

BGA

**BİLGİ GÜVENLİĞİ
AKADEMİSİ**

www.bga.com.tr

Mobil Uygulama Güvenlik Testlerinde Sertifika Sabitlemeyi Devre Dışı Bırakma

[Certificate Pinning Bypass]

Candan BOLUKBAS <candan.bolukbas@bga.com.tr>

[Pentestlerinde çokça kullanılan SSL sonlandırma kapasitesine sahip proxy uygulamaları SSL trafiğini sonlandırırsa dahi, kullanıcı tarafındaki uygulama kendi içerisindeki Pinned Certificate haricinde bir sertifika ile karşılaşınca iletişimi sonlandırır. Bu durum testleri oldukça zorlaştırır, çözüm ise Man-in-The-Middle (MiTM) yerine Man-in-The-Front (MiTF) yöntemidir.]

İçerik Tablosu

Android	2
Android SSL Trust Bypass	7
Android-SSL-TrustKiller	9
iOS.....	11
iOS SSL Kill Switch	14
Referanslar.....	18

Certificate Pinning Bypass

(Sertifika Sabitlemeyi Devre Dışı Bırakma)

Certificate Pinning yöntemi bir uygulamada bir sertifika otoritesine (CA) güvenmeyi veya bir DNS adına bağımlı olmayı çözen, aynı zamanda PKI ortamında anahtar değişimi ihtiyacının ortadan kaldıran oldukça güvenli bir yöntemdir. Temel olarak çalışma mantığında kullanıcı tarafı ile sunucu arasında önceden belirlenmiş bir Public-Private anahtar ikilisinin Public anahtarının kullanıcı tarafı uygulamasının içine hard-coded veya dosya tabanlı olarak gömülmesi vardır.

Certificate Pinning'e alternatif yöntemlerden bazıları: Ephemeral Keys, Pinning Gaps, Revocation ve X509 Validation.

Zaafiyet testlerinde çokça kullanılan SSL sonlandırma kapasitesine sahip proxy uygulamaları (Fiddler, BurpSuite, Zap vs.) SSL trafiğini sonlandırırsa dahi, kullanıcı tarafındaki uygulama kendi içerisindeki (Pinned Certificate) haricinde bir sertifika ile karşılaşınca iletişimi sonlandırır. Bu durum testleri oldukça zorlaştırır. Bu duruma çözüm ise Man-in-The-Middle (MiTM) yerine Man-in-The-Front (MiTF) yöntemidir.

Man-in-The-Front (önde duran adam) yöntemi iletişimin başlangıcında durmaktır. Bu yöntem kağıt kıyma makinasından ilham alınarak ortaya çıkarılmıştır. Bazı kağıt kıyma makinaları kağıtları kıymadan önce kıyma bıçaklarından önünde bulunan tarayıcı ile kırılacak kağıdı tarayıp, bir sunucuya yüklüyor. Aynı şekilde MiTF yöntemiyle kullanıcı tarafındaki uygulama gelen sertifikanın doğruluğunu kontrol ederken işletim sistemi fonksiyonlarını çağırırken, biz tam bu noktada yapılan bu kontrolü manipüle edip gelen tüm sertifikalara güvenmesini sağlıyoruz.

Android

Android işletim sistemlerinde Certificate Pinning uygulaması için aşağıdaki kütüphaneler kullanılır

- `javax.net.ssl`
- `org.apache.http`

`javax.net.ssl` kütüphanesi uygulama spesifik `X509TrustManager` oluşturulmasını sağlar. Eğer bir uygulama işletim sisteminin sertifika otoritelerine (CA) güvenmek istemezse, bu kütüphane ile `HttpsURLConnection` oluştururken kendi `SSLConnectionFactory`'sini oluşturur ve böylece kendi `X509TrustManager`'ını uygulamış olur. Aşağıda örnek bir kod bloğu gösterilmiştir:

```
// Load CAs from an InputStream
// (could be from a resource or ByteArrayInputStream or ...)
```

```
CertificateFactory cf = CertificateFactory.getInstance("X.509");  
  
// From the file that already in local file system namely load-der.crt  
InputStream caInput = new BufferedInputStream(new FileInputStream("load-der.crt")); // [1]  
Certificate ca;  
  
try {  
    ca = cf.generateCertificate(caInput);  
    System.out.println("ca=" + ((X509Certificate) ca).getSubjectDN());  
} finally {  
    caInput.close();  
}  
  
// Create a KeyStore containing our trusted CAs  
String keyStoreType = KeyStore.getDefaultType();  
KeyStore keyStore = KeyStore.getInstance(keyStoreType);  
keyStore.load(null, null);  
keyStore.setCertificateEntry("ca", ca); // [2]  
  
// Create a TrustManager that trusts the CAs in our KeyStore  
String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();  
TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);  
tmf.init(keyStore);  
  
// Create an SSLContext that uses our TrustManager  
SSLContext context = SSLContext.getInstance("TLS");  
context.init(null, tmf.getTrustManagers(), null); // [3]  
  
// Tell the URLConnection to use a SocketFactory from our SSLContext  
URL url = new URL("https://certs.cac.washington.edu/CAtest/");  
HttpsURLConnection urlConnection = (HttpsURLConnection)url.openConnection(); // [4]  
urlConnection.setSSLSocketFactory(context.getSocketFactory()); // [5]  
InputStream in = urlConnection.getInputStream();  
copyInputStreamToOutputStream(in, System.out);
```

Bu kod bloğunda kısaca:

- [1] yazan kısımda uygulama içerisine gömülü olan sertifika CA olarak yükleniyor.
- [2] yazana kısımda bu CA kullanılarak KeyStore objesi tanımlanıyor
- [3] yazan kısımda SSLContext'i X509TrustManager objesi kullanılarak tanımlanıyor
- [4] ve [5] numaralı kısımlarda oluşturulan SSLContext'i kullanılarak soket açılıyor.

<https://developer.android.com/reference/java/net/URLConnection.html>

Bu şekilde bir uygulama oldukça güvenli duruyor ancak eğer biz [5]'teki SSLContext'in getSocketFactory() fonksiyonundan dönecek SSLSocketFactory objesinin aşağıdaki şekilde güvenlik kontrollerini yeniden yazarsak, soket güvenli olmayan bir sertifika ile de açılabilir.

