The iOS Need for Mobile Threat Protection

Nikias Bassen

Principal Security Researcher, zLabs
@pimskeks

Zuk Avraham

Chairman, Founder and CTO
@ihackbanme

Contents:

Executive Summary

Today's mobile workforce relies on devices from three main players: iOS, Android and Windows Phone. The first two have achieved significant market share. Android security, in particular, has been the subject of extensive research, and iOS benefits from a popular assumption that it is safe from the same or similar risks encountered on Android. The number and severity of the vulnerabilities that have been discovered, disclosed, and demonstrated in iOS, however, is significant. We'll discuss the main attack vectors that expose most of the risk in iOS environments.

In this whitepaper we'll answer key questions:

- What is the weakest link?
- Are iOS-only environments safer?
- Is the "walled garden" protecting me from cyber espionage?
- Threat Landscape
- Patching and updates
- Solution

Who's the Weakest Link?

As we approach mobile security from a risk perspective, a general assumption is that iOS devices are safer than other smartphones mainly due to one of iOS' key features known as the "walled garden". This "walled garden" approach is a closed iOS app ecosystem where only apps that have been approved by Apple can be run on an iOS device. It is one of Apple's security controls that helps keep malicious code and malware from entering the ecosystem. However, the "walled garden" does not play a significant role in targeted attacks. Also, it has been proven that the system's security mitigations have not been sufficient to block attacks, as evidenced by high profile threats, such as XCodeGhost, that made its way into apps in the App Store (further discussed in the "Wall Garden" chapter).

If we also consider that most critical infrastructure, popular services, vendors and carriers are being constantly targeted by government agency and professional hacking groups (e.g., Hacking Team), we can safely assume that if your corporation is holding confidential information, has access to critical infrastructure, or is simply "of interest", then the issue is not so much if you are going to get attacked, but when.

Since such an event is likely to happen, we'll assess, based on recent leaks and threat intelligence, who within an organization is most likely to be targeted.

Users of Interest

From reviewing the recent leaks, conversations and threat intelligence from our clients, it appears that the following users are mainly targeted:

- IT Admins
- People of Interest
- C-Level Executives
- Traveling Executives

We'll walk through a description of each target and explain why these respective roles increase the chance of being targeted.

IT Admins

IT Admins are the kings of the kingdom. They grant permissions and certificates that can benefit any malicious entities. Instead of trying to break into all systems, an attack on the IT staff would provide the attackers with the keys to the kingdom – certificates, passwords and complete understanding of a network structure.

People of Interest

Threats to this group are more intricate. Everyone who carries or has access to sensitive information (from the CFO, to sales reps with access to a CRM) would be a valuable target for an attacker. The idea here is similar to the IT Admin scenario but requires aiming at an individual with access to a specific file or email account.

C-Level Executives

C-Level Executives have access to sensitive and strategic corporate information -- from revenue to bidding proposals sent by email. Attackers would want to gain access to these highly ranked executives by either breaking directly into their mobile devices, or through one of the other scenarios.

Traveling Executives

This type of attack is used more by government spying agencies than professional hacking groups. A recent Hacking Team leak, though, indicates that the boundaries between both types of attackers may not be clearly defined. This attack is so popular that it even got a name (evil-maid).

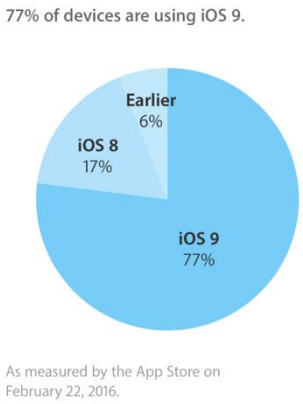
iOS Fragmentation

While the Android ecosystem is much more diverse and fragmented than Apple's, the latter still faces an issue associated with securing a significant number of devices. The latest official Apple iOS market share numbers were published on February 22nd, 2015.

Analyzing the graph:

The information displayed provides some insights. Almost one of every four (~23%) iOS users is running outdated versions that are vulnerable to publicly available exploits. On the positive side, though, it shows that 77% are running iOS 9+. However, this information is misleading.

