the last year. In the 2017 report, we presented the analysis of approximately 7,500 tested web applications containing approximately 2.4 million unique instances of vulnerabilities in all categories combined. This year we analyzed over 7,800 applications, and the number of unique vulnerabilities jumped to approximately 7.8 million.

The median number of unique issues reported is 35 in 2017 vs. 25 in 2016, implying that half of our sample size had fewer than 35 unique issues. Additionally, 92% of applications reported fewer than 1,000 issues. In 2016, 96% of the overall applications reported 1,000 or fewer issues. Interestingly, the kingdom distribution graph in Figure 9 looks very similar even when plotted for only applications with 1,000 or fewer issues.

Security Feature remains the number one domain among all reported issues.
Mobile Results by Kingdom

Mobile applications showed a different trend than web applications. Vulnerability counts went down in most kingdoms except in Code Quality issues. Null Dereference increased to 21% (3% in 2016). Unreleased Resources: Streams went up to 24% (4% in 2016). Java based Android applications are responsible for most of these issues at over 18% for Null Dereference issues and 17% for Unreleased Resources: Streams issues. Meanwhile, for the same categories, iOS applications were responsible for 4% and 7% of the vulnerabilities respectively. Both of these categories have critical consequences.

Another observation from Figure 10 is the reduction in the Environment and API Abuse kingdoms. From Environment, Android Misconfiguration: Debug Information saw a reduction from 15% in 2016 to 8% this year. Xcode Misconfiguration: Personal Information and Xcode Misconfiguration: Lacking Memory Protection both saw reduced occurrences as well at 31% and 2% respectively from 43% and 6% respectively in 2016.

Often Misused: Ad/Analytics Frameworks and Often Misused: General Pasteboard in the API Abuse kingdom, both went down from 46% and 35% in 2016 to 26% and 23% in 2017.

Figure 10. The likelihood of mobile applications found to be vulnerable per kingdom

Mobile applications showed a different trend than web applications. Vulnerability counts went down in most kingdoms except in Code Quality issues. Null Dereference increased to 21% (3% in 2016). Unreleased Resources: Streams went up to 24% (4% in 2016). Java based Android applications are responsible for most of these issues at over 18% for Null Dereference issues and 17% for Unreleased Resources: Streams issues. Meanwhile, for the same categories, iOS applications were responsible for 4% and 7% of the vulnerabilities respectively. Both of these categories have critical consequences.

Another observation from Figure 10 is the reduction in the Environment and API Abuse kingdoms. From Environment, Android Misconfiguration: Debug Information saw a reduction from 15% in 2016 to 8% this year. Xcode Misconfiguration: Personal Information and Xcode Misconfiguration: Lacking Memory Protection both saw reduced occurrences as well at 31% and 2% respectively from 43% and 6% respectively in 2016.

Often Misused: Ad/Analytics Frameworks and Often Misused: General Pasteboard in the API Abuse kingdom, both went down from 46% and 35% in 2016 to 26% and 23% in 2017.
Next we will look at the top vulnerabilities that were reported this year.

**Top Vulnerabilities in Web Applications**

The 10 most commonly occurring vulnerabilities in the Web applications dataset

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>% AppCount 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden Field</td>
<td>35%</td>
</tr>
<tr>
<td>Cross-Frame Scripting</td>
<td>36%</td>
</tr>
<tr>
<td>Web Server Misconfiguration: Unprotected Directory</td>
<td>37%</td>
</tr>
<tr>
<td>Cookie Security: HTTP Only not Set</td>
<td>42%</td>
</tr>
<tr>
<td>Privacy Violation: Autocomplete</td>
<td>45%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Protocol</td>
<td>47%</td>
</tr>
<tr>
<td>Insecure Transport: Missing Public Key Pinning</td>
<td>48%</td>
</tr>
<tr>
<td>Insecure Transport: HSTS not Set</td>
<td>51%</td>
</tr>
<tr>
<td>Cookie Security: Cookie not Sent Over SSL</td>
<td>51%</td>
</tr>
<tr>
<td>System Information Leak: External</td>
<td>58%</td>
</tr>
</tbody>
</table>

Figure 11. Top 10 vulnerabilities in web applications

Figure 11 shows the top 10 issues in web applications. Nine out of 10 vulnerability categories are the same as last year. The addition of Insecure Transport: Missing Public Key Pinning replaced Web Server Misconfiguration: Server Error Message, which is down to twelfth place this year. A total of 48% of applications in 2017 did not opt for using HTTP Public Key Pinning header, vs. 26% last year.