<https://android.googlesource.com/platform/external/apache-http/+2f0e9505248dc19d7fcfd4c1efd7c3daf863f03/src/org/apache/http/conn/ssl/SSLSocketFactory.java>

```
public class SSLSocketFactory implements LayeredSocketFactory {
    ...
    private final SSLContext sslcontext;
    private final javax.net.ssl.SSLSocketFactory socketfactory;
    private final HostNameResolver nameResolver;
    private X509HostnameVerifier hostnameVerifier = BROWSER_COMPATIBLE_HOSTNAME_VERIFIER;
    // burayı ALLOW_ALL_HOSTNAME_VERIFIER olarak modifiye ediyoruz
    ...
    public Socket createSocket(final Socket socket, final String host, final int port, final
    boolean autoClose) throws IOException, UnknownHostException {
        SSLSocket sslSocket = (SSLSocket)
        this.socketfactory.createSocket(socket, host, port, autoClose);

        hostnameVerifier.verify(host, sslSocket); // bu noktayı modifiye edip tüm
        sertifikalar için verify() fonksiyonunun true dönmelerini sağlıyoruz.

        return sslSocket;
    }
    public void setHostnameVerifier(X509HostnameVerifier hostnameVerifier) {
        if ( hostnameVerifier == null ) {
            throw new IllegalArgumentException("Hostname verifier may not be null");
        }
        this.hostnameVerifier = hostnameVerifier;
    }
    public X509HostnameVerifier getHostnameVerifier() {
        return hostnameVerifier;
    }
    ...
    public interface X509HostnameVerifier extends HostNameVerifier {
        boolean verify(String host, SSLSession session);
        void verify(String host, SSLSocket ssl) throws IOException;
    }
}
```

```
void verify(String host, X509Certificate cert) throws SSLException;

/**
 * Checks to see if the supplied hostname matches any of the supplied CNs
 * or "DNS" Subject-Alts. Most implementations only look at the first CN,
 * and ignore any additional CNs. Most implementations do look at all of
 * the "DNS" Subject-Alts. The CNs or Subject-Alts may contain wildcards
 * according to RFC 2818.
 *
 * @param cns CN fields, in order, as extracted from the X.509
 * certificate.
 * @param subjectAlts Subject-Alt fields of type 2 ("DNS"), as extracted
 * from the X.509 certificate.
 * @param host The hostname to verify.
 * @throws SSLException If verification failed.
 */
void verify(String host, String[] cns, String[] subjectAlts)
    throws SSLException;
}
```

Sertifika kontrolü yapan bir diğer kütüphane de `org.apache.http` kütüphanesidir. Bu kütüphane de `javax.net.ssl` kütüphanesi gibi uygulama içerisine gömülen sertifikadan bir `KeyStore` objesi, bu objeden bir `SSLContext` ve bu context'ten bir `SSLSocketFactory` objesi oluşturur. Aynı şekilde `SSLSocketFactory` sınıfında aşağıda ki kontroller yapılır;

<https://android.googlesource.com/platform/external/apache-http/+2f0e9505248dc19d7fcfed4c1efd7c3daf863f03/src/org/apache/http/conn/ssl/SSLSocketFactory.java>

```
/* Checks whether a socket connection is secure.
 * This factory creates TLS/SSL socket connections
 * which, by default, are considered secure.
 * Derived classes may override this method to perform
 * runtime checks, for example based on the cypher suite.
 * @param sock the connected socket
 * @return true
 * @throws IllegalArgumentException if the argument is invalid
 */
public boolean isSecure(Socket sock)
```

```

        throws IllegalArgumentException {
        if (sock == null) {
            throw new IllegalArgumentException("Socket may not be null.");
        }
        // This instanceof check is in line with createSocket() above.
        if (!(sock instanceof SSLSocket)) {
            throw new IllegalArgumentException
                ("Socket not created by this factory.");
        }
        // This check is performed last since it calls the argument object.
        if (sock.isClosed()) {
            throw new IllegalArgumentException("Socket is closed.");
        }
        return true;
    } // isSecure

    // non-javadoc, see interface LayeredSocketFactory
    public Socket createSocket(
        final Socket socket,
        final String host,
        final int port,
        final boolean autoClose
    ) throws IOException, UnknownHostException {
        SSLSocket sslSocket = (SSLSocket) this.socketfactory.createSocket(
            socket,
            host,
            port,
            autoClose
        );
        hostnameVerifier.verify(host, sslSocket); // bu noktayı modifiye edip tüm
sertifikalar için verify() fonksiyonunun true dönmelerini sağlıyoruz.
        return sslSocket;
    }

    public void setHostnameVerifier(X509HostnameVerifier hostnameVerifier) {
        if ( hostnameVerifier == null ) {

```

```
        throw new IllegalArgumentException("Hostname verifier may not be null");
    }

    this.hostnameVerifier = hostnameVerifier;
}

public X509HostnameVerifier getHostnameVerifier() {
    return hostnameVerifier; // burada hostnameVerifier'in değerini
ALLOW_ALL_HOSTNAME_VERIFIER olarak modifiye ediyoruz
}
```

Bu şekilde çalışacak bir düzenleme Certificate Pinning'i bypass edecektir. Bunu gerçekleştirmenin birkaç yolu var;

1. Custom VM/ROM: İlgili modifikasyonları yaptıktan sonra kendi Android işletim sistemimizi derleyebiliriz. Bu metod en iyi yol ancak Android'i derlemek pek kolay değil. Birkaç yüz GB disk alanınızı ve birkaç haftanızı harcamanız gerekebilir.
2. Test yapmak istediğimiz uygulamayı Decompile edip, ilgili Patch'lemeyi yapıp, tekrar derleyip, imzalayıp, yeniden yükleyebiliriz. Bu yöntem de oldukça zor. Her uygulamayı başarıyla Decmpile/Recompile etmek pek kolay değil.
3. JDWP debugger: Java Debug Wire Protocol (JDWP)'u kullanarak manipülasyon yapmak istediğimiz yere breakpoint atarak, ilgili yerde manipülasyon yapmak mümkün. Ancak pentester'lar için bu şekilde manuel bir metod pek uygun sayılmaz.
4. Native code hooking (Mulliner) veya native code debugger (gdb, vtrace): Doğal kod kancalama yönteminde işletim sisteminin manipülasyon yapmak istediğimiz yere hook (kanca) atarak, tüm uygulamalardan gelen istekleri manipüle edebiliriz. ISEC Research Labs tarafından yazılan Android-SSL-TrustKiller uygulaması tam olarak bu işi yapıyor.

Android SSL Trust Bypass

Android-SSL-TrustKiller, Cydia Substrate üzerinde çalışan bir eklenti. Cydia Substrate iOS ve Android işletim sistemlerinde işletim sisteminin yeniden derlenmesine gerek kalmadan veya bir uygulamanın kaynak kodlarında modifikasyon yapmadan yazılan bir eklentinin hafızada bir uygulamaya bağlanmasını veya belirtilen izinlerle bir uygulamanın “class load” çağrılarını enjekte olabiliyor. Kısaca Cydia Substrate için API'si olan, oldukça güçlü bir çeşit mobil rootkit diyebiliriz. Detaylar: <http://www.cydiasubstrate.com> Android-SSL-TrustKiller, Cydia Substrate üzerinde çalışan bir eklenti ve yukarıda bahsettiğimiz şekilde aşağıdaki modifikasyonları yapıyor:

```
// This hook overrides the getTrustManagers method
// from javax.net.ssl.TrustManagerFactory
// and returns a Trust Manager that doesn't perform checks
MS.hookClassLoad("javax.net.ssl.TrustManagerFactory",
    new MS.ClassLoadHook() {
    ...
}
```



```
// This hook overrides the setSSLSocketFactory method
// from javax.net.ssl.HttpURLConnection
// and sets a SSL Socket Factory that uses a Trust Manager
// that doesn't perform checks
MS.hookClassLoad("javax.net.ssl.HttpURLConnection",
    new MS.ClassLoadHook() {
    ...
// This hook overrides the init method
// from javax.net.ssl.SSLContext
// and init the SSLContext with a Trust Manager that doesn't perform checks
MS.hookClassLoad("javax.net.ssl.SSLContext",
    new MS.ClassLoadHook() {
    ...
// This hooks setHostnameVerifier
// from org.apache.http.conn.ssl.SSLSocketFactory
// and overrides the second parameter with
// "SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER"
// to accept any hostname
MS.hookClassLoad("org.apache.http.conn.ssl.SSLSocketFactory",
    new MS.ClassLoadHook() {
    ...
// This hook forces the isSecure method
// from org.apache.http.conn.ssl.SSLSocketFactory
// to always return true
MS.hookClassLoad("org.apache.http.conn.ssl.SSLSocketFactory",
    new MS.ClassLoadHook() {
    ...
// Overrides the setDefaultHostnameVerifier method
// from org.apache.http.conn.ssl.SSLSocketFactory
// to accept any hostname
MS.hookClassLoad("javax.net.ssl.HttpURLConnection",
    new MS.ClassLoadHook() {
    ...
```

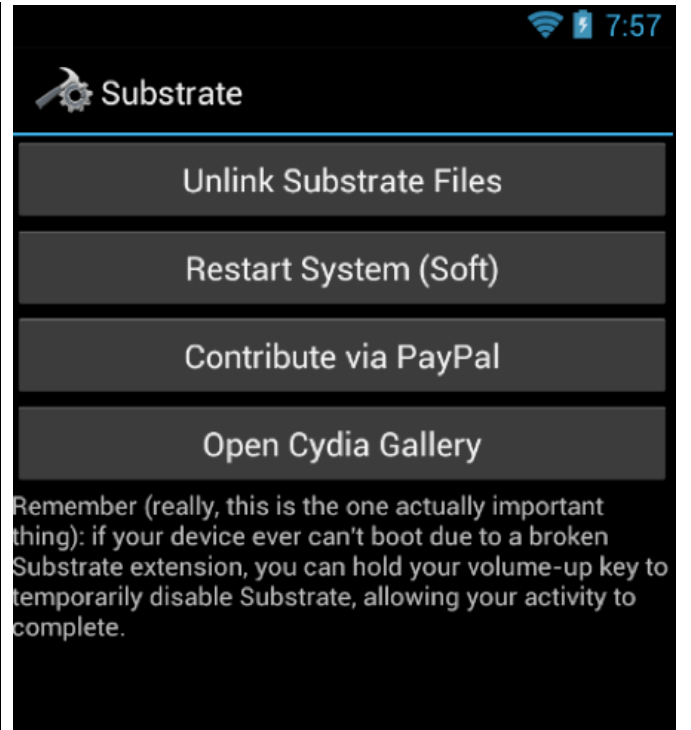
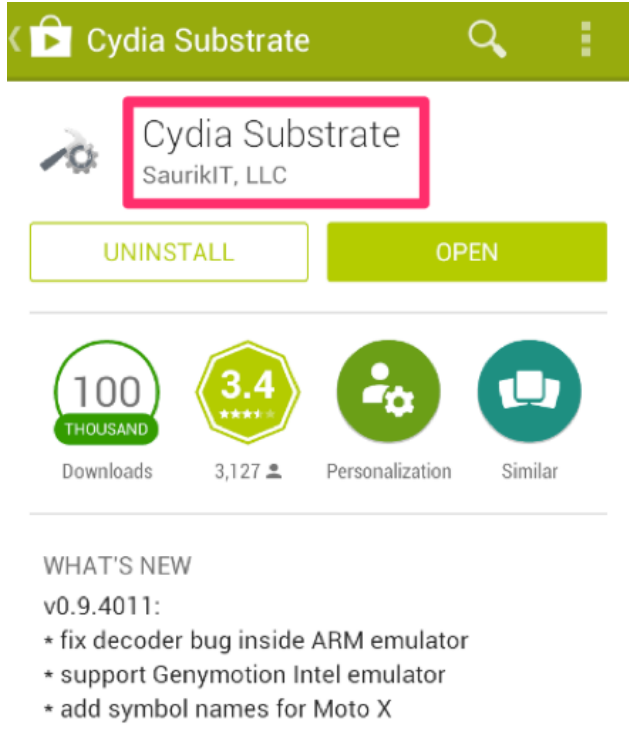
<https://github.com/iSECPartners/Android-SSL-TrustKiller>

Android-SSL-TrustKiller

Bu eklenti Cydia Substrate yardımı ile çeşitli sistem fonksiyonlarına kanca atarak, Certificate Pinning'i devre dışı bırakıp, tüm sertifikalar ile iletişim kurulmasını sağlar.


Android-SSL-TrustKiller için:

1. Test yapılacak cihaz üzerinde root seviyesinde erişim izninin olması gerekiyor.
2. Cydia Substrate uygulamasının kurulması gerekiyor. Uygulama Google Play Store'dan ücretsiz indirilebiliyor.
<https://play.google.com/store/apps/details?id=com.saurik.substrate&hl=en> adresinden Cydia Substrate uygulamasını indirip kuruyoruz.



3. Kendi Android-SSL-TrustKiller eklentimizi <https://github.com/iSECPartners/Android-SSL-TrustKiller> adresinde indirip derleyebileceğimiz gibi, hazır derlenmiş APK dosyasını <https://github.com/iSECPartners/Android-SSL-TrustKiller/releases> adresinden indirebiliriz.