Based on Apple's security update records, iOS 9.0 and iOS 9.1, 9.2 and 9.2.1 have had many security vulnerabilities. The following graph describes the types of vulnerabilities by version, including versions 9.0, 9.1 and 9.2 that represent a significant portion of iOS 9 users.



iOS 9.2, 9.1 and 9.0

On iOS 9.2, Apple fixed a total of 85 security vulnerabilities in iOS 9 and iOS 9.1. Out of these 85 vulnerabilities:

- **67** Code Execution vulnerabilities
- **10** Kernel vulnerabilities
- **1** MITM
- **6** Denial of Service
- **1** Privacy

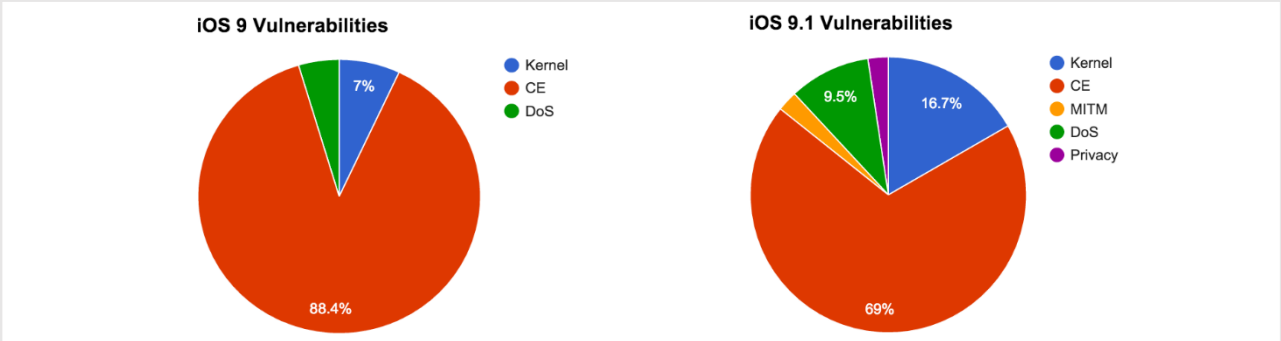


Figure 1: Vulnerabilities by Version

The graph below shows the severity levels associated with each update:

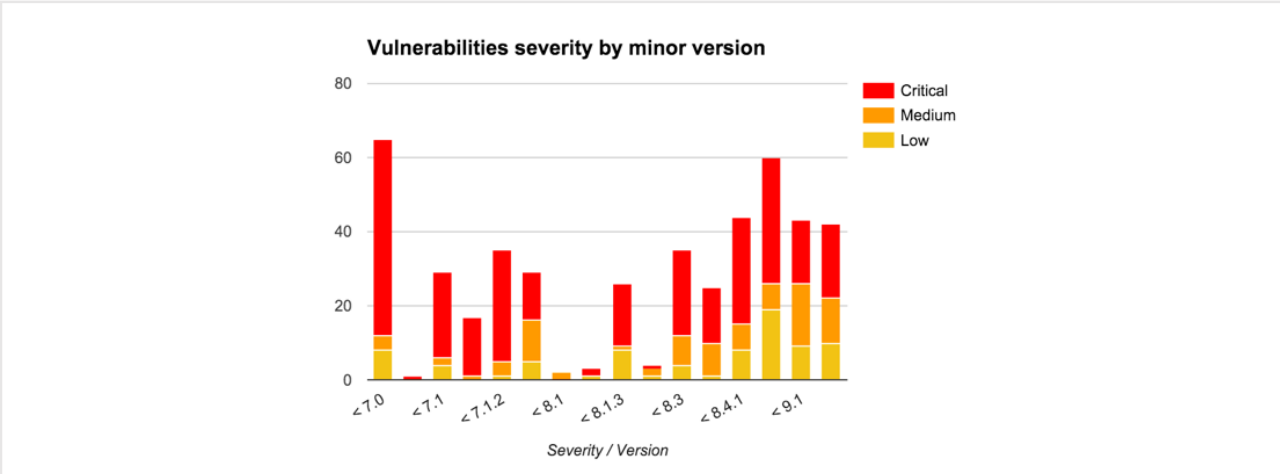


Figure 2: Severity by Version

iOS 9.2.1 Update

In the recent iOS update (9.2.1 – published on January 19th), Apple patched what we initially classified as 7 critical, 4 high, and 2 moderate severity vulnerabilities. These include at least five remotely exploitable vulnerabilities (CVE-2016-1723 through CVE-2016-1727) and at least one critical local kernel vulnerability triggerable from userland with low privileges (CVE-2016-1719). CVE-2015-7995 also appears to be exposed remotely, but determining exploitability will require further investigation. The following graph summarizes the mentioned issues.

