جلوگیری از حمله مرد میانی در IOS با استفاده از SSL-Pinning

علیرضا ارجمند

۱۳۹۹/۵/۱۷

تحقیق پایانی درس برنامه‌سازی موبایل

چکیده

امروزه به علت ارزش زیاد اطلاعات ذخیره شده بر روی لوازم الکترونیکی مانند کامپیوترهای شخصی و تلفن‌های همراه این ابزار بسیار مورد حمله افراد شرور هستند. قصد اکثر این حملات استخراج غیرقانونی اطلاعات یک فرد خاص بدون اطلاع وی است. این اطلاعات می‌توانند پس از استخراج برای باج گیری، رسواسازی و یا حتی حمله‌های بزرگتر مورد استفاده قرار گیرند. در این مقاله به طور خاص می‌خواهیم "حملات مرد میانی" را بر روی تلفن‌های همراه IOS مورد بررسی قرار دهیم. برنامه‌ها در IOS به طور کلی درگیر امنیت ارتباط برقرار شده با سرور نمی‌شوند و امنیت را به Certificate های تایید شده به وسیله خود IOS می‌سپرند، حمله کننده می‌تواند با تولید یک Certificate معتبر ارتباط میان برنامه و سرور را شنود کند، در اینجا ما در تلاشیم تا با اجرای یک سری مراحل برنامه تحت IOS خود را امن‌تر کنیم. در ابتدا به مفاهیم پایه‌ای مانند ارتباطات SSL و TLS می‌پردازیم، سپس توضیحی کلی از نحوه حمله مرد میانی می‌دهیم، پس از آن به بررسی Certificate ها می‌پردازیم، در ادامه نحوه امن سازی ارتباط یک برنامه تحت IOS را به سرور توضیح می‌دهیم و در آخر کار کلی را تست می‌کنیم.

درباره SSL و TLS

TLS یا Transport Layer Security یک ورژن جدیدتر و بهتر شده از SSL و یا Secure Sockets Layer است و هدف اصلی آن ایجاد امنیت در ارتباطات افراد است به شکلی که امکان شنود و تغییر داده توسط فرد خارجی وجود نداشته باشد. ارتباطات TLS به طور کلی در ۳ مرحله انجام می‌شوند(شکل یک):

مرحله صفر: TLS بر روی TCP اجرا می‌شوند و قبل از شروع کار TLS، ارتباط TCP باید برقرار باشد.

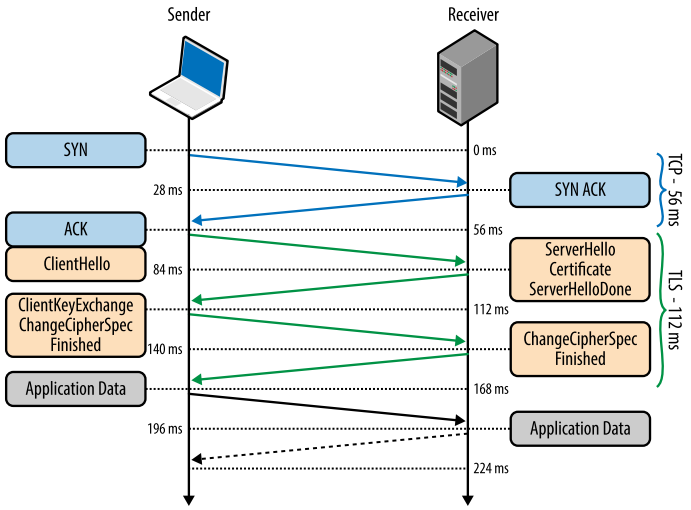
مرحله یک: در مرحله Hello کلاینت ورژن‌های TLS و نحوه رمزنگاری‌های قابل پشتیبانی را برای سرور می‌فرستد و سرور درکنار انتخاب یک ورژن و رمزنگاری، به تعداد یک یا بیشتر Certificate برای کلاینت برمی‌گرداند.

سپس کلاینت در این بخش Certificate ها را بررسی و در صورت معتبر بودن این مدارک به مرحله بعدی می‌رود.

مرحله دو: کلاینت با استفاده از Certificate سرور public key آن را استخراج کرده و با سرور بر سر یک کلید به توافق می‌رسد.

مرحله سه: در این مرحله کلاینت و سرور با استفاده از کلید مرحله قبل به تبادل اطلاعات می‌پردازند.

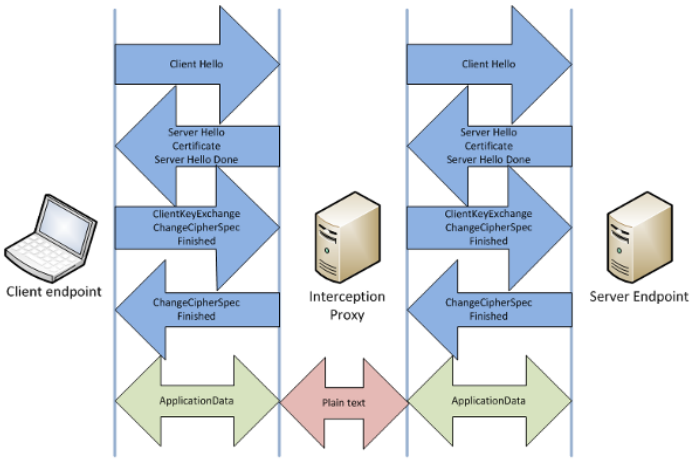
نکته مهم در این بخش این است که به علت زمان بر بودن encode و decode به روش RSA، کلاینت و سرور تنها یک بار از RSA برای برقراری ارتباط و دستیابی به یک کلید Symmetric مشترک برای تبادل اطلاعات استفاده می‌کنند، پس از آن انتقال اطلاعات به وسیله کلید Symmetric انجام می‌شود.



شکل ۱

مشکل اصلی این روش

قدرت این پروتکل به نحوه بررسی Certificate و اعتبار آن است، اگر یک حمله کننده دارای یک Certificate معتبر باشد می‌تواند مانند شکل دو میان سرور و کلاینت قرار گیرد.



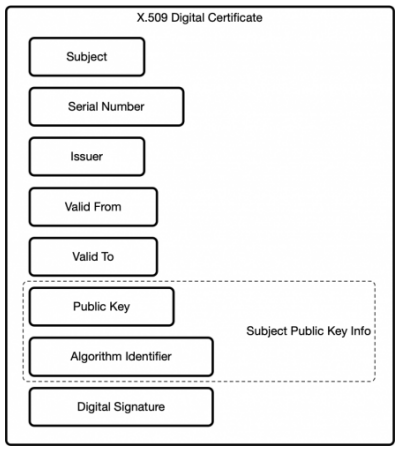
شکل ۲

به این نوع حمله، حمله مرد میانی می‌گویند. حمله کننده در برابر سرور خود را به جای کلاینت و برای کلاینت خود را به جای سرور جا می‌زند و با هردوی آنها جداگانه بر سر کلید به توافق می‌رسد و می‌تواند تمامی اطلاعات را شنود کند.

در اینجا این سوال پیش می‌آید که بررسی Certificate ها چگونه است؟ در بخش بعدی به این مهم می‌پردازیم.

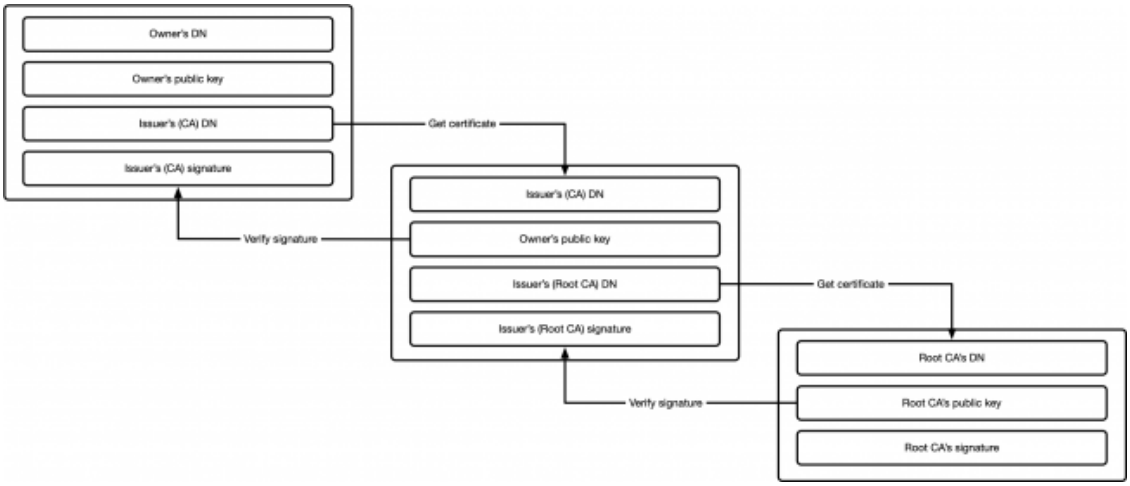
بررسی Certificate ها

یک Certificate به طور کلی یک فایل است که دارای اطلاعات کلی از طرف سرور است، این اطلاعات را می‌توانید در شکل سه مشاهده کنید.



شکل ۳

بررسی تک تک این موارد از حوصله بحث خارج است. به طور کلی چندین CA Authority شناخته شده وجود دارند و با ایجاد یک زنجیره اعتماد به مانند شکل چهار، یک Certificate تایید می‌شود. حمله کننده می‌تواند با نفوذ و یا هک کردن یکی از این CA Authority ها این پروتکل را دور بزند که به عنوان یک مثال می‌توان از هک شدن DigiNotar در ۲۰۱۱ یاد کرد.



شکل ٤

برای امن‌تر کردن برنامه IOS چه کار می‌توان کرد؟

برنامه IOS برای برقراری امنیت به IOS Trust Store اعتماد می‌کند ولی برای اینکه از امنیت برنامه خود مطمئن شویم می‌توانیم از SSL Pinning استفاده کنیم، در این روش با یکی از دو راه که در ادامه توضیح خواهیم داد می‌توانیم برنامه خود را با یک یا چند Certificate از سرور های مورد نیاز به طور مستقیم آشنا کنیم. در این روش به اصطلاح چند آدرس مشخص را White List کرده و دیگر نیازی به اعتماد به CA Authority ها و یا IOS Trust Store نیست. روش‌های SSL Pinning عبارت‌اند از:

Pinning the certificate : در این روش با دانلود کردن Certificate سرور مورد نظر آن را به طور مستقیم به برنامه می‌شناسانیم.

Pinning the public key : در این روش می‌توانیم Public Key سرور را در برنامه Hard Code کنیم.

هردوی این روش‌های می‌توانند مورد استفاده قرار گیرند و تصمیم بر عهده برنامه نویس است.

پیاده‌سازی SSL Pinning با استفاده از Alamofire 5

با استفاده از Alamofire 5 و دو تابع با اسامی زیر می‌توان ویژگی SSL Pinning را پیاده سازی کرد:

- PinnedCertificatesTrustEvaluator
- PublicKeysTrustEvaluator

در ادامه به استفاده از روش اول یعنی اضافه کردن Certificate به کد می‌پردازیم. در اینجا باید Certificate مربوطه را دانلود و به فایل پروژه خود اضافه کنید سپس می‌توانید از درون کد به عنوان Bundle به آن دسترسی داشته باشید. با استفاده از کد شکل پنج می‌توانید Certificate را از bundle استخراج کنید و در کد استفاده کنید.

```
struct Certificates {
  static let stackExchange =
    Certificates.certificate(filename: "stackexchange.com")

  private static func certificate(filename: String) -> SecCertificate {
    let filePath = Bundle.main.path(forResource: filename, ofType: "der")!
    let data = try! Data(contentsOf: URL(fileURLWithPath: filePath))
    let certificate = SecCertificateCreateWithData(nil, data as CFData)!

    return certificate
  }
}
```

شکل ۵

در این کد SecCertificateCreateWithData فایل Certificate را از فایل DER-encoded استخراج می‌کند.

حال در زمان باز کردن یک TLS Session به جای سپردن کار به TLS نیاز به کمی تغییرات مانند شکل شش داریم.

```
// 1
let evaluators = [
  "api.stackexchange.com":
    PinnedCertificatesTrustEvaluator(certificates: [
      Certificates.stackExchange
    ])
]

let session: Session

// 2
private init() {
  session = Session(
    serverTrustManager: ServerTrustManager(evaluators: evaluators)
  )
}
```

شکل ۶

در شکل شش با ساختن یک Dictionary به نام evaluators تمامی Certificate هایی را که به آنان اعتماد داریم وارد می‌کنیم و سپس آن را به serverTrustManager پاس می‌دهیم تا از آن استفاده کند. اکنون در صورتی که Certificate دریافت شده از سرور با Certificate مشخص شده در کد همخوانی نداشته باشد ارتباط برقرار نشده و به ارور برخورد می‌کنیم، برای نوشتن یک ریکوئست هم می‌توان به شکل زیر عمل کرد:

```
NetworkClient.request(yourData)
  .responseDecodable { (response: <YourDataType>) in
    switch response.result {
    case .success(let value):
      Do Things Here
      case .failure(let error):
      let isServerTrustEvaluationError =
        error.asAFError?.isServerTrustEvaluationError ?? false
      let message: String
      if isServerTrustEvaluationError {
        message = "Certificate Pinning Error"
      } else {
        message = error.localizedDescription
      }
      self.presentError(withTitle: "Oops!", message: message)
    }
}
```

شکل ۷

در کد شکل هفت می‌بینیم که در صورت موفقیت مراحل مانند قبل است ولی در صورت مواجه شدن با خطا و اشتباه بودن Certificate مربوطه را به کاربر نشان می‌دهیم.

تست کار کردن SSL Pinning

در این مرحله برای تست کردن کار خود می‌توانید به راحتی از یک proxy server که Certificate آن را به برنامه خود اضافه نکرده‌اید استفاده کنید و مشاهده کنید که پیغام "Certificate Pinning Error" نمایش داده شود. علت این پیغام را می‌توانید در شکل دو مشاهده کنید زیرا یک proxy server میان شما و سرور مورد نظر قرار می‌گیرد و اکنون برنامه شما این مهم را تشخیص می‌دهد و جلوی آن را می‌گیرد.

منابع:

[۱] Preventing Man-in-the-Middle Attacks in iOS with SSL Pinning

[۲] Transport Layer Security (TLS)

[۳] WikiPedia Man In The Middle Attack

[٤] Does https prevent man in the middle attacks by proxy server?