In 2017, security researchers have reversed their opinion about using the public key pinning header to make a site inaccessible in case of a lost private key or to mitigate the possibility that an attacker can inject this header with keys he/she controls. Cross-frame scripting went down, from second place to tenth place, with 16% fewer applications vulnerable to this weakness. Interestingly, among the top 10 in Figure 11, only three categories are of critical or high severity.
Next, we take a look at the top 10 issues that were considered critical and high severity.

The 10 most commonly occurring critical-class vulnerabilities in the applications dataset

- **Log Forging**: 11%
- **Insecure Transport: Weak SSL Cipher**: 13%
- **Privacy Violation**: 13%
- **Unreleased Resource: Streams**: 14%
- **Password Management: Hardcoded Password**: 15%
- **Cross-Frame Scripting**: 18%
- **Insecure Transport: Weak SSL Protocol**: 19%
- **Null Dereference**: 20%
- **Cross-Site Scripting: Reflected**: 21%
- **Password Management: Password in Configuration**: 21%

![Figure 12a. 10 Critical Web Application Vulnerabilities 2017](image)

The 10 most commonly occurring critical-class vulnerabilities in the applications dataset

- **Password Mgmt: Hardcoded Password**: 11%
- **Often Misused: Login**: 13%
- **Privacy Violation**: 13%
- **Password Mgmt: Password in Config File**: 13%
- **Unreleased Resource: Streams**: 13%
- **Insecure Transport**: 14%
- **Null Dereference**: 17%
- **Insecure Transport: Weak SSL Protocol**: 21%
- **Cross-Site Scripting: Reflected**: 24%
- **Cross-Frame Scripting**: 29%

![Figure 12b. Top 10 Critical Web Application Vulnerabilities 2016](image)
As shown in Figure 12a, Log Forging and Insecure Transport: Weak SSL Cipher ascended into the top 10 replacing Insecure Transport and Often Misused: Login. Password Management: Password in Configuration File arose to become the top critical issue at 21% in 2017 vs. only 13% in 2016 (Figure 12b shows the top 10 from the 2016 data for reference). Applications developed using JAVA/J2EE and .NET technology showed most of these instances accounting for 90% of those applications. PHP and JavaScript/XML/HTML based applications also showed this password management vulnerability, while not significant enough for our overall sample size, but noticeable when only compared with applications based on PHP and JS/XML/HTML respectively. Passwords stored in clear text within configuration files can compromise the whole application if the contents of the file ever get exposed.

With the increased focus on SSL/TLS security since the Heartbleed vulnerability, it is not a surprise that Insecure Transport: Weak SSL Protocol and Insecure Transport: Weak SSL Cipher issues dominated the top 10 this year. Recent increased focus on SSL flaws disqualified some of the popular cipher suite configurations (e.g. Sweet32 attack and FREAK) that may have resulted in many Weak Cipher SSL configurations. This category affected only 10% of applications in 2016.

Overall, 79% of tested applications had at least one critical or high severity issue, vs. 80% last year.

Next we analyze the top 10 mobile vulnerabilities.

Top Vulnerabilities in Mobile Applications

The 10 most commonly occurring vulnerabilities in the mobile applications dataset

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>% AppCount 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Injection</td>
<td>34%</td>
</tr>
<tr>
<td>Privacy Violation: HTTP GET</td>
<td>34%</td>
</tr>
<tr>
<td>Privacy Violation: Screen Caching</td>
<td>35%</td>
</tr>
<tr>
<td>Privacy Violation: Geolocation</td>
<td>37%</td>
</tr>
<tr>
<td>Privacy Violation: iOS Property List</td>
<td>38%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Protocol</td>
<td>50%</td>
</tr>
<tr>
<td>Insecure Storage: Insufficient Data Protection</td>
<td>51%</td>
</tr>
<tr>
<td>Weak Cryptographic Hash</td>
<td>52%</td>
</tr>
<tr>
<td>Insecure Storage: Lacking Data Protection</td>
<td>56%</td>
</tr>
<tr>
<td>System Information Leak: Internal</td>
<td>58%</td>
</tr>
</tbody>
</table>

Figure 13. Insecure Transport: Weak SSL Protocol which went up by 10% to 50% in 2017 and Privacy Violation: HTTP GET which went up to 34% from 17% in 2016
Apart from changing ranks within the top 10 as shown in Figure 13, most common categories remain the same as in 2016. The two new categories this year in the top 10 are Insecure Transport: Weak SSL Protocol which went up by 10% to 50% in 2017 and Privacy Violation: HTTP GET which went up to 34% from 17% in 2016. They are both related as they both can cause sensitive data to leak while in transit from client to server. Developers and application owners need to understand that rules for securing data sent over HTTP protocol are the same regardless of whether it is mobile or web application.