CVE	Component	Impact	Severity
CVE-2016-1717	DiskImage	Kernel Code Execution	High
CVE-2016-1719	IOHIDFamily	Kernel Code Execution	Critical
CVE-2016-1720	IOKit	Kernel Code Execution	High

CVE-2016-1721	Kernel	Kernel Code Execution	High
CVE-2015-7995	libxslt	Remote Code Execution	Critical
CVE-2016-1722	syslogd	Code Execution w/EOP	High
CVE-2016-1723 CVE-2016-1724 CVE-2016-1725 CVE-2016-1726 CVE-2016-1727	WebKit	Remote Code Execution	Critical
CVE-2016-1728	WebKit CSS	Privacy Leak	Moderate
CVE-2016-1730	WebSheet	Privacy Leak	Moderate

iOS 9.3/9.3.1 Update

The most recent major update is iOS 9.3, published on March 21st. This update contains numerous security fixes. Apple also published iOS 9.3.1 on March 31st to fix an issue with Safari and other apps being unresponsive when clicking on specific links, but this update does not provide any additional security fixes compared to 9.3.

We classified the severity of the vulnerabilities as 19 critical, 14 high, and 2 moderate. Among the critical vulnerabilities, 18 are considered remotely exploitable, and the remaining one is a code signature bypass vulnerability.

The high severity vulnerabilities mostly include issues that might lead to code execution in the kernel. However, they are only locally exploitable and most of them also require root privileges.

Most interesting are the components libxml2 with 9 CVEs, and the WiFi driver with 2 critical CVEs (CVE-2016-0801 and CVE-2016-0802). Surprisingly, the latter are identical to those that have recently been fixed in an Android security update. These vulnerabilities are considered remotely exploitable, but due to the nature of WiFi it would require an attacker to be in close proximity and/or in the same network as the victim. While we still need further analysis of these particular vulnerabilities on iOS, it is interesting to see that Apple and Android are apparently sharing the same codebase for Broadcom-based WiFi drivers. The following table provides an overview of the CVEs that have been fixed in this update and the impact of the underlying vulnerabilities for the affected components.

CVE	Component	Impact	Severity
CVE-2016-1734	AppleUSBNetworking	Kernel Code Execution	High
CVE-2016-1740	FontParser	Remote Code Execution	Critical
CVE-2015-8659	nghttp2	Remote Code Execution	Critical
CVE-2016-1748	IOHIDFamily	Info Leak	High
CVE-2016-1752	Kernel	DoS	Moderate
CVE-2016-1750	Kernel	Kernel Code Execution	High

CVE-2016-1753	Kernel	Kernel Code Execution	High
CVE-2016-1751	Kernel	Code Signing Bypass	Critical
CVE-2016-1757	Kernel	Kernel Code Execution	High
CVE-2016-1756	Kernel	Kernel Code Execution	Moderate
CVE-2016-1754 CVE-2016-1755	Kernel	Kernel Code Execution	High
CVE-2016-1758	Kernel	Info Leak	High
CVE-2015-1819 CVE-2015-5312 CVE-2015-7499 CVE-2015-7500 CVE-2015-7942 CVE-2015-8035 CVE-2015-8242 CVE-2016-1761 CVE-2016-1762	libxml2	Remote Code Execution	Critical
CVE-2016-1788	Messages	Privacy Leak	Critical
CVE-2016-1766	Profiles	Certificate Bypass	High
CVE-2016-1950	Security	Remote Code Execution	Critical
CVE-2016-1775	TrueTypeScaler	Remote Code Execution	Critical
CVE-2016-1778 CVE-2016-1783	WebKit	Remote Code Execution	Critical
CVE-2016-1781	WebKit	Privacy Leak	High
CVE-2016-1780	WebKit	Privacy Leak	High
CVE-2016-1779	WebKit	Privacy Leak	High
CVE-2016-1786	WebKit Page Loading	UI Spoofing	High
CVE-2016-1785	WebKit Page Loading	Cross-Origin Requests	High
CVE-2016-0801 CVE-2016-0802	WiFi (Kernel driver)	Remote Code Execution (Kernel)	Critical

To summarize, whereas iOS has a better update mechanism than Android, a significant portion of iOS devices remains vulnerable for a long period of time at enterprise scale. This security gap allows attackers to launch attacks using off-the-shelf vulnerabilities.

But maybe most importantly, there is clearly not a trend of CVE decline with iOS, but the opposite. Many as-yet undisclosed vulnerabilities still exist, and every major iOS update injects countless additional opportunities for new exploits to be discovered. This is an unfortunate reality of complex software from any vendor.

Are iOS-only environments safer?

As the previous chapter reveals, each iOS update version contains security patches for critical vulnerabilities that can be exploited remotely or locally.

iOS-only environment issues in an ideal world

iOS-only environments have an advantage – their closed source (lack of fragmentation) and the continuous support by Apple reduce the risk significantly. Yet, as the updates in every major (and minor) version of iOS show, iOS environments continue to have many vulnerabilities.

Once a new update becomes available, a race takes place between professional malware groups and CIO/CISOs' teams. The attackers strive to quickly write an exploit, using publicly available hints, comparing updated binaries included in the updated release and reverse engineering various components.

The CIO/CISOs' teams, on the other hand, alert their users and ask them to update their devices. In many cases attackers are able to use the time lag between an update's public release, and the actual time when users have updated their phones, in order to gain remote access to their devices.

The latest update shows that there were 18 remotely exploitable bugs, many of which can be triggered through a browser, 9 kernel code execution vulnerabilities, and also a codesign bypass. A user running 9.2.1 or below instead of 9.3/9.3.1 would therefore expose the organization to these vulnerabilities.

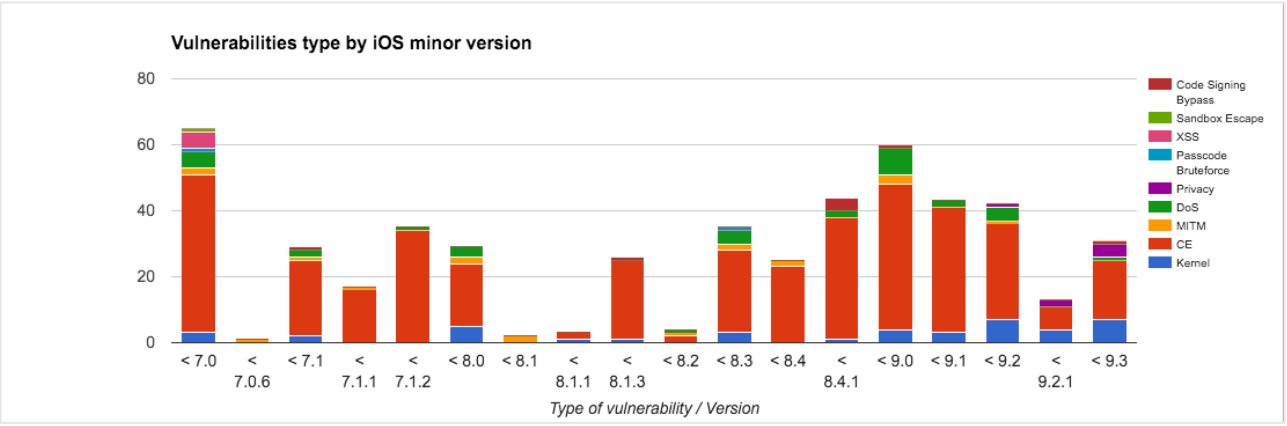


Figure 3: Type of Vulnerability / Version

Case in Point

On September 21st, Zerodium, a zero-day exploit acquisition firm, announced a bounty for an iOS 9 remote jailbreak that would be carried through a browser attack [1]. The competition offered $1,000,000 in awards to the person or group who would share a completely remote jailbreak occurrence meeting one of the following conditions:

"Eligible submissions must include a full chain of unknown, unpublished, and unreported vulnerabilities/exploits (aka zero-days) which are combined to bypass all iOS 9 exploit mitigations including: ASLR, sandboxes, rootless, code signing, and bootchain.

The exploit/jailbreak must lead to and allow a remote, privileged, and persistent installation of an arbitrary app" ... "on a fully updated iOS 9 device..."

The initial attack vector must be either:

- *a web page targeting the mobile browser (Mobile Safari OR Google Chrome) in its default configuration; OR*
- *a web page targeting any application reachable through the browser; OR*
- *a text message and/or a multimedia file delivered through a SMS or MMS.*

The whole exploitation/jailbreak process should be achievable remotely, reliably, silently, and without requiring any user interaction except visiting a web page or reading a SMS/MMS (attack vectors such as physical access, bluetooth, NFC, or baseband are not eligible for the Million Dollar iOS 9 Bug Bounty. ZERODIUM may, at its sole discretion, make a distinct offer to acquire such attack vectors.).

The exploit/jailbreak must support and work reliably on the following devices (32-bit and 64-bit when applicable):

- *iPhone 6s / iPhone 6s Plus / iPhone 6 / iPhone 6 Plus*
- *iPhone 5 / iPhone 5c / iPhone 5s*
- *iPad Air 2 / iPad Air / iPad (4rd generation) / iPad (3th generation) / iPad mini 4 / iPad mini 2*

Partial or incomplete exploits/jailbreaks will not be eligible for the Million Dollar iOS 9 Bug Bounty. ZERODIUM may, at its sole discretion, make a distinct offer to acquire such partial exploits."

On November 1st, Zerodium announced that their iOS 9 Bug Bounty had expired and only one team had won the competition gaining $1,000,000 as an award. Zerodium is a private company that sells zero-day's to governments without validating the usage and purpose of these exploits.

We know that zero-day's are being used in the wild but this is the first real world confirmation that "we know that we don't know", even on iOS 9 (before it came out). We have no confirmation if this bug was patched in iOS 9.1, 9.2 or 9.2.1 and we can and should assume that this bug is being exploited in the wild against fully patched (iOS 9.2.1), or partially patched (iOS 9.0) devices.

The "Walled Garden" in iOS Attacks

The "walled garden" used to be a differentiator for Apple. Multiple vulnerabilities and malicious apps revealed that the advantage over Android's open-garden is that it helps reduce risk but doesn't mitigate it completely.

In the last few years we have observed:

iOS XCodeGhost has infected dozens of apps that made their way into the official App Store, including WeChat, that was downloaded by tens of millions of users.

Every user with an infected version of WeChat or any other XCodeGhost infected app was at the mercy of attackers.

Vulnerabilities since the launch of major and minor OS versions

The table below describes the timeline for a first publicly available jailbreak for every major iOS version. Each of the jailbreaks contains at least two distinct vulnerabilities, including a kernel exploit and initial code execution vulnerability, code signing vulnerability or sandbox escape. In many cases, the jailbreakers had released a complete jailbreak within days of the iOS release. Also, the difference between jailbreak and an attack is often the method of delivery of the exploit.

Attackers, as in the case of Zerodium, would prefer to deliver the exploit remotely (e.g., via browser exploit) and continue the attack-chain in a similar way to vulnerabilities used in the original jailbreaks.

The recent iOS 9.2.1 fixed five critical, remote vulnerabilities in WebKit. Chaining a WebKit vulnerability with the vulnerabilities used by a complete jailbreak would result in a complete remote device-takeover.

Table of jailbreaks by device and iOS version, 2007-present:

Device OS Both

Device/OS	Release date	Tool	Developer(s)	Date of first jailbreak	Time(days) until first jailbreak
iPhone / iPhone OS 1.0	June 29, 2007	(no name)	iPhone Dev Team[116]	July 10, 2007[75]	11
iPod touch	September 5, 2007	(no name)	niacin and dre	October 10, 2007[117][118]	35
iPhone 3G /iPhone OS 2.0	July 11, 2008	PwnageTool	iPhone Dev Team	July 20, 2008[81]	9
iPod touch (2nd generation)	September 9, 2008	redsn0w	iPhone Dev Team and Chronic Dev Team	January 30, 2009[119][120]	143
iPhone OS 3.0	June 17, 2009	PwnageTool	iPhone Dev Team	June 19, 2009[121]	2
iPhone 3GS	June 19, 2009	purplera1n	George Hotz	July 3, 2009[122]	14
iPad	April 30, 2010	Spirit	comex	May 3, 2010[91]	3
iOS 4.0	June 21, 2010	PwnageTool	iPhone Dev Team	June 23, 2010[123][124]	2
iPhone 4	June 24, 2010	JailbreakMe2.0	comex	August 1, 2010[92]	38
Apple TV (2nd generation)	September 1, 2010	PwnageTool	iPhone Dev Team	October 20, 2010[125]	49
iPad 2	March 11, 2011	JailbreakMe 3.0	comex	July 5, 2011[95]	116
iOS 5.0	October 12, 2011	redsn0w	iPhone Dev Team	October 13, 2011[87]	1

	iPhone 4S	October 14, 2011	Absinthe	pod2g, Chronic Dev Team, iPhone Dev Team	January 20, 2012[24][25]	98
	Apple TV (3rd generation)	March 7, 2012	-	-	-	-
	iPad (3rd generation)	March 16, 2012	Absinthe 2.0	pod2g, Chronic Dev Team, iPhone Dev Team	May 25, 2012	70
	iOS 6.0	September 19, 2012	redsn0w	iPhone Dev Team	September 19, 2012	0
	iPhone 5	September 21, 2012	evasion	evad3rs	February 4, 2013	136
	iPod touch (5th generation)	October 23, 2012	evasi0n	evad3rs	February 4, 2013	104
	iPad (4th generation)	November 2, 2012	evasi0n	evad3rs	February 4, 2013	94
	iPad Mini	November 2, 2012	evasi0n	evad3rs	February 4, 2013	94
	iOS 7	September 18, 2013	evasi0n7	evad3rs	December 22, 2013	95
	iPhone 5C	September 20, 2013	evasi0n7	evad3rs	December 22, 2013	93
	iPhone 5S	September 20, 2013	evasi0n7	evad3rs	December 22, 2013	93
	iPad Air	November 1, 2013	evasi0n7	evad3rs	December 22, 2013	51
	iPad Mini 2	November 12, 2013	evasi0n7	evad3rs	December 22, 2013	40
	iOS 7.1—7.1.2	May 29, 2014	Pangu	Pangu Team	June 23, 2014	25
	iOS 8	September 17, 2014	Pangu8	Pangu Team	October 22, 2014	35
	iPhone 6	September 19, 2014	Pangu8	Pangu Team	October 22, 2014	33
	iPhone 6 Plus	September 19, 2014	Pangu8	Pangu Team	October 22, 2014	33
	iPad Air 2	October 22, 2014	Pangu8	Pangu Team	October 22, 2014	0
	iPad Mini 3	October 22, 2014	Pangu8	Pangu Team	October 22, 2014	0
	iOS 8.1.1—8.4	November 17, 2014	TaiG, PP Jailbreak	TaiG, PP Jailbreak	November 29, 2014	12
	Apple Watch	April 24, 2015	-	-	-	-
	iPod touch (6th generation)	July 15, 2015	TaiG, PP Jailbreak	TaiG, PP Jailbreak	July 16, 2015	1

	iOS 9	September 16, 2015	Pangu9	Pangu Team	October 14, 2015	28
	iPhone 6S	September 25, 2015	Pangu9	Pangu Team	October 14, 2015	19
	iPhone 6S Plus	September 25, 2015	Pangu9	Pangu Team	October 14, 2015	19
	iPad Mini 4	September 9, 2015	Pangu9	Pangu Team	October 14, 2015	35
	iOS 9.1	October 21, 2015	Pangu9.1	Pangu Team	March 11, 2016	142
	Apple TV (4th generation)	October 30, 2015	Pangu9.1	Pangu Team	March 23, 2016	145
	iPad Pro	November 11, 2015	-	-	-	-

Source: https://en.wikipedia.org/wiki/IOS_jailbreaking

A Working Solution

Zimperium Mobile Threat Protection delivers continuous and real-time threat protection to both devices and applications to stop mobile attacks.

Over the past four to five years, Zimperium has detected all jailbreak attempts without modifying our core z9 engine code. We added security mechanisms to detect advanced host attacks on all iOS versions including the latest iOS version 9.3.1. So far, we detected all the publicly available exploits including the following: JailbreakMe 2.0, JailbreakMe 3.0, PwnageTool, Redsn0w, Absinthe, Absinthe 2.0, evasi0n, evasi0n7, Pangu, Pangu 8, Taig, PP Jailbreak, and Pangu 9/9.1.

CONTACT US

101 Mission Street
San Francisco, CA 94105
Main: 415.992.8922 | Toll Free: 844.601.6760
sales@zimperium.com
www.zimperium.com