

Peril in a Pandemic:

The State of Mobile Application Security

A security analysis of the
most popular Android apps
during the COVID-19
pandemic



CyRC

Executive summary

The Synopsys Cybersecurity Research Center (CyRC) analyzed more than 3,000 popular Android applications to assess the state of mobile app security during the COVID-19 pandemic. The study targeted the most downloaded and highest grossing apps across 18 categories, many of which have seen explosive growth during the pandemic. The research focused on three core areas of mobile app security:

- **Vulnerabilities:** The presence of known software vulnerabilities in the applications' open source components
- **Information leakage:** Sensitive data such as private keys, tokens, and passwords exposed in the application code
- **Mobile device permissions:** Applications requiring excessive access to mobile device data and features

The analysis reveals that the majority of apps contain open source components with known security vulnerabilities. It also highlights other pervasive security concerns including myriad potentially sensitive data exposed in the application code and the use of excessive mobile device permissions.

For consumers, this report highlights the jarring reality that even the most popular mobile apps are not immune to security and privacy weaknesses and should not be trusted implicitly. For app developers, this underscores the urgent need for secure software development practices and better overall privacy and security hygiene.

3,335 mobile apps analyzed



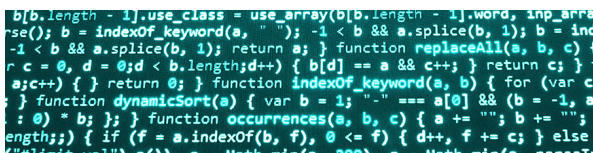
18 of the most popular app categories during the pandemic, including games, education, banking, and health & fitness

63% of the apps contained known security vulnerabilities



- Average of **39** vulnerabilities per app
- **44%** of the vulnerabilities are considered high risk
- **94%** of the vulnerabilities have publicly documented fixes
- **73%** of the vulnerabilities were first disclosed more than two years ago

Thousands of pieces of sensitive data exposed in the application code



- Passwords, tokens, and keys: **2,224**
- Email addresses: **10,863**
- IP addresses and URLs: **392,795**

Excessive use of mobile device permissions



- Normal permissions: **33,385**
- Sensitive permissions: **15,139**
- Permissions not intended for third-party use: **10,653**

Table of contents

- Peril in a Pandemic: The State of Mobile Application Security 1
- The what: Targeted analysis of mobile applications 3
- The how: Synopsys' industry-leading Black Duck Binary Analysis 3
- The results: What Synopsys found 4
 - Open source vulnerability findings 4
 - Known vulnerabilities 4
 - Findings by application category 5
 - Analyzing vulnerabilities 6
 - Vulnerability deep dive with Black Duck Security Advisories 7
 - Vulnerabilities with proven exploit risks 8
 - Detailed open source vulnerability findings 9
 - Information leakage findings 10
 - Mobile permissions findings 11
- Summary 15

Peril in a Pandemic: The State of Mobile Application Security

In this challenging time, limitations driven by social distancing and lockdowns have moved the world online in remarkable ways, perhaps forever changing the way we work, learn, and interact. Society has quickly adapted, making resources traditionally available only in the physical world accessible virtually. The result is a culture increasingly reliant on mobile applications to conduct daily activities.

Through the lens of the COVID-19 pandemic, the Synopsys Cybersecurity Research Center (CyRC) set out to explore the state of application security in this increasingly application-driven world. It boiled its analysis down to **two key questions:**

- **Are popular mobile applications reasonably secure, or do they represent low-hanging fruit for attackers?**
- **Do app developers prioritize security and privacy when determining which device permissions and data their applications can access?**

Whether you're a consumer of mobile applications, a developer, or both, it's important that you understand the relative risk you take on when moving your life, or your business offerings, online. By using the Synopsys software composition analysis (SCA) tool Black Duck® Binary Analysis to perform a thorough analysis of the most popular mobile applications, Synopsys offers a detailed glimpse into potential risks, implications, and dangers in this new remote lifestyle era.

"As the world continued to cope with the impact of the coronavirus outbreak, the second quarter of 2020 became the largest yet for mobile app downloads, usage, and consumer spending. According to new data from app store intelligence firm App Annie, mobile app usage grew 40% year-over-year in the second quarter of 2020."¹



The CyRC analyzed 3,335 of the most popular free and paid Android applications on the Google Play Store as of Q1, 2021.

Number of apps scanned, by category

Top free apps 315	Top-grossing apps 300	Top free games 288	
Top free dating apps 277	Top-grossing dating apps 267	Top-grossing games 257	
Top paid apps 216	Productivity 158	Food and drink 153	
Top paid games 211	Health & fitness 150	Budgeting 112	Banking 107
Educational apps 159	Entertainment 129	Tools for teachers 101	Lifestyle 45
		Payment 90	

Average number of open source components per app, by category

Top free games 27.8	Top-grossing games 29.9	
Budgeting 24.3	Banking 26.4	
Payment 21.9	Top-grossing apps 22.8	
Food and drink 18.4	Entertainment 20.1	Top free apps 20.9
Health & fitness 17.7	Top paid games 18.1	Tools for teachers 18.2
Lifestyle 16	Top-grossing dating apps 16.9	Educational apps 17.0
Top paid apps 11.8	Top free dating apps 15.5	Productivity 17.0

Business, health and fitness, and education applications saw quarter-over-quarter growth in downloads of 115%, 75%, and 50% respectively on Google Play.²



The what: Targeted analysis of mobile applications

To examine the “COVID state” of mobile application security, the CyRC analyzed 3,335 of the most popular free and paid Android applications on the Google Play Store as of Q1, 2021. The applications span a range of categories that have experienced significant growth due to the pandemic, including education, entertainment, games, health and fitness, food and drink, productivity, and tools for teachers.