The **10 most commonly occurring critical-class vulnerabilities in the mobile applications dataset**

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>% AppCount 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password Management</td>
<td>15%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Cipher</td>
<td>15%</td>
</tr>
<tr>
<td>Null Dereference</td>
<td>17%</td>
</tr>
<tr>
<td>Privacy Violation: HTTP GET</td>
<td>19%</td>
</tr>
<tr>
<td>Unreleased Resource: Streams</td>
<td>19%</td>
</tr>
<tr>
<td>Input Interception: Keyboard</td>
<td>20%</td>
</tr>
<tr>
<td>Insecure Storage: Lacking Data</td>
<td>22%</td>
</tr>
<tr>
<td>Privacy Violation</td>
<td>23%</td>
</tr>
<tr>
<td>Insecure Transport: Disabled App</td>
<td>30%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Protocol</td>
<td>34%</td>
</tr>
</tbody>
</table>

*Figure 14. Top 10 Critical and High-Severity Vulnerabilities in mobile applications 2017*
The 10 most commonly occurring critical-severity vulnerabilities in the mobile dataset

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>% AppCount 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insecure Storage: Lacking Data Protection</td>
<td>7%</td>
</tr>
<tr>
<td>Privacy Violation: Insufficient Authentication Mitigation</td>
<td>8%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Cipher</td>
<td>9%</td>
</tr>
<tr>
<td>Parameter Tampering: Special Characters</td>
<td>9%</td>
</tr>
<tr>
<td>Insecure Transport</td>
<td>12%</td>
</tr>
<tr>
<td>Intent Manipulation: Unvalidated Input</td>
<td>13%</td>
</tr>
<tr>
<td>Cross-Site Scripting: Reflected</td>
<td>15%</td>
</tr>
<tr>
<td>Insecure Transport: Weak SSL Protocol</td>
<td>19%</td>
</tr>
<tr>
<td>Password Management: Weak Password Policy</td>
<td>20%</td>
</tr>
<tr>
<td>Account Management: Inadequate Account Lockout</td>
<td>22%</td>
</tr>
</tbody>
</table>

Figure 14 shows the most common critical and high-severity issues in mobile applications. Unlike the top 10 common issues overall, which remained mostly unchanged compared to last year, the critical issues list had 7 new categories surface to the top 10 this year (Figure 15).

Insecure Transport: Disabled App Transport Security positioned at #2, at 30%, is a new category for Swift applications; it detects when applications are configured to partially or entirely opt out of App Transport Security (ATS). ATS enforces best practices for secure network connections such as TLS1.2 and forward secrecy. Insecure Transport: Weak SSL Protocol indicates a similar issue where the application is configured with a weak protocol (e.g. SSLv3 and TLS1.0). A significant number of applications showing ATS as being disabled could be due to the fact that Apple enables it by default and developers might be finding it hard to configure applications correctly which might lead to disabling the feature altogether. This may also signify the need to provide better training and tools for developers to be able to choose strong configurations to mitigate these issues.

Overall, 89% of tested mobile applications had at least one critical or high-severity issue, which is up by 23% from last year’s percentage at 66%.

Consistent with the security issues that have made recent headlines, our data shows that applications—both web and mobile—are still being developed initially without the proper use of cryptography to secure data at rest and data in transit.
The following section will investigate how the web application trends we see in 2017’s sample applications compared to the 2017 OWASP Top 10.

**Standards and Best Practices**

Developers and auditors often seek guidance from security researchers for a way to prioritize a large number of weaknesses since finding and fixing each of them takes significant effort.