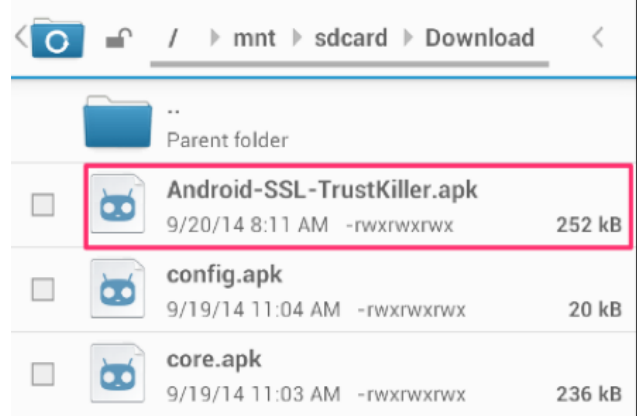
Android SSL TrustKiller v1

 nabla-c0d3 released this on Dec 14, 2013 · 2 commits to master since this release

Initial release.

↓ Android-SSL-TrustKiller.apk

Source code (zip)



4. Son olarak bu APK'yı kurduktan sonra işletim sistemini yeniden başlatınca artık cihazımız uygulamalar içerisinde Certificate Pinning olsa dahi her türlü sertifika ile SSL bağlantısı kuracaktır. Artık cihazın proxy ayarlarını ZAP, Burp veya Fiddler üzerinden geçirip sertifika kontrolünü devre dışı bıraktığını test edebiliriz. Tabi kullandığınız proxy uygulamasının CA sertifikasını cihazımızın "TrustStore"una eklenmesi gerekiyor. Twitter uygulaması Certificate Pinning kontrolü yapan bir uygulamadır ve Samsung Galaxy S3 - Android 4.1.1 işletim sistemi ve ZAP ile yaptığımız test sonucunda aşağıdaki şekilde sertifika kontrolü devre dışı bırakılarak tüm trafik izlenebilmiştir.

Örnek istek (Request): Gönderilen isteğin içeri sertifika pinlemeyi devre dışı bırakarak görebiliyoruz.

```
GET
https://api.twitter.com/1.1/trends/timeline.json?woeid=1&timezone=GMT&lang=en&pc=true&
include_media_features=true&include_cards=true&cards_platform=Android-
10&include_descendent_reply_count=true HTTP/1.1

X-Twitter-Client-Limit-Ad-Tracking: 0

X-Twitter-API-Version: 5

Accept-Language: en-US

Authorization: OAuth realm="http://api.twitter.com/", oauth_version="1.0",
oauth_token="2413629533-td18cunHkOBTzqVHhFrxxxxxxxxxxxxxxxxxxxx",
oauth_nonce="8509887043348768201576888275811", oauth_timestamp="1411132012",
oauth_signature="%2FcRiqq%2BaV2uHPBCsS9y1k0sQbuc%3D",
oauth_consumer_key="3nVuSoBZnx6U4vzUxxxx", oauth_signature_method="HMAC-SHA1"

X-Twitter-Client: TwitterAndroid

X-Twitter-Client-AdID: b9269adc-9028-46ac-9223-d44842fxxxxx

User-Agent: TwitterAndroid/5.26.0 (3030737-r-677) Samsung Galaxy S3 - 4.1.1 - API 16 -
720x1280/4.1.1 (Genymotion;Samsung Galaxy S3 - 4.1.1 - API 16 -
720x1280;generic;vbox86p;0;;1)

X-Twitter-Client-DeviceID: 34fc178c0339d9ff

X-Twitter-Client-Version: 5.26.0

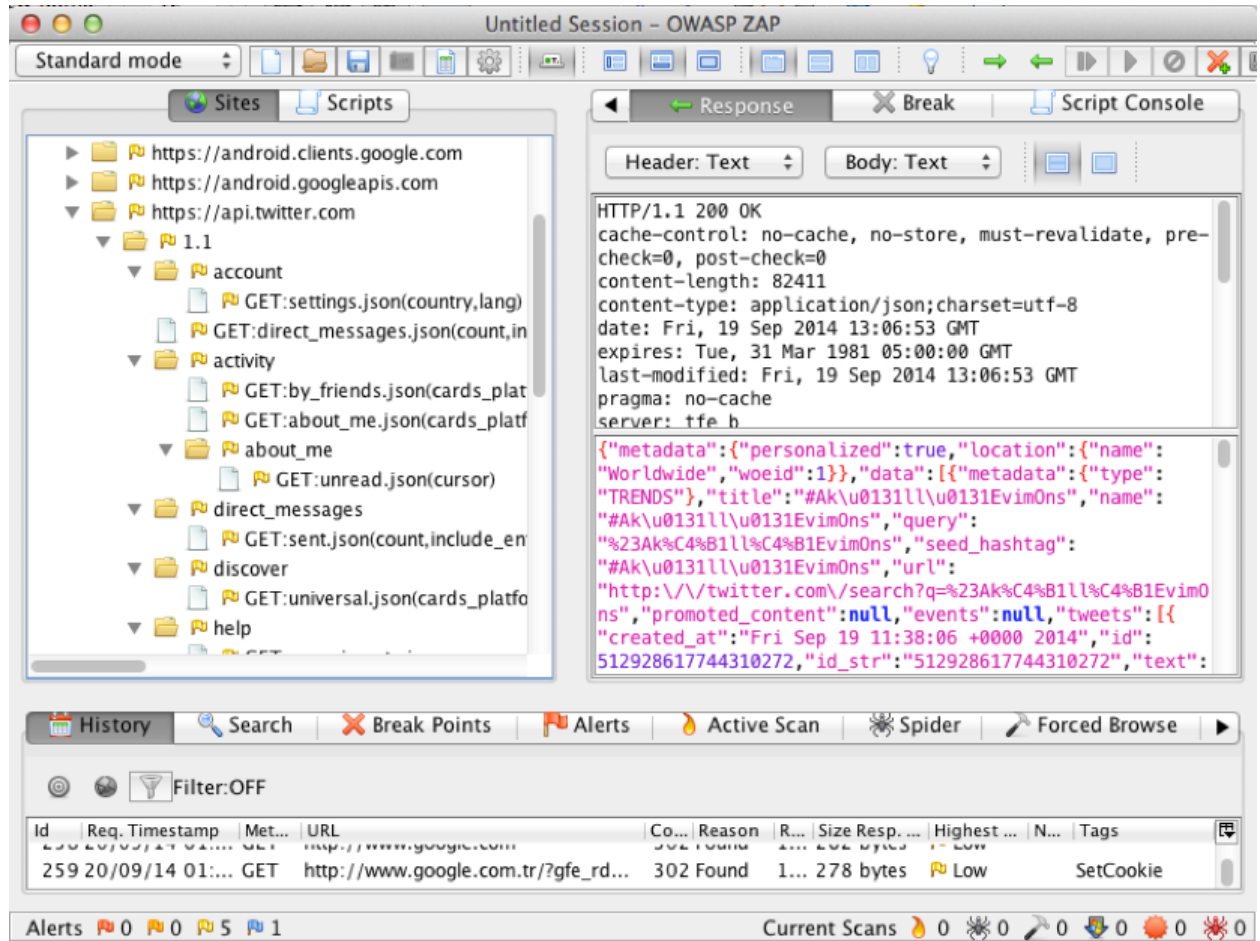
X-Client-UUID: e5a826d4-6028-4c6a-bc96-ec6f52bc3db1

Connection: Keep-Alive
```

Host: api.twitter.com

Gelen Cevap (Response): Cevabın içeriğini sertifika pinlemeyi devre dışı bırakarak görebiliyoruz.

```
{
  "metadata": {
    "personalized": true,
    "location": {
      "name": "Worldwide",
      "woeid": 1
    }
  },
  "data": {
    "metadata": {
      "type": "TRENDS",
      "title": "#Ak\u0131ll\u0131EvimO\u0131ns",
      "name": "#Ak\u0131ll\u0131EvimO\u0131ns",
      "query": "%23Ak%C4%B1ll%C4%B1EvimO\u0131ns",
      "seed_hashtag": "#Ak\u0131ll\u0131EvimO\u0131ns",
      "url": "http://twitter.com/search?q=%23Ak%C4%B1ll%C4%B1EvimO\u0131ns",
      "promoted_content": null,
      "events": null,
      "tweets": [
        {
          "created_at": "Fri Sep 19 11:38:06 +0000 2014",
          "id": 512928617744310272,
          "id_str": "512928617744310272",
          "text": "William Butler\n#Ak\u0131ll\u0131EvimO\u0131ns",
          "source": "\u003ca href=\"http://www.twitter.com\" rel=\"nofollow\" \u003eTwitter for Windows\u003c/a\u003e",
          "truncated": false,
          "in_reply_to_status_id": null,
          "in_reply_to_status_id_str": null,
          "in_reply_to_user_id": null,
          "in_reply_to_user_id_str": null,
          "in_reply_to_screen_name": null,
          "user": {
            "id": 515732038,
            "id_str": "515732038",
            "name": "@FBBar\u0131\u015f",
            "screen_name": "bar_n1",
            "location": "",
            "url": null,
            "description": "\u015eampiyon FeNeRbAh\u00c7e",
            "protected": false,
            "followers_count": 1071,
            "friends_count": 2009...
          }
        }
      ]
    }
  }
}
```



iOS

iOS işletim sistemlerinde network istekleri için NSStream, CFStream, NSURLConnection API'leri kullanılır. iOS uygulama geliştiricilerinin çoğu URL isteklerini yapmak, sunucu sertifikalarını doğrulamak ve Certificate Pinning yapabilmek için NSURLConnection'ı

override ederek kullanırlar. `NSURLConnection` class referansı aşağıdaki şekilde tanımlanmıştır.

```
@interface NSURLConnection : NSObject
{
    @private
    NSURLConnectionInternal *_internal;
}

/* Designated initializer */
- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate
startImmediately:(BOOL)startImmediately NS_AVAILABLE(10_5, 2_0);

- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate;
```

Certificate Pinning yapmak için genellikle `NSURLConnection` sınıfının `initWithRequest` constructor'ı kullanılır. Yapılan request sonrasında gelen cevap parametre olarak verilen delegate objesine gönderilir. Bu delegate sınıfı `NSURLConnectionDelegate` tipinde ve aşağıdaki şekilde tanımlanmıştır;

```
@interface NSURLConnection : NSObject
{
    @private
    NSURLConnectionInternal *_internal;
}

/* Designated initializer */
- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate
startImmediately:(BOOL)startImmediately NS_AVAILABLE(10_5, 2_0);

- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate;
```

Certificate Pinning işleminde `NSURLConnectionDelegate` sınıfının `willSendRequestForAuthenticationChallenge` metodu veya `canAuthenticateAgainstProtectionSpace`, `didReceiveAuthenticationChallenge` ve `didReceiveAuthenticationChallenge` metodlarından oluşan bir sertifika doğrulama sistemi kullanılır.

Bu fonksiyonlar SSL bağlantısı kurarken `SSLContext` oluşturulur ve sertifika kontrolü bu kısımda olur. Bağlantı esnasında kullanılacak sertifika `SSLSetCertificate` metodu ile kontrol edilir.

```
OSStatus SSLSetCertificate ( SSLContextRef context, CFArrayRef certRefs );
```

Certificate Pinning'i bypass etmek için `SSLContext` oluşturulurken `SSLContextRef` ve `SSLSetCertificate` metodunu manipule edip her türlü sertifika ile bağlantı kurmasını ve sertifika hatası olduğu durumda ortaya çıkan `kSSLSessionOptionBreakOnServerAuth` opsiyonunun hata oluşturmamasını sağlamak yeterli olacaktır.

<https://developer.apple.com/library/prerelease/mac/documentation/Security/Reference/secureTransportRef>

Android işletim sisteminde daha önce bahsettiğimiz **Cydia Substrate**'in iOS üzerinde çalışabilen versiyonu bulunmaktadır ve aynı şekilde işletim sistemi seviyesinde yapılan çağrılar ve

runtime'da class yüklemelerini modifiye edebilmektedir. Detyalar: <http://www.cydiasubstrate.com>

Cydia Substrate üzerinde çalışan ve iSEC Partners tarafından geliştirilen iOS SSL Kill Switch eklentisi bahsettiğimiz bu modifikasyonları yapmamızı sağlar. Kaynak kodlarını incelediğimizde eklentinin SSLContext oluşturulurken SSLContextRef ve SSLSetCertificate metodunu manipule edip her türlü sertifika ile bağlantı kurmasını ve sertifika hatası olduğu durumda ortaya çıkan kSSLSessionOptionBreakOnServerAuth opsiyonunun hata oluşturmamasını sağladığı görülmektedir.

```
static OSStatus replaced_SSLHandshake(
    SSLContextRef context
) {
    OSStatus result = original_SSLHandshake(context);

    // Hijack the flow when breaking on server authentication
    if (result == errSSLServerAuthCompleted) {
        // Do not check the cert and call SSLHandshake() again
        return original_SSLHandshake(context);
    }
    else
        return result;
}
...
static SSLContextRef replaced_SSLCreateContext (
    CFAllocatorRef alloc,
    SSLProtocolSide protocolSide,
    SSLConnectionType connectionType
) {
    SSLContextRef sslContext = original_SSLCreateContext(alloc, protocolSide,
    connectionType);

    // Immediately set the kSSLSessionOptionBreakOnServerAuth option in order to
    // disable cert validation
    original_SSLSetSessionOption(sslContext, kSSLSessionOptionBreakOnServerAuth,
    true);
    return sslContext;
}
...
static OSStatus replaced_SSLSetSessionOption(
    SSLContextRef context,
    SSLSessionOption option,
    Boolean value) {

    // Remove the ability to modify the value of the
    // kSSLSessionOptionBreakOnServerAuth option
    if (option == kSSLSessionOptionBreakOnServerAuth)
        return noErr;
    else
        return original_SSLSetSessionOption(context, option, value);
}
...
MSHookFunction((void *) SSLHandshake, (void *) replaced_SSLHandshake, (void **)
    &original_SSLHandshake);
MSHookFunction((void *) SSLSetSessionOption, (void *) replaced_SSLSetSessionOption,
    (void **) &original_SSLSetSessionOption);
MSHookFunction((void *) SSLCreateContext, (void *) replaced_SSLCreateContext, (void
    **) &original_SSLCreateContext);
```

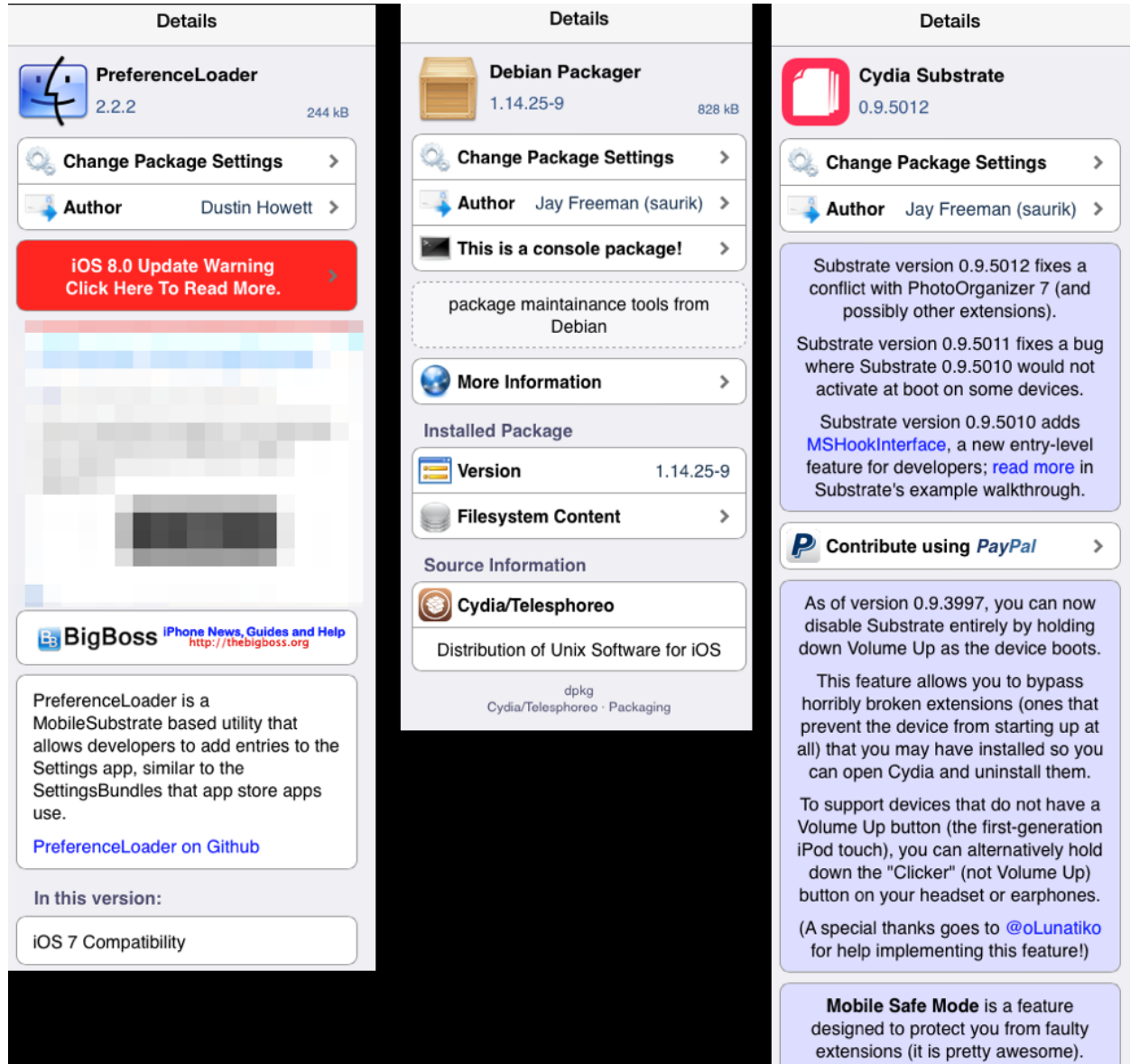
<https://github.com/iSECPartners/ios-ssl-kill-switch>

iOS SSL Kill Switch

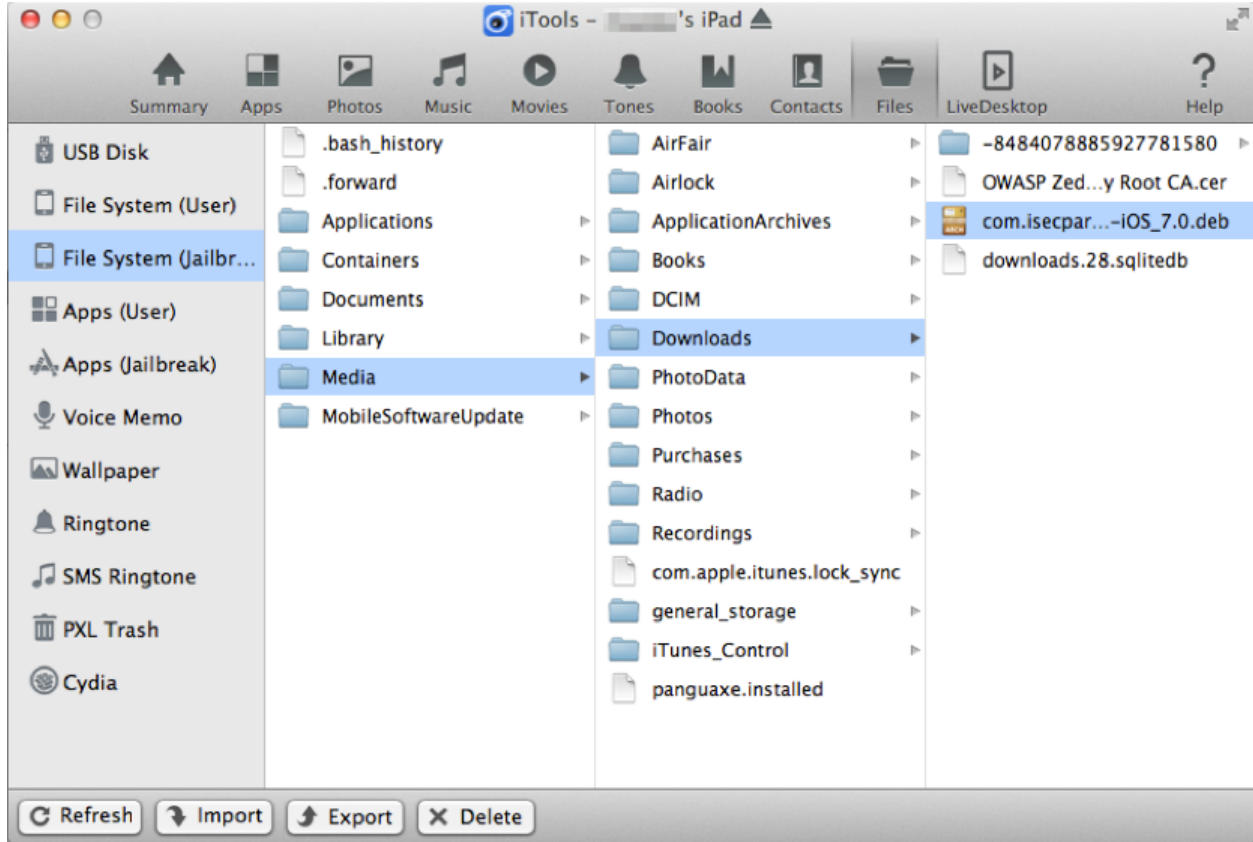
Bu eklenti / tweak Cydia Substrate üzerinde çalışan ve çeşitli sistem fonksiyonlarına kanca atarak, Certificate Pinning'i devre dışı bırakıp, tüm sertifikalar ile iletişim kurulmasını sağlar.

iOS SSL Kill Switch için:

1. Test yapılacak cihazın jailbreak yapılmış olması gerekiyor. Cydia Substrate sadece jailbreak'li cihazlarda çalışır. Bu yazının yazıldığı tarihte iOS 8 yeni çıktığı için henüz jailbreak edilebilir durumda değil ancak 7.1.2 ve öncesi versiyonlar için jailbreak mevcut. Nasıl jailbreak yapılabileceği ile ilgili adımlar ve uygun yazılımlar <http://jailbreak-me.info/> adresinden bulunabilir.
2. Jailbreak yapılmış cihaz üzerine Cydia Store'dan aşağıdaki eklentiler yüklemelidir;
 - a. PreferenceLoader
 - b. Debian Packager
 - c. Cydia Substrate



3. Yapacağımız konsol işlemleri için Cydia Store'dan bir "Mobile Terminal" uygulaması yüklemek uygun olacaktır. Ama isterseniz cihaza SSH ile bağlanıp da aynı işlemleri yapabilirsiniz.
4. Bilgisayarınıza <https://github.com/iSECPartners/ios-ssl-kill-switch/releases> adresinden iOS SSL Kill Switch deb paketini indirip iTools veya iFunBox gibi bir uygulama ile bu paketi test yapmak istediğimiz cihaza kopyalıyoruz.

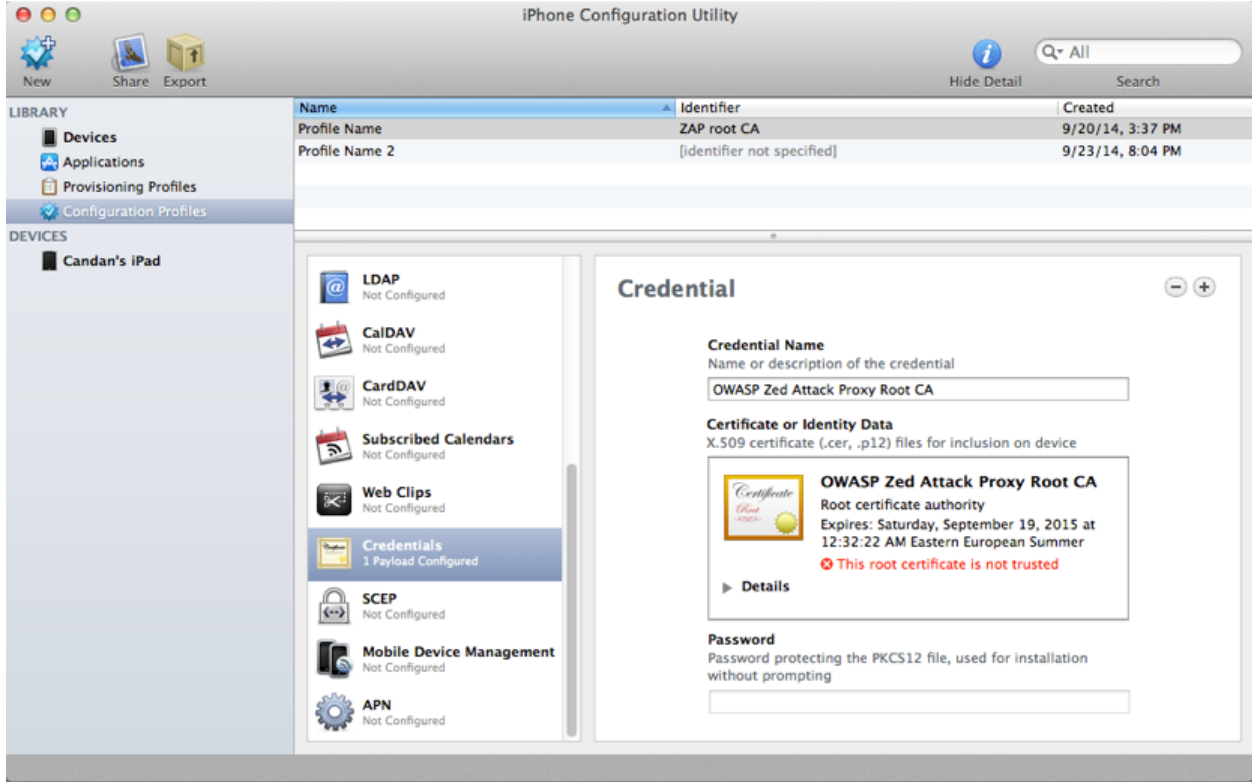


5. Cihaza bağlandıktan sonra konsolda root erişimi alıp (jailbreak'li iOS cihazlarda default root şifresi "alpine"dir) aşağıdaki şekilde deb paketini kurup, SpringBoard'u yeniden başlatıyoruz;

```
dpkg -i com.isecpartners.nabla.sslkillswitch_v0.6-iOS_7.0.deb
killall -HUP SpringBoard
```

```
Candans-iPad:~ root# pwd
/var/root
Candans-iPad:~ root# cd /private/var/mobile/
Candans-iPad:/private/var/mobile root# ls
Applications/ Documents/ Media/
Containers/ Library/ MobileSoftwareUpdate/
Candans-iPad:/private/var/mobile root# cd Media/
Candans-iPad:/private/var/mobile/Media root# cd Downloads/
Candans-iPad:/private/var/mobile/Media/Downloads root# dpkg -i com.isecpartners.nabla.sslkillswitch_v0.6-iOS_7.0.deb
Selecting previously deselected package com.isecpartners.nabla.sslkillswitch.
(Reading database ... 1896 files and directories currently installed.)
Unpacking com.isecpartners.nabla.sslkillswitch (from com.isecpartners.nabla.sslkillswitch_v0.6-iOS_7.0.deb) ...
Setting up com.isecpartners.nabla.sslkillswitch (0.6-1) ...
Candans-iPad:/private/var/mobile/Media/Downloads root#
```

6. iOS Cihaza test yapmak istediğimiz proxy uygulamasının (ZAP, Fiddler veya Burp) root sertifikasını yüklüyoruz. Bunu yapmanın birkaç yolu var, kullanmak istediğiniz metodu size bırakıyorum.
 - a. Mail atıp, iOS cihazın içinden bu sertifikayı kurabiliriz
 - b. Bir web sunucusuna yükleyip safari üzerinden bu sertifikayı yükleyebiliriz
 - c. iPhone Configuration Utility kullanarak ilgili sertifikayı bir profil içerisine yükleyip sonra bu profili iOS cihaza yükleyebilirsiniz. Ben aşağıda görüldüğü şekilde bu metodu kullandım.



7. iOS cihazın proxy ayarlarını test yapmak istediğimiz proxy uygulamasına yönlendirip HTTPS trafiğin Certificate Pinning kullanan uygulamalarda dahi sertifika uyarısı almadan decrypt edilebildiğini aşağıdaki şekilde görebilmekteyiz.

Örnek İstek (Request):

```
GET
https://api.twitter.com/1.1/account/verify_credentials.json?application_id=com.apple.P
references HTTP/1.1
Authorization: OAuth oauth_nonce="7E3E57E0-16CA-4D40-B7DB-511XXXXXXX",
oauth_signature_method="HMAC-SHA1", oauth_timestamp="1411218235",
oauth_consumer_key="WXZE9QillkIZpTANxxxxxx", oauth_token="2413629533-
pNKcOWzwYGmWKhKzE1A7qz36Nm1xxxxxxxx",
oauth_signature="NhD0X4C%2BDIaETSSnDYUXXXXXXX", oauth_version="1.0"
Accept: */*
Cookie: guest_id=v1%3A141121693454xxxxxx
Accept-Language: en-us
Connection: keep-alive
```

```
Proxy-Connection: keep-alive
User-Agent: Settings/1.0 CFNetwork/672.1.15 Darwin/14.0.0
Host: api.twitter.com
```

Örnek Cevap (Response):

```
{"id":2413629533,"id_str":"2413629533","name":"Anonymous","screen_name":"xxxxxxx","location":"Earth","description":"","url":"http://t.co/LC0vThu4ID","entities":{"url":{"urls":[{"url":"http://t.co/LC0vThu4ID","expanded_url":"http://www.google.com","display_url":"google.com","indices":[0,22]}]},"description":{"urls":[]},"protected":false,"followers_count":4,"friends_count":2,"listed_count":0,"created_at":"Sat Mar 15 21:00:21 +0000 2014","favourites_count":5,"utc_offset":null,"time_zone":null,"geo_enabled":false,"verified":false,"statuses_count":14,"lang":"en","status":{"created_at":"Wed Apr 02 22:15:12 +0000 2014","id":451483009699819521,"id_str":"451483009699819521","text":"Ger\u00e7ek para de\u011fil belki ama yinede kur de\u011flerleri \u00f6nemli\nBitcoin = $ 437.33\nLitecoin = $ 11.20\nPeercoin = $ 1.77\nMastercoin = $ 32.00","source":"\u00cahref=\"http://twitter.com/download/iphone\" rel=\"nofollow\"\u00eTwitter for iPhone\u00c/a\u00e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,...
```

Not: Cihazınızda bu dökümanda gösterilen değişiklikleri yapmak çok büyük güvenlik zaafiyeti oluşturmaktadır. Lütfen testlerinizi test cihazları veya emülatörler üzerinde yapınız.

Referanslar

https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

<https://www.owasp.org/images/f/f1/Pubkey-pin-supplement.pdf>

https://www.owasp.org/index.php/Pinning_Cheat_Sheet

[http://media.blackhat.com/bh-us-](http://media.blackhat.com/bh-us-12/Turbo/Diquet/BH_US_12_Diquet_Osborne_Mobile_Certificate_Pinning_Slides.pdf)

[12/Turbo/Diquet/BH_US_12_Diquet_Osborne_Mobile_Certificate_Pinning_Slides.pdf](http://media.blackhat.com/bh-us-12/Turbo/Diquet/BH_US_12_Diquet_Osborne_Mobile_Certificate_Pinning_Slides.pdf)

<https://developer.android.com/training/articles/security-ssl.html>

<https://developer.android.com/reference/javax/net/ssl/HttpsURLConnection.html>

<https://developer.android.com/reference/javax/net/ssl/SSLSocketFactory.html>

<https://developer.android.com/reference/javax/net/ssl/TrustManagerFactory.html>

<https://android.googlesource.com/platform/external/apache->

[http/+2f0e9505248dc19d7fcfed4c1efd7c3daf863f03/src/org/apache/http/conn/ssl/SSLSocketFactory.java](http://2f0e9505248dc19d7fcfed4c1efd7c3daf863f03/src/org/apache/http/conn/ssl/SSLSocketFactory.java)

<https://github.com/iSECPartners/Android-SSL-TrustKiller>

<http://www.cydiasubstrate.com/>

<http://www.cydiasubstrate.com/inject/android/>

<https://developer.apple.com/library/prerelease/mac/documentation/Security/Reference/secureTransportRef/>