The CyRC leverages industry-leading expertise, technology, and resources to help share software security knowledge and best practices. The CyRC team used Black Duck to scan and analyze these top applications for three key areas of potential security risk:

- **Open source security vulnerabilities:** Are there known security vulnerabilities lurking in the open source components of the applications you use every day?
- **Potential instances of information leakage:** Did the app developers inadvertently leave sensitive data such as passwords, email addresses, or AWS and Google Cloud keys exposed in the compiled application?
- **Mobile permissions:** What permissions does an application require from a device? Are there more permissions than necessary or are there permissions that could compromise data if used improperly?

The how: Synopsys’ industry-leading Black Duck Binary Analysis

To conduct a robust investigation, the CyRC used Black Duck Binary Analysis, a unique feature of the Black Duck tool, to analyze the open source software components and other contents of the Android packages associated with each application. Because Black Duck analyzes apps as compiled binaries (as opposed to source code), it can scan virtually any software, from desktop and mobile applications to embedded system firmware.

Black Duck not only identifies open source and matches it to vulnerabilities and licenses, it also pinpoints potential risks to sensitive information, like information leakage and mobile permissions that could impact the security of the application.

The results: What Synopsys found

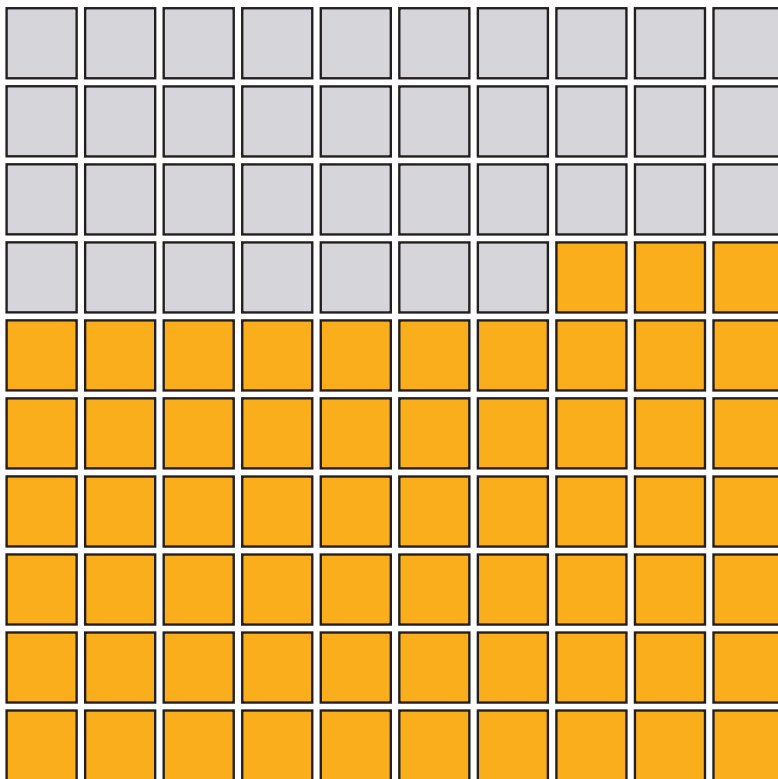


The CyRC analyzed a total of 3,335 applications and uncovered noteworthy risk findings relevant to both application developers and the organizations they work for, as well as to the consumers of these applications. Whether you are a developer or a consumer (or both), the analysis unearthed information that affects the security of the applications being developed and used.

Open source vulnerability findings

Overall, 98% of the applications scanned contain open source components. This number is consistent with other studies by Synopsys, and it emphasizes an overall trend well-known to be true: open source forms the foundation of nearly every application today.

The presence of open source itself does not present risk as long as it's actively managed and maintained. These findings, however, reveal that 63% of the applications scanned contained open source software components with at least one known security vulnerability, and an average of 39 vulnerabilities per vulnerable application. While some app categories were better than others, none were completely immune. At least one-third of the applications in all 18 categories contained applications with known security vulnerabilities.