Often based on data analytics, best practices are suggestions for which vulnerabilities should be found and fixed first (e.g. OWASP Top 10). Standards provide requirements that suggest the minimum acceptance criteria for ensuring application and data security (e.g. PCI DSS - Payment Card Industry Data Security Standard). Applications that accept electronic payment using credit and debit cards must meet requirements laid out in the PCI DSS compliance standards. Security related standards are developed and enforced by standards bodies consisting of security experts from that business domain.

Figure 16 views web vulnerabilities from the analyzed data, grouped by OWASP Top 10 2017 mappings. Evidently, when input injection issues such as Cross-Site Scripting (A7) and XML External Entity (A4) injection are separated, the other injection weakness types combined under A1 occur in fewer applications than the most prevalent vulnerability grouping (A3). Sensitive Data Disclosure encompasses critical categories like Insecure Transport, Password Management, and Privacy Violation issues which, as shown in Figure 16, also occur in the majority of the web application in this analysis.

![Web Application vulnerabilities by OWASP 2017 Mappings](image)

**Figure 16.** Web application vulnerabilities grouped by OWASP 2017 mappings
The statistic that draws our attention in Figure 16 is “None.” A staggering 90% of applications have at least one issue outside of the OWASP top 10. Vulnerabilities that do not fit in any of the OWASP Top 10 categories are marked None in the Micro Focus analysis of FoD data. It contains categories such as System Information Leak (#1 in the web top 10 in Figure 11) and Cross-Frame Scripting (#9 in the web top 10 in Figure 11). System Information Leak is different than Sensitive Data Exposure represented by A3 in the OWASP Top 10. System Information Leak reveals information about the application stack that is used to develop and host applications vs Sensitive Data Exposure which is about an application revealing personal data about the user (such as username, password, or IP address). Sensitive data should be protected using strong cryptography whenever possible.

In an earlier section, we point out that there are critical vulnerabilities outside of the OWASP Top 10, and the analysis in Figure 16 and Figure 17 provide convincing evidence. Categories in None are not lower in severity, either. Figure 17 shows only critical and high severity issues, and None still takes a position at #2, with 49% of tested applications containing a critical or high-severity weakness that is not covered by the OWASP Top 10.

Critical Web Application vulnerabilities by OWASP 2017 Mappings

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>% AppCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4 XML External Entities (XXE)</td>
<td>9%</td>
</tr>
<tr>
<td>A8 Insecure Deserialization</td>
<td>11%</td>
</tr>
<tr>
<td>A9 Using Components with Known Vulnerabilities</td>
<td>13%</td>
</tr>
<tr>
<td>A2 Broken Authentication</td>
<td>14%</td>
</tr>
<tr>
<td>A5 Broken Access Control</td>
<td>21%</td>
</tr>
<tr>
<td>A7 Cross-Site Scripting (XSS)</td>
<td>23%</td>
</tr>
<tr>
<td>A1 Injection</td>
<td>28%</td>
</tr>
<tr>
<td>A6 Security Misconfiguration</td>
<td>39%</td>
</tr>
<tr>
<td>None</td>
<td>49%</td>
</tr>
<tr>
<td>A3 Sensitive Data Exposure</td>
<td>64%</td>
</tr>
</tbody>
</table>

Figure 17. Critical and high-severity web application vulnerabilities grouped by OWASP 2017 mappings

While none of the standards covers all vulnerability categories, standards and best practices other than the OWASP Top 10 2017 may provide better coverage of critical and high severity issues than OWASP Top 10 (which is one of most popular best practices). For example, in the case of NIST 800-53, None was ranked at the third spot, with 58% of tested applications having at least one issue that was not mapped.
However, only 20% of tested applications had issues with critical or high severity that were not covered by this mapping.

When viewed from the perspective of the PCI DSS, the data showed similar trends. While 74% of tested applications contained at least one issue outside of the PCI DSS standard and mapped to None, only 21% of the identified vulnerabilities mapped to None were considered critical severity. PCI DSS, however, has a catch-all mechanism to ensure critical issues are remediated, which demands that all high-risk vulnerabilities are addressed (Requirement 6.5.6).

**Web Applications Vulnerable to GDPR Categories**

![Figure 18. Application vulnerabilities grouped by GDPR categories.](image)

Figure 18 shows applications that contain vulnerabilities which could undermine GDPR because they have issues in one of the four GDPR logical groupings of vulnerabilities outlined in the section “Applications Impacted by GDPR”. Using the same population of tested applications that we have studied above, we logically grouped the vulnerabilities by their relevance to Privacy Violation, Insufficient Data Protection, Access Violation, and Indirect Access to Sensitive Data. We assume that all applications are potentially within the scope of GDPR legislation for this comparison.
Some 63% of applications have one or more issues that are not mapped and are not part of the compliance. This number goes down to 25% when we consider only critical and high-severity issues (see Figure 19).

The graphs in Figure 18 and Figure 19 imply that most applications attempt to mitigate critical privacy violations. However, they fail to properly implement encryption and pseudonymization of the data. Of the applications under consideration, 59% contained critical vulnerabilities such as Insecure Transport: Weak SSL Protocol, Insecure Storage: Insufficient Data Protection, and Password Management: Weak Password

**Top 10 Critical Web Vulnerabilities in GDPR: Insufficient Data Protection**

![Figure 19. Critical only application vulnerabilities grouped by GDPR categories](image1)

![Figure 20. Top 10 critical web vulnerabilities cause for GDPR: Insufficient Data Protection](image2)
Policy. Above, in Figure 20, we show the top 10 critical vulnerabilities, related to the GDPR logical grouping of vulnerability categories for Insufficient Data Protection, by the percentage of applications that contained at least one reported vulnerability of the specified type.

It is evident that applications apply secure SSL/TLS connections but fail to select a strong configuration settings in addition to not handling keys and passwords in a secure manner. The top 10 occurring categories, shown in Figure 19, highlight the need to eliminate underlying weaknesses in the design, implementation, and configuration of software to prevent undermining the benefits of using encryption to protect sensitive data.

Vulnerabilities in Open Source Dependencies
Many of today’s applications utilize open source libraries as a means for outsourcing implementation of common software utilities. This practice enables fast-paced development that fits into a modern development strategy such as DevOps. However, keeping track of risks inherited from these external dependencies is just as important for the application’s overall security strategy as it is to track the vulnerabilities arising from code developed in-house.

Below we present an analysis of the security posture from a dataset of 651 applications as it relates to software dependencies. The section highlights the most common weaknesses, vulnerable libraries, and severity of the weaknesses reported in web applications scanned by Fortify on Demand with Sonatype. Our sample of 651 applications referenced 293 external libraries (1,234 when we consider each version as a unique library reference). There were 434 CVEs reported across the 293 external libraries which affected the 651 applications.

% OpenSource Component

![Figure 21. Open source percentages in referenced components](image-url)
RELIANCE ON OPEN SOURCE COMPONENTS

CVE by Severity

![Chart showing CVE by severity](image)

**Figure 22.** Percentage of CVE occurrences grouped by severity

Last year, we observed that 83% of applications studied had referenced at least one open source component. This year, 100% of the 651 applications in our sample set had at least one open source component; more than half of the applications used many more than one. As Figure 21 shows, 53% of applications, which consumed components, chose to select 75% or more of their components from the open source domain.

% Applications containing CVE by severity

![Chart showing % applications containing CVE by severity](image)

**Figure 23.** Applications containing one or more vulnerability grouped by severity

There were 434 CVEs reported in these components. As shown in Figure 22, more than 90% of CVEs reported in referenced libraries are either severe or critical (CVSS base score >4.0).
Figure 23 shows applications that contain at least one CVE by severity type, which shows that 64% of the applications contain at least one critical known open source vulnerability.

Figure 24 shows the CVE distribution categorized by the year the CVE was reserved. It is a common misconception to assume that the year reference in a CVE identification represents the disclosure year of the CVE. As we detailed in our 2017 Application Security Research Update \textsuperscript{56}, it is at the subject organization or security researcher’s discretion—the one who found the issue—about when to report and disclose the product’s weakness to the National Vulnerability Database. The flaw’s discoverer also has an option to reserve the ID until he or she is ready to publish the details. Hence the year may reflect the tentative time when the vulnerability existed in the library, but not how long it has been known, or when the fix/patch was made available.

**CVE Count by the Year**

![ CVE Count by the Year Chart ](image)

**Figure 24.** CVE count registration year in referenced libraries

Some of the issues inherited in our data set through the inclusion of vulnerable dependencies, were disclosed more than 15 years ago dating back to 2003.

---

\textsuperscript{56} http://files.asset.microfocus.com/8864/en/8864.pdf
Having said that, some of the issues inherited in our data set, through the inclusion of vulnerable dependencies, were disclosed more than 15 years ago: dating back to 2003. Application owners should really look for alternatives to libraries containing 15-year-old vulnerabilities. Two alternatives may be to look for a fixed version of the same library or use a different library altogether.

Figure 25 shows the top libraries referenced by the number of applications in our data set. Apache commons-beanutils ascended as the top library, replacing last year’s commons-httpclient, now at number four. Commons-beanutils contained one critical CVE (CVE-2014-0114), reserved in 2014. That CVE depicts the Classloader Manipulation vulnerability in the Struts1 library.

To provide a better understanding of vulnerabilities inherited from CVEs, we researched details disclosed about each of the CVEs and mapped them to the Fortify Software Security Taxonomy.

Figure 26 shows the top categories of vulnerabilities from CVEs inherited in applications via referenced libraries. Denial of Service remains the top issue concerning an inherited library. In last year’s report, Dynamic Code Execution: Unsafe Deserialization took the number seven position; this year it rises to number three. In 2015 and 2016, after a few highly publicized exploits, Unsafe Deserialization really got the security community’s focus and thus increased disclosure of this issue. Prior to this weakness type being known by the application security community, we could consider this critical risk an unknown-unknown since enterprises had no way of knowing how many weaknesses of an unknown type existed in the software stack.
Last year, Dynamic Code Evaluation: Unsafe Deserialization had four CVE disclosures in referenced libraries; this year that number has gone up to 18—a quadruple increase. Additionally, 14 of these issues contain CVE identifications reserved in the 2016-2017 timeframe. As we mentioned earlier, it is hard to pinpoint exactly when the issue details became public.

In Figure 11, we show commonly occurring vulnerability categories in web applications. In Figure 27 we represent popular vulnerability categories making libraries vulnerable. Denial of Service is the most pervasive issue, affecting 22% of the libraries and, in turn, projects. Critical input injection issues such as XML External Entity Injection, Cross-Site Scripting, and Unsafe Deserialization also continue to dominate as the top concerns regarding external libraries.
In terms of the number of disclosures in each category of vulnerability, Cross-Site Scripting has been the most popular overall.

**Top 10 Categories by CVE Disclosures**

- **Cross-Site Scripting**: 53%
- **Denial of Service**: 43%
- **XML External Entity Injection**: 41%
- **Directory Traversal**: 22%
- **Buffer Overflow**: 20%
- **Access Control: Authorization Bypass**: 19%
- **Dynamic Code Evaluation: Unsafe Deserialization**: 18%
- **ONGL Expression Injection: Struts**: 12%
- **Weak Encryption**: 12%
- **Other**: 37%

*Figure 28. Top 10 categories grouped by CVE disclosure*
However, as we showed in Figure 24, some of the CVEs that plague web applications in the current data-set were as old as 2003, so looking at all the CVE disclosures in the dataset does not correctly represent what type of issues security researchers are focusing on today. Hence we looked at the CVEs disclosed (or reserved) only in 2016 and 2017.

**Top 10 Categories by CVE Disclosures**

<table>
<thead>
<tr>
<th>Category</th>
<th>CVE Count 2016-2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Overflow</td>
<td>20%</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>14%</td>
</tr>
<tr>
<td>Dynamic Code Evaluation: Unsafe Deserialization</td>
<td>14%</td>
</tr>
<tr>
<td>XML External Entity Injection</td>
<td>12%</td>
</tr>
<tr>
<td>Cross-Site Scripting</td>
<td>11%</td>
</tr>
<tr>
<td>Weak Encryption</td>
<td>9%</td>
</tr>
<tr>
<td>Access Control: SecurityManager Bypass</td>
<td>7%</td>
</tr>
<tr>
<td>Access Control: Authorization Bypass</td>
<td>6%</td>
</tr>
<tr>
<td>Directory Traversal</td>
<td>6%</td>
</tr>
<tr>
<td>Dynamic Code Evaluation: Code Injection</td>
<td>5%</td>
</tr>
</tbody>
</table>

**Figure 29.** Top 10 categories by CVE disclosures made in 2016-2017

It is evident from Figure 29 that an increased focus is on Buffer Overflow and Unsafe Deserialization. Buffer Overflow was number five in the overall top 10 vulnerabilities disclosed in 2016 or 2017. Further, if you are using a SecurityManager component in your Java environment, ensuring a safe configuration and regular patches for it should be a priority for your organization.

