『IOS』

**모바일 취약점**

**진단 보고서**

**2018.09.30**



**제∙개정 이력**

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **제∙개정 일자** | **내용** | **제∙개정자** |
| 1.0 | 2018.09.30 | 신규 작성 | 나은빈 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

목 차

1. 개요 7

1.1. 수행 대상 7

1.2. 목적 7

1.3. 수행 인력 8

1.4. 수행 항목 8

1.4.1. 취약점 진단 목록 8

1.5. 점검 기준 9

1.5.1. 취약점 진단 기준 9

2. 총평 11

2.1. iGoat 11

3. 상세결과 11

3.1. 취약점 진단 11

3.1.1. 결과요약 11

3.1.2. 불충분한 전송 계층 보호 12

3.1.3. 안전하지 않은 데이터 저장 15

3.1.4. 의도치 않은 데이터 유출 26

3.1.5. 클라이언트 측 공격 31

3.1.6. 신뢰할 수 없는 입력을 통한 보안 결정 33

3.1.7. 리버스 엔지니어링 35

4. 대응방안 36

4.1. 취약점 보안 대책 36

4.1.1. 불충분한 전송계층 보호 36

4.1.2. 안전하지 않은 데이터 저장 37

4.1.3. 의도치 않은 데이터 유출 37

4.1.4. 클라이언트 측 공격 38

4.1.5. 신뢰할 수 없는 입력을 통한 보안 결정 38

4.1.6. 리버스 엔지니어링 38

표 목차

[표 1] 수행 대상 7

[표 2] 수행 인력 8

[표 3] 진단 항목 8

[표 4] 취약점 진단 기준 9

[표 5] 도출된 취약점 목록(iGoat) 11

[표 6] 진단 결과 11

[표 7] 불충분한 전송계층 보호 대응방안 37

[표 8] 안전하지 않은 데이터 저장 대응방안 37

[표 9]의도치 않은 데이터 유출 대응방안 37

[표 10] 클라이언트 측 공격 대응방안 38

[표 11] 신뢰할 수 없는 입력을 통한 보안 결정 대응방안 38

[표 12] 리버스 엔지니어링 대응방안 38

그림 목차

[그림 1] macOs에서의 웹 프록시 설정 12

[그림 2] Burp Suite설정을 통한 HTTP Proxy 설정 13

[그림 3] 정보가 평문으로 노출 13

[그림 4] macOs에서의 웹 프록시 설정 14

[그림 5]Burp Suite설정을 통한 HTTP Proxy 설정 14

[그림 6] 주요 정보가 평문으로 노출 15

[그림 7] 현재 실행되는 프로세스의 경로 확인 15

[그림 8] find를 사용하여 데이터베이스 파일 경로 확인 15

[그림 9] 데이터 베이스 파일이 존재하는 경로 16

[그림 10]SQL Query를 이용해 데이터베이스 정보 확인 16

[그림 11] 찾은 데이터를 문제에 입력 16

[그림 12] 주요 정보 입력 17

[그림 13] 저장된 데이터의 경로 및 데이터 확인 17

[그림 14]데이터베이스 내 정보 확인 18

[그림 15] 찾은 정보를 문제에 입력 18

[그림 16] YapDatabase의 경로 18

[그림 17] DB browser를 통해 email, password 확인 19

[그림 18]찾은 데이터를 문제에 입력 19

[그림 19]cache의 경로 20

[그림 20] 찾은 데이터를 문제에 입력 20

[그림 21] Cookie 데이터의 경로와 내용 확인 21

[그림 22] 찾은 데이터를 문제에 입력 21

[그림 23] Realm 데이터가 존재하는 경로 21

[그림 24] 010editor로 realm의 내용 확인 22

[그림 25] 찾은 데이터를 문제에 입력 22

[그림 26]NSUserDefault로 저장된 파일의 경로 23

[그림 27] 파일의 내용 확인 23

[그림 28] 찾은 데이터를 문제에 입력 23

[그림 29] Plist 파일이 존재하는 경로 24

[그림 30] 파일 내용 확인 24

[그림 31] 찾은 데이터를 문제에 입력 24

[그림 32]Hopper를 통해 파일의 이름과 경로를 찾음 25

[그림 33] 파일이 위치하는 경로 25

[그림 34] 파일의 내용 확인 25

[그림 35] 찾은 데이터를 문제에 입력 26

[그림 36] 데이터를 입력하고 제출 27

[그림 37] log파일이 존재하는 경로 27

[그림 38] 디바이스 내의 로그에서 입력한 데이터 노출 확인 27

[그림 39] log파일이 존재하는 경로 28

[그림 40] 로그 내에 암호화된 키의 정보가 노출 28

[그림 41] 찾은 데이터를 문제에 입력 28

[그림 42] 데이터 입력 후 기기 홈 버튼 입력 29