Known vulnerabilities

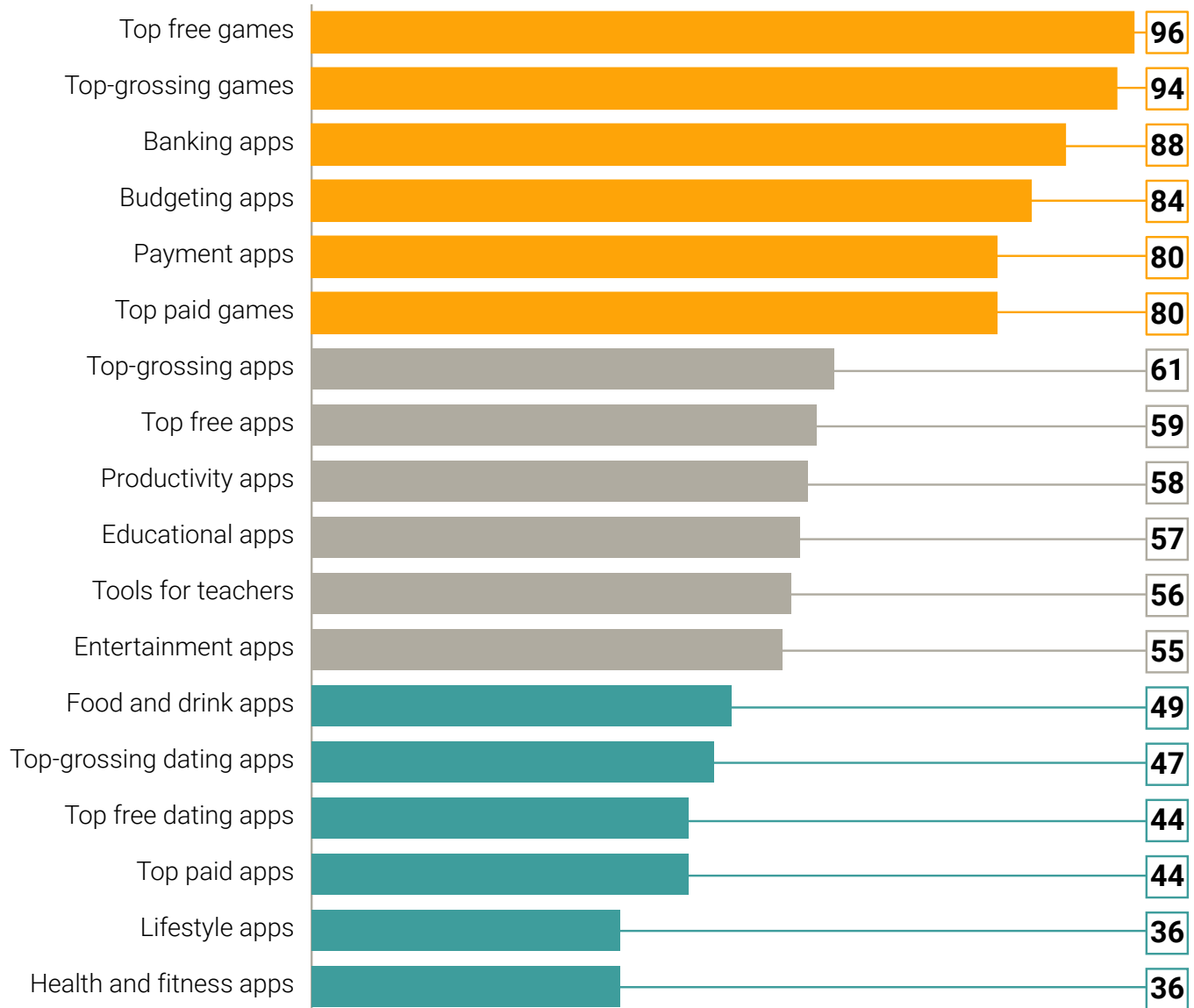
Of the 3,335 applications the CyRC scanned using binary analysis, 2,115 contained vulnerable components (63%) with an average of 39 vulnerabilities per vulnerable app. In total, the vulnerabilities that were discovered occurred over 82,000 times across all applications. This means the majority of the top Android apps used today have some sort of vulnerability, regardless of application category (lifestyle, finance, etc.) or who developed it.

The prevalence of these vulnerabilities is concerning for consumers who actively use a number of popular apps because their personal information and security is at risk. It's concerning for application owners (developers/their organization) because their organization's data, employee data, customer data, and reputation are all at risk.

Findings by application category

To contextualize the troubling findings noted throughout this report, the CyRC filtered results by application category, shedding light on those categories found to be most vulnerable. Most concerning was the volume of vulnerabilities found in financial, gaming, and productivity applications—all categories that have gained immense popularity during the pandemic.

Percentage of scanned apps that contained vulnerable components, by category



In analyzing the overall Common Vulnerabilities and Exposures (CVE) data, the most dramatic findings resulted from analysis of financial and banking applications. Of the 107 banking applications scanned, 94 contained a vulnerability—that's 88%, well above the average of 63%. With a total of 5,179 vulnerabilities identified, the average application contained 55 vulnerabilities. Financial applications require some of the most personally sensitive data, making these numbers alarming due to the potential impact of a security breach.

Gaming applications had similar results. Top-grossing and free games contained 10,404 and 9,829 vulnerabilities in the respective 257 and 288 applications scanned. That translates to 94% of top-grossing games, and 96% of top free games, containing vulnerabilities. The average top-grossing and free applications have 43 and 35 average vulnerabilities per application respectively. With gaming's leap in popularity during COVID, and especially given the younger and more trusting gaming audience, this finding raises concerns regarding children's relative level of risk using online applications and what personal data could be exposed in the event of a breach.

Productivity applications such as those used for personal and professional business activities (spreadsheets, documentation) yielded similar if slightly less dramatic results. Of the 158 applications scanned, 92 contained vulnerabilities (58%). That's lower than the overall average, but 4,787 vulnerabilities were identified, which translates to 52 average vulnerabilities per vulnerable application, which is quite high. Suffice to say, these applications have some work to do to ensure users' security.

Analyzing vulnerabilities

In addition to conducting research for reports such as this one, the CyRC also began publishing Black Duck Security Advisories (BDSAs) in 2018. BDSAs are highly detailed vulnerability records that have been researched, enhanced, and augmented to provide thorough analysis of open source security concerns, beyond what is available in the National Vulnerability Database. BDSAs also offer remediation and workaround information, Common Vulnerability Scoring System scores, exploit information, and Common Weakness Enumeration classifications.

In order to leverage this enhanced data for this report, CyRC's more detailed vulnerability analysis focuses on the vulnerabilities for which BDSA records exist—namely the vulnerabilities that were disclosed after 2018.

When assessing the CyRC's BDSA coverage for the vulnerabilities identified in this mobile app research, it also became clear that nearly three-quarters of the vulnerabilities were disclosed prior to 2018. In other words, for the most part, these are not new issues, and developers simply aren't considering the security of the open source components they use to build their apps.

3,137

unique vulnerabilities identified by CyRC in mobile app research

2,285 (73%)

were disclosed 2+ years ago

852 (27%)

vulnerabilities disclosed between 2018 and 2021

782 (25%)

vulnerabilities with a BDSA record

Vulnerability deep dive with Black Duck Security Advisories

Looking at the BDSA records, the CyRC found that just over 94% of the 782 vulnerabilities were fixable, meaning there are security patches or more-secure versions of the affected open source components available. Only 5.75% of the 782 had no known solution. Why then have organizations not addressed them? Vulnerability remediation is always important and should be a high priority for developers.

The applications the CyRC studied are in high demand, making their security even more critical. Failure to address this leaves popular applications and their broad user bases vulnerable to exploit. While it's nearly impossible to address all vulnerabilities in an application immediately, those with the greatest probability of exploit should be prioritized. To understand why these vulnerabilities have remained in these applications, let's examine what a development team needs before it can apply a fix.

What a development team needs before it can apply a fix



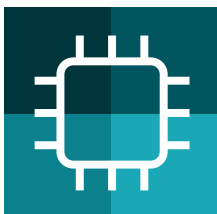
Information sources: Open source software, unlike commercial software, relies on the consumer to pull patches rather than have them pushed to them. To do this, teams need to have reliable and diverse sources for vulnerability data, and ideally have that information pushed to them via the alert systems they use every day (i.e., email, Slack, Teams, etc.).



Discovery and identification of vulnerable components: To fix an open source vulnerability, teams have to first know an affected component is there. Pinpointing vulnerable components in applications depends on first identifying and inventorying all the open source components in applications.



A way to know what to fix first: Armed with an inventory and a list of known vulnerabilities, teams often lack the time and resources to fix everything immediately. Being able to leverage additional information about the vulnerabilities, such as their severity, exploitability, exploit impact, and reachability within the application can all be vital to ensuring resources focus on the most critical first.



Complete patch information: After the team is aware of the vulnerabilities in the components it's using and prioritizing what to fix first, the final core piece of information is how to actually make the fix. Clear and concise patch information attached to the alert arms the team with the information it needs to quickly apply the fix.

Vulnerabilities with proven exploit risks

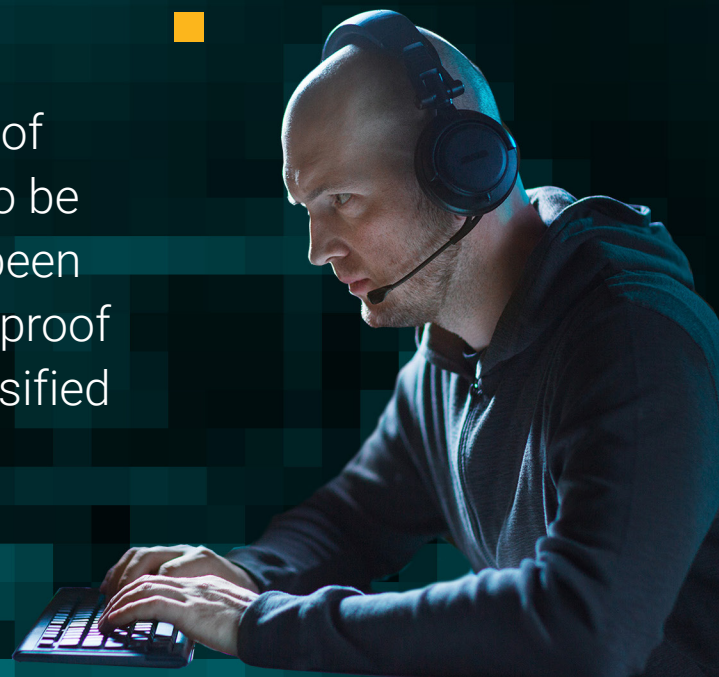
The CyRC considers close to half of the vulnerabilities identified (44%) to be high risk because they either have been actively exploited, have documented proof of concept (PoC) exploits, or are classified as remote code execution (RCE) vulnerabilities. Unlike potential or unverified risk factors, these vulnerabilities represent a tangible threat.

To simplify, a proof of concept (PoC) exploit is a script, file, or piece of code that can be used to reproduce the vulnerability but would require modification by a skilled attacker to be used to attack a target in a real-world scenario. An exploit is similar to a PoC, but it can be used in a cyber attack with little or no modification. RCE refers to a remote attacker executing code directly on the targeted system. BDSAs document RCE vulnerabilities because RCE vulnerabilities that have a PoC or exploit available indicate that this type of attack (by a remote attacker) is in fact possible for the given vulnerability, making it critical to address. The CyRC found this to be the case for only five of the vulnerabilities identified.

The vulnerabilities that have either a PoC or exploit, or that are classified as an RCE vulnerability, are inherently risky and represent low-hanging fruit for attackers. They aren't just potential risks—they are real threats to the application's integrity and the security of the user. The CyRC also found that although these vulnerabilities were serious in nature, the majority also had a known and published solution to mitigate the risk. All the RCE vulnerabilities have a known solution, and 95% of those with a PoC exploit have a current solution.

The longer a vulnerability remains unaddressed, the more likely it is to be exploited, and the more difficult it can become to make the fix. This is especially true for the more serious vulnerabilities with proven exploits. With an increasing consumer base, the potential consequences of neglecting to fix these vulnerabilities is concerning. Failure to remediate vulnerabilities that have a known fix is always a questionable security practice—and exceedingly risky, given how frequently cyber attacks are linked to unpatched open source software components.

The ability to address these risks, at scale and pace, in an increasingly fast and demanding environment comes down to having the right tools and processes in place to evaluate the applications, and having access to the information needed to understand, prioritize, and fix security vulnerabilities.



The CyRC considers close to half of the vulnerabilities identified (44%) to be high risk because they either have been actively exploited, have documented proof of concept (PoC) exploits, or are classified as RCE vulnerabilities.

Detailed open source vulnerability findings

The CyRC grouped its vulnerability findings by category, helping to provide more detailed insight into which apps pose the greatest security risk. The key findings, as well as a detailed data breakdown by category, are as follows.

Education: During COVID, educational applications have been instrumental in enabling remote schooling. Concerningly, Synopsys found that educational apps have the highest number of total vulnerabilities, as well as the highest percentage of vulnerabilities with a PoC, exploit, or RCE. Fortunately, these applications don't fall within the top three categories of vulnerable applications without solutions, but it raises questions regarding security practices. Why have these vulnerabilities with known solutions not been fixed? And how large would the impact be if they were breached?

These applications suddenly became very important in the context of COVID, and consumers trusted them with a good deal of personal information. Culturally, we associate education with trust and let our guard down, assuming the security of educational tools. We also trust that our children will be safe using these applications, so these findings are particularly distressing.

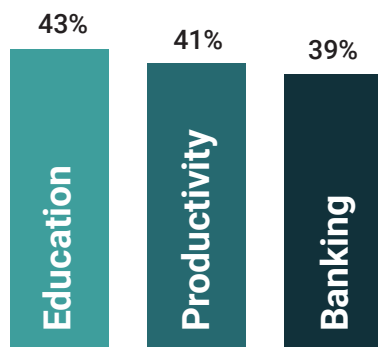
Banking: Another troubling discovery the CyRC made is that banking application vulnerabilities are in the top three for the highest number of fixable and nonfixable vulnerabilities. There's much at stake when it comes to financial data; we trust sensitive personal information to these apps. With the implementation of a security solution that can point them toward available vulnerability solutions, developers could easily knock out almost 40% of the vulnerabilities found in this study. By prioritizing vulnerabilities and spending valuable resources on the most pressing and potentially dangerous vulnerabilities, developers could find workarounds to close security gaps.

The specifics of all categories analyzed

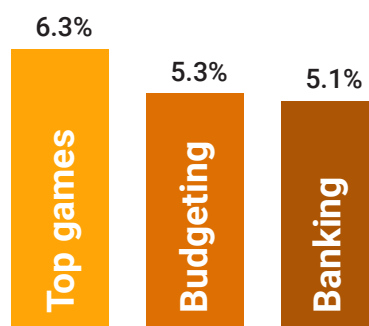
Of all vulnerabilities with a corresponding BDSA:



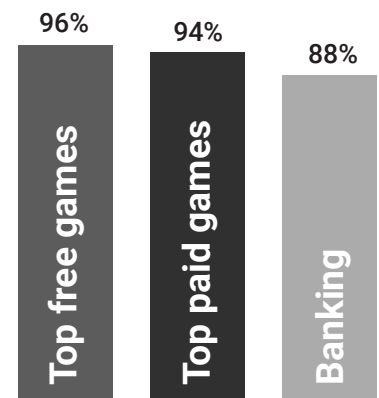
The categories with the highest percentage of exploitable BDSAs with fixes available:



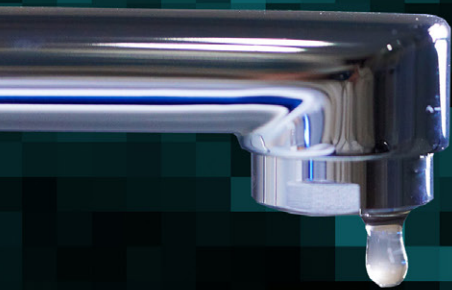
The categories with the highest percentage of exploitable BDSAs with no fixes available:



The categories with the highest percentage of applications containing vulnerabilities:



Detailed information leakage findings



JSON Web Tokens (JWTs)

65

AWS keys

26

Twilio tokens

406

Google Cloud tokens

804

Facebook tokens

27

Asymmetric private keys (RSA)

60

OAuth tokens

817

Email addresses

10,863

IP addresses

27,568—4 of which are marked
suspicious

URLs

365,227—11 of which are
marked suspicious by Google
Safebrowsing

Information leakage findings

Put simply, information leakage is when developers accidentally leave personal or sensitive data in the source code or configuration files of the application. In the wrong hands, this information can be used maliciously. The CyRC's findings indicate that popular applications are not free from information leakage.

To perform this analysis, the CyRC used Black Duck to examine key types of information leakage. To better understand the implications of these findings, let's examine what each type of leakage entails.

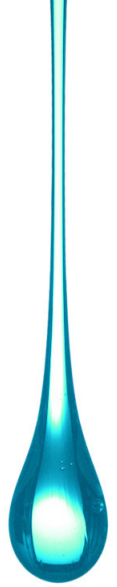
Tokens, keys, and passwords: If developers leave behind this type of information (AWS keys, Google Cloud tokens, user credentials, etc.), it can pose great potential risk. This information allows an individual to access someone's servers, systems, or sensitive properties. From there an attacker can steal IP, plant malware, or launch compute resources that attribute costs to the application owner.

For example, JSON Web Tokens (JWTs) serve as a way to securely pass information between parties. Although these can be encrypted, they often aren't. JWTs require a digitally signed secret key and essentially act as credentials, so they should never be kept longer than absolutely necessary. When left behind in source code, they can be easily decoded to reveal information that helps an attacker exploit the application. The CyRC found four JWTs in banking apps, and three in budgeting apps—a big cause for concern.

IP addresses and URLs: If applications point to certain URLs/IPs, they can unwittingly enable detrimental activities. For example, some URLs/IPs can actively serve malicious content, or content otherwise inappropriate. Other URLs may simply disclose private APIs, internal systems, or hidden vendor resources, offering malicious actors an open attack vector. Either way, the usage of them should trigger red flags and questions.

Email addresses: Email addresses accidentally left behind in source code can unintentionally disclose internal systems (e.g., domains) and usernames. They can then be used as attack points for usernames of systems (especially when combined with tokens or IP addresses) or for phishing attacks.

A combination of any of the above: When an attacker gathers bits and pieces of information, it can be used in combination with other data they possess to fill in gaps and form a picture that allows access. For example, combining emails and usernames with URLs for remote APIs and passwords provides an easy route for exploit. Leaked information of any kind should be prioritized as a high security risk, with potentially devastating consequences for developers and users alike.



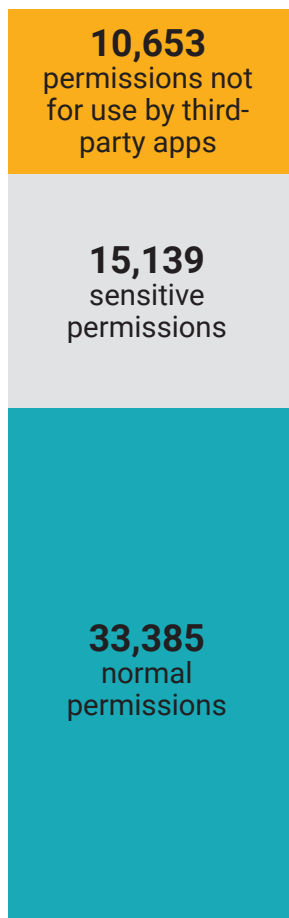
An information leakage story

In an incredibly timely real-world example of information leakage, the company SolarWinds recently fell victim to a supply chain attack. As announced on February 26, 2021 careless information security practices by a former intern exposed a critical—and ridiculously simple—internal password (solarwinds123).

Once that password was compromised, suspected Russian hackers were able to access a system that SolarWinds used to assemble updates to Orion, one of its flagship products. From here, the attackers inserted malicious code into an otherwise legitimate software update. Once SUNSPOT malware was inserted into the codebase, it was able to monitor and identify running processes that were involved in the compilation of Orion, and replace source files to include SUNBURST malware, which is a malicious version of a legitimate Orion plugin. Taking great care to not break any Orion builds, and by blending with other SolarWinds network traffic, the hackers were able to escape the notice of development teams. Once the Orion updates were deployed to an estimated 18,000 customers, SUNBURST sent information back to the attackers that was used to identify targets of additional malware, broadened access, and spying. Among these targets were several Fortune 500 companies, including Microsoft and Intel, and government agencies, including the Department of Homeland Security and the Treasury Department.

Through the lens of this story, the implications of information leakage are clear. The large number of instances of potential information leakage found by the CyRC reveals a troubling trend. Failure to address information leakage security concerns leaves both an application and its users vulnerable.

By analyzing applications in their compiled states before deployment, tools like Black Duck can help easily identify and mitigate risk introduced by information leakage. In doing so, an organization can reinforce its security posture and avoid costly mistakes like that of SolarWinds.



Mobile permissions findings

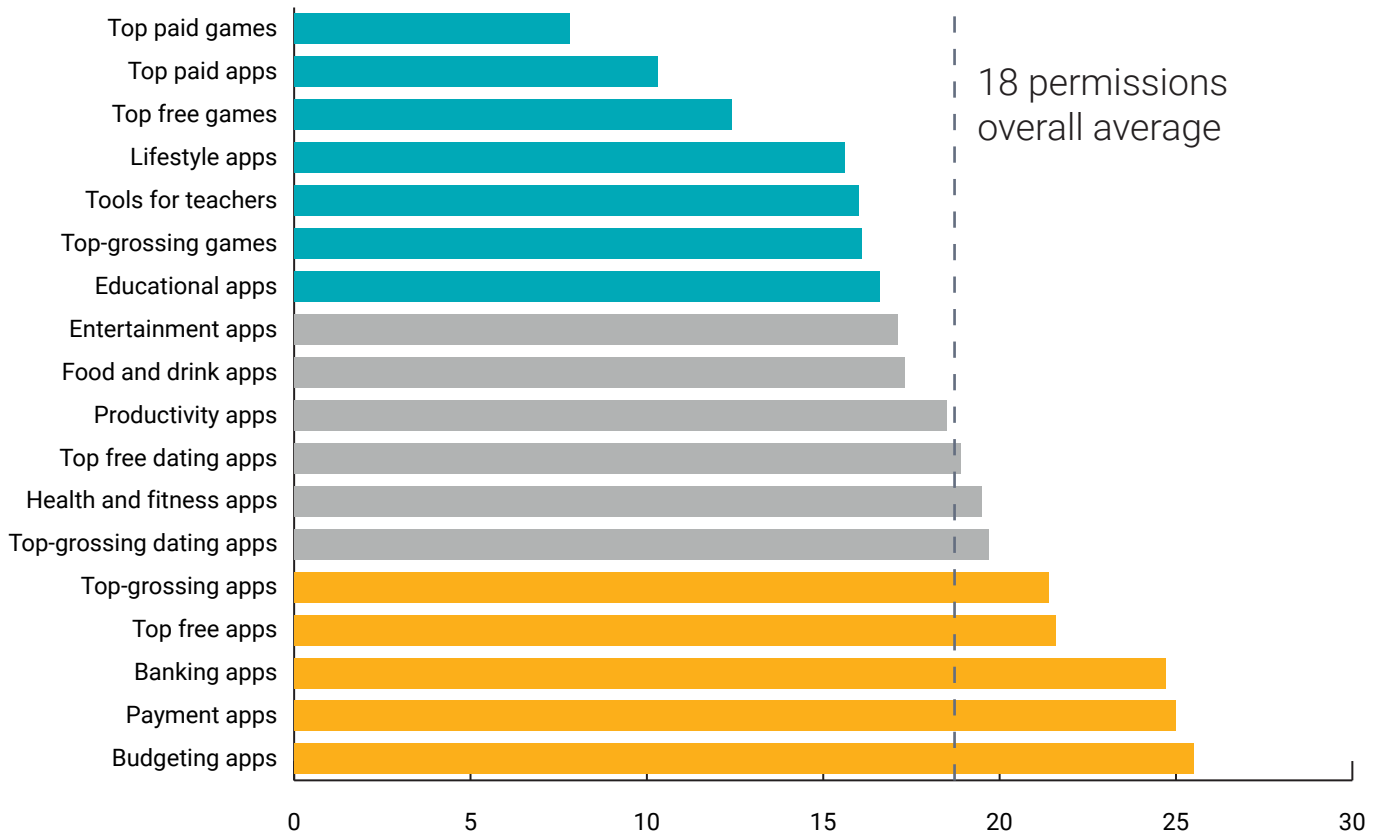
Consumers need to be aware of what personal data they are allowing apps to access. They also need to understand what level of permission they are granting, and the potential implications of a hacker getting ahold of that information. It could range from running up a giant phone bill, to accessing their home security system, to even gaining access to their personal location data.

Businesses and developers need to be aware of what third-party mobile applications they have embedded in their hardware. They need to understand the potential security gaps and the amount of personal data that third-party applications can access—and it's vital to know the associated level of liability. And from a supply chain perspective, when procuring third-party apps, businesses and developers must ensure security without access to source code.

The CyRC used Black Duck to examine the mobile permissions tied to top Android applications. The team first reviewed the average number of permissions required of a user, per application, both by category and as a whole. The CyRC then looked at specific applications with results that were outside of that average number by more than two standard deviations. Special attention was given to those that asked for significantly more permissions than the average application.

The CyRC found an average of 4.5 sensitive permissions per application and 3 not for use by third-party apps (apps not developed by the provider of the OS). These numbers seem unnecessarily high given the personal data and control they require users to relinquish.

Average number of permissions required by an application



Detailed mobile permissions findings

After conducting the analysis, the CyRC determined that in total, 111 applications required far more permissions than the average of 18 per application (about 3% of our study group).

Certain categories of applications had higher percentages of outliers (the total percent of apps with far more mobile permissions than the average):

- Tools for teachers: 7%
- Health and fitness apps: 6.5%
- Top-grossing dating apps: 6%
- Educational apps: 5.6%

Certain categories had a higher than average number of permissions:

- Budgeting apps: 26 permissions on average
- Payment apps: 25 permissions on average
- Banking apps: 25 permissions on average

In order to get a clearer understanding of what these excessive permissions required, the CyRC did a deep-dive into two categories that have experienced increased demand during COVID—tools for teachers and health and fitness.

Tools for teachers: This is a very popular category given that remote learning during COVID is heavily reliant on technology for teachers and children of all ages. The CyRC found that outliers in this category required 26 or more permissions to use the application.

One application with over 1 million downloads required 11 permissions that Google classifies as “Protection Level: Dangerous.”

This is greatly concerning, as Google classifies only 32 types of permissions as dangerous. Any application asking for 11 potentially exposing bits of information or access should be seriously reconsidered. These dangerous permissions are also known as runtime permissions, and they allow the application to access restricted data and perform restricted actions. Many allow access to private or sensitive user data.

In this application, there were some permissions that should raise red flags for parents, particularly the Access Fine Location permission. There might be use cases when the application actually needs this level of intelligence, but in the event of a hack, the child’s exact location would be readily available. App developers have other options; the Access Coarse Location permission provides approximate location data without being too precise, so it may be more appropriate when it comes to minors.



Health and fitness: As workouts have moved online and into the home, the CyRC set out to explore security in health and fitness applications. The CyRC found that outliers in this category require 38 or more permissions.

One application with over 5 million downloads required a total of 56 permissions, 31 of which Google classifies as “Protection Level: Dangerous” or as signature permissions that are not to be used by third-party apps.

These permission levels likely indicate that this is a particularly poorly developed application, and the developers didn't fully understand Android permissions. And while the intent may not have been malicious, the result is still potential danger and exposure for users.

Permissions not for use by third-party apps included:



Factory Test

The “Factory Test” permission gives the application the ability to run as the root user, which gives the application access to the entire phone. No other application analyzed included this permission. This permission also requires “read logs,” which enable the application to read system log files filled with users’ private information. This type of permission should never be used by third-party apps. There are only two reasons for this permissions issue: either a developer made mistakes when setting up this application, or there is malicious or suspicious behavior occurring. Even if it were a development mistake, this type of overarching permission could be exceedingly harmful in the case of a hack. The hacker would have full access to the mobile device, along with access to the data of 5 million users of this application.



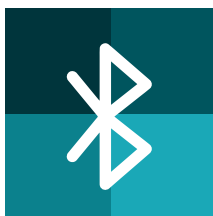
Call Privileged

The “Call Privileged” permission enables the application to call any phone number, including emergency numbers, without having to go through the phone’s dialer or require the phone’s owner to confirm the call. This could result in costly behaviors such as international calls. It’s not necessary for an application to require this type of permission in order to achieve its desired functionality. It seems that this again is a case of developers not understanding the permission levels and associated consequences written into the application.



Process Outgoing Calls

The “Process Outgoing Calls” permission allows the app to see numbers dialed in outgoing calls, with the option to redirect calls. It’s hard to find a reason a health and fitness app would need this level of access.



Bluetooth Privileged

The “Bluetooth Privileged” permission allows the application to pair Bluetooth devices without any user interaction, and allows contacts and messages access. This is simply excessive and unnecessary access.

In all, this application required 10 permissions that are deemed not for use by third-party apps. The invasive and clearly unnecessary level of access requested by this application is an indication of weak security and poor coding practices. Many of these permissions have alternatives that would achieve a similar desired functionality without being as invasive for users. Further, users should ask why a health and fitness app needs to redirect phone calls or gain access as the root user of a phone.

While these egregious examples were found in some very popular applications, the overall data shows that every category contained applications with problematic permissions. The CyRC makes no determination about the intent of these potentially dangerous permissions, but seeks only to warn users and developers. Users need to understand what level of access they are providing an application to ensure the security of their data. Developers need to understand the permissions they are including and explore potential alternatives that could be less harmful if exploited. This is a key step to ensure application security.

Summary

Undoubtedly this immersion into the COVID state of application security has sparked both concern and questions. As a developer or businessperson, you might have questions about how to access and reinforce your own application security. As a consumer, you might have questions about your relative level of current risk and how much of your data you want an application to have access to.

It's important that consumers understand what information they may potentially release online. You may consider increasing the scrutiny you apply when agreeing to new and updated terms of conditions and permissions when you download an application. Does your child need an app that requests their precise physical location? Or access to their camera? Are you sure you need to allow your fitness app to access your contacts and phone call privileges? When consumers demand security and privacy, vendors tend to deliver.

Developers should also know what they can do to arm themselves and their team against avoidable security oversights and coding mistakes. Solutions like Synopsys' Black Duck SCA and Black Duck Binary Analysis keep you informed about open source vulnerabilities, potential instances of information leakage, and mobile permissions information. Armed with these tools and the information they provide, you can take informed actions and streamline application security.

Leveraging a robust AppSec solution capable of analyzing your open source, proprietary, and compiled binaries and that provides real-time and actionable remediation advice can set you up to win and keep your apps and their users secure.

To learn more about Black Duck Binary Analysis, view our [webinar](#) or visit our [website](#).

¹ Sarah Perez, [Coronavirus impact sends app downloads, usage, and consumer spending to record highs in Q2](#), TechCrunch, July 9, 2020.

² Ibid.

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

About the CyRC

The Synopsys Cybersecurity Research Center (CyRC) works to accelerate access to information around the identification, severity, exploitation, mitigation, and defense against software vulnerabilities. Operating within the greater Synopsys mission of making the software that powers our lives safer and of the highest quality, the CyRC helps increase awareness of issues by publishing research supporting strong cyber security practices.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.

185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

Contact us:

U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com