Denial of Service also continues to be important. Out of 434 CVEs in the analyzed dataset, 165 were disclosed in 2016 or 2017. Some 155 of these 165 had a CVSS base score of 4 or greater. This means disclosures have gone up both in number and severity.
Conclusion & Next Steps

Reported flaws more than doubled in 2017

Overall, the number of reported vulnerabilities in 2017 was more than double that from the year before, according to the National Vulnerability Database (NVD). A few categories of vulnerabilities appeared to be responsible for a large portion of the change, including disclosed vulnerabilities that led to denial of service, code execution, memory overflows, information exposure, and cross-site scripting. Flaws in web applications accounted for much of the growth.

As the number and types of vulnerabilities continued to evolve in 2017, so did the attack surface of organizations. As a result, defenders need to assess and counter potential weaknesses related to both traditional and newer architectures.

Client/server, web, and peer-to-peer models have existed for a long time. In the last decade we have seen the advent of smartphones, cloud computing, big-data analytics, and large networks of interconnected devices (including supervisory control and data acquisition systems, embedded systems, and the Internet of Things). These newer technologies bring with them a whole ecosystem of new software, communications protocols and hardware—meaning that the classic enterprise perimeter has also expanded. Applications, wherever they reside, are the new perimeter.

An increasing range of regulations and best practices help security practitioners defend against these new threats. Some are industry-specific, like HIPAA for the healthcare business and PCI DSS for financial services. Some go broader, including the General Data Protection Regulation (GDPR)—now mandatory for anyone processing EU citizens’ personal data—and the OWASP Top 10 list of web application vulnerabilities.

Attackers are penetrating further into the hardware and software stack

Attackers are going deep as well as broad, penetrating further into the hardware and software stack. Flaws in open source libraries and components, device drivers, and processors caused billions of dollars in damages and lost information, from large data breaches due to unpatched open source components to the WannaCry and NotPetya ransomware attacks caused from vulnerabilities underlying the EternalBlue exploit of Microsoft Windows implementation of the Server Message Block protocol (SMBv1). Meltdown and Spectre went after major microprocessor families from Intel, AMD, and ARM. Connected devices from IoT to cars were also under attack.

Developers, and development tools, have become targets of attacks as well. The NotPetya ransomware attack initially spread from a Ukrainian tax-software company whose update server had been compromised. The attack underscored that vulnerabilities can infiltrate the software supply-chain and infect unwitting applications up the chain.
Operating systems continued to be a major focus: nine of the top 10 software products with the highest count of reported vulnerabilities were operating systems in 2017.

“Known-knowns” are a problem that can be handled by improved vulnerability management and processes. However, the unknown vulnerabilities—often called zero-day vulnerabilities—are more difficult to mitigate. Processes and tools continue to lack maturity for detecting attacks on vulnerabilities in IoT devices or in firmware.

Hardening systems against the exploitation of unknown vulnerabilities means focusing on better overall security, rather than focusing on specific vulnerabilities. Detection of anomalous activity becomes much more important. Testing both software source code and software operation—two tenets of DevOps—can help companies detect vulnerabilities in their own code. Testing the source code of the dependencies within the software stack for previously unknown weaknesses is also key.

**Fortify on Demand analysis reveals application vulnerabilities**

We analyzed anonymized and sanitized vulnerability data collected by the Micro Focus Fortify on Demand (FoD) platform, and found that 65% of the applications we studied—both web and mobile—contain at least one critical known vulnerability. Security Features again topped the list of web application vulnerabilities. Some 92% of web applications had an issue of this type, vs. 91% last year. Except for the Errors kingdom, every other kingdom saw the number of vulnerabilities go up.

There is a significant jump in the number of unique issues reported over the last year. Last year, we analyzed approximately 7,500 web applications in total and found approximately 2.4 million unique instances of vulnerabilities in all categories combined. This year we analyzed more than 7,800 applications, and the number of unique vulnerabilities jumped to approximately 7.8 million.

With increased focus on SSL/TLS security since the Heartbleed vulnerability, it is not a surprise that Insecure Transport issues dominated the top 10 web vulnerabilities this year, according to our FoD data. Weak SSL protocol saw a 23% jump this year; this could partly be attributed to NIST and PCI DSS guidance against using TLS1.0.

Overall, 81% of mobile applications had at least one critical or high-severity issue, which is up by 15% from last year. Mobile applications showed a different trend than web applications. This year saw a significant reduction in the number of issues in the Errors kingdom for mobile applications, making that kingdom almost negligible.

Consistent with the security issues that have made recent headlines, our data shows that applications—both web and mobile—have a lot of catching up to do with the recommendation of using secure cryptography to secure data at rest and data in transmission.
OWASP Top 10 2017 is only a starting point

The OWASP Top 10 list of web vulnerabilities was refreshed in 2017. The OWASP Top 10 is a starting point for security practitioners. There are hundreds of issues that can affect software security—for instance, there are more than 700 common software security weaknesses identified by the Common Weakness Enumeration (CWE) community and nearly 900 defined in the Fortify Taxonomy of Software Security Errors—so organizations need to look beyond the OWASP Top 10 to forge an effective security program.

Some of the most-cited vulnerabilities didn’t make it into the OWASP Top 10 2017, including Information Leakage/Disclosure (CWE-200), Mass Assignment (CWE-915) and Insufficient Intrusion Detection and Response (no CWE).

According to our analyzed FoD data, a staggering 90% of applications have at least one issue outside of the OWASP Top 10. Furthermore, 49% of tested applications contained a critical or high-severity weakness that is not covered by the OWASP Top 10.

The OWASP Top 10 is a list of web application security risks. While many attacks on organizations do indeed involve the web layer, there are other attack surfaces that need to be considered when formulating a security policy. For example, social engineering, which does not require breaking an application, is frequently used to gain unauthorized access to systems.

There are numerous government and professional resources that can be helpful for formulating a security policy, such as the Defense Information Systems Agency’s Security Technical Implementation Guides, FIPS 200, NIST Special Publication 800-53 R4, and the ISO 27000 series.

GDPR is forcing strong protection of customer data

No security policy is complete without strong protection for customer data. The EU’s General Data Protection Regulation (GDPR) is a best-practice privacy framework on how to approach encryption and other recommended measures. On May 25, 2018 it became law for anyone doing business involving EU citizens, but it is also a useful measure for any organization that wants to protect its customers’ data.

A major driver behind the GDPR is the principle that data related specifically to a person belongs to that person—not to the organization creating, possessing, or processing it. As a custodian of EU citizens’ personal data, the organization is responsible for protecting the data under the framework laid out in the GDPR.

Personal data is essentially any information that relates to an identified, or identifiable, living individual and may include a name or surname, home address, location data, and IP address.

Businesses cast a wide net to collect data for analytics and could include EU citizen personal data that would require protection under the GDPR. In practical terms, the GDPR necessitates a complete rethinking of data-handling processes, forcing a privacy-by-design methodology wherever personal data is collected and stored—regardless of the technology used. Although the GDPR does not specify the technologies
required to enable compliance, it strongly hints at the use of encryption and pseudonymization as acceptable approaches to protect personal data.

Encryption and pseudonymization are very generic categories, and there are many variants with differing levels of coverage, capability, flexibility, and manageability. It can be useful to think of encryption as taking place either relative to, or at different levels of, the TCP/IP stack, thus creating a notional "encryption stack" that closely parallels the TCP/IP stack. There are good reasons to encrypt at different places relative to the TCP/IP stack, but it is important to understand that when you encrypt at a particular place in the stack, the encryption protects only against threats that target layers at or below where the encryption takes place.

**Next steps for protecting your organization**

A. Encrypting at the application layer is the only form of encryption that will address these serious threats. The biggest and most severe data breaches that have affected both the public and private sectors all operate at the application layer. This includes almost all versions of both malware and advanced persistent threat (APT) attacks.

B. Encryption plus strong key management is required to ensure data security. Encryption alone will not suffice to ensure data security. It must be supported by strong key management for the encryption to be useful at all. In a nutshell, key management is all aspects of encryption that do not directly relate to encrypting or decrypting data. It includes generating keys, storing keys, transporting keys, and so on.

C. Take a holistic view about application security. For a comprehensive assessment of data security, it is important that organizations take a holistic view and include application security and vulnerability assessment as part of their software development program.

Authors:
Harley Adams
Alexander M. Hoole
Sejal Kamani
Robert Lemos
Luther Martin
John Mello
Nidhi Shah
Stan Wisseman

Contributors:
Matt Gibbs
Sonatype Inc.

**Learn More At**