[그림 43] 스크린샷 기능을 통해 파일이 저장되는 경로에 파일 존재 유무, 내용 확인 29

[그림 44] 데이터를 클립보드 내에 복사 30

[그림 45] 실행파일에 pid와 cycript를 이용하여 데이터 확인 30

[그림 46] 주요 데이터 입력 31

[그림 47] 키보드 캐시 파일의 위치 31

[그림 48] 키보드 캐시 파일의 내용 확인 31

[그림 49] Script 내용을 삽입 후 성공 32

[그림 50]Sql 코드를 주입 33

[그림 51] 공격 전, 후 화면 33

[그림 52] 주요 데이터를 입력 34

[그림 53] log를 통해 URL Scheme관련 파라미터 확인 34

[그림 54] URL Scheme을 활용 35

[그림 55] Hopper를 통해 정답을 찾음 36

[그림 56] 찾은 데이터를 문제에 입력 36

1. 개요
   1. 수행 대상

[표 1] 수행 대상

|  |  |
| --- | --- |
| 대상 | 구 분 |
| iGoat.ipa | 어플리케이션 |

* 1. 목적

IOS 애플리케이션의 취약점 진단을 통해 IOS의 전반적인 이해를 걸쳐 취약점이 발견될 시 이에 맞는 보호 대책을 수립하고자 보고서를 작성하였다.

* 1. 수행 인력

[표 2] 수행 인력

|  |  |  |
| --- | --- | --- |
| 소 속 | 기 수 | 성 명 |
| 아이투섹 | 서울 5기 | 나은빈 |

* 1. 수행 항목
     1. 취약점 진단 목록

[표 3] 진단 항목

|  |  |  |
| --- | --- | --- |
| **유형** | **수행항목** | **항목설명** |
| 불충분한 전송계층 보호 | 비암호화 통신(http) | HTTP프로토콜을 이용해 주요정보를 평문으로 전송하거, GET방식을 통해 외부자에게 유출될 수 있는 취약점으로, 네트워크 스니핑 또는 중간자 공격을 통해 주요정보가 제 3자에게 노출될 수 있는 취약점이다. |
| 안전하지 않은 데이터 저장 | 데이터베이스 내 주요정보 저장(CoreData) | 주요정보를 평문으로 저자하거나, 별도의 접근제한을 걸지 않아 악의적인 목적을 가진 사용자 또는 어플리케이션이 접근 시 주요정보를 확인할 수 있는 취약점이다. |
| Keychain 내 주요 테이터 저장 |
| NSUserDefaults로 주요 데이터 저장 |
| Plist 내 주요 데이터 확인 |
| 의도치 않은  데이터 유출 | Device Log를 통한 데이터 유출 | 타 어플리케이션에서 접근 가능한 데이터영역에 민감한 정보를 저장 시 발생할 수 있는 취약점이다. |
| 스크린샷 파일을 통한 데이터 유출 |
| 클립보드 내 데이터 유출 |
| 키보드 로그를 통한 데이터 유출 |
| 클라이언트 측 공격 | Cross Site Scripting | SQL Injection, Local File Inclustion, XML Injection, Javascript Injectjon, Binary Injection 등 클라이언트에서 발생할 수 있는 공격이다.  주요 정보 유출(암호, 세션 값, 개인 정보) 또는 DoS 등의 피해가 있을 수 있는 취약점이다.. |
| 신뢰할 수 없는 입력을 통한 보안 결정 | URL Scheme을 통한 호출 | 어플리케이션이나 프로세스 간 통신 시 발생할 수 있는 취약점이다. |
| 역공학 | 역공학 | 이미 제작 완료되어 배포된 시스템을 거꾸로 추적하여 원본문서나 소스코드, 알고리즘, 설계 기술 등의 정보를 얻어내는 취약점이다 |

* 1. 점검 기준
     1. 취약점 진단 기준

[표 4] 취약점 진단 기준

|  |  |
| --- | --- |
| **진단 결과** | **설 명** |
| **O** | 취약점이 발견되지 않았음 |
| **P** | 악용될 수 있음, 보안조치 권고 |
| **X** | 운영에 악영향을 줄 수 있는 취약점이 발견됨  신속한 보안조치 필요 |
| **N/A** | 해당사항이 없거나 진단 항목이 없음 |

1. 총평
   1. iGoat

[표 5] 도출된 취약점 목록(iGoat)

|  |  |  |
| --- | --- | --- |
| **대상** | **도출 항목** | **취약점 내용** |
| iGoat.ipa | 비암호화 통신(http) | 정보를 전송할 때 비암호화 통신을 사용하여 정보가 평문으로 노출된다. |
| 데이터베이스 내 주요정보 저장(CoreData) | 데이터베이스 내 주요 정보가 평문으로 저장되어 정보가 노출된다. |
| Keychain 내 주요 테이터 저장 | 주요 데이터가 Keychain내에 평문으로 저장되어 정보가 노출된다. |
| NSUserDefaults로 주요 데이터 저장 | 특정 경로에 데이터가 평문으로 저장되어 정보가 노출된다. |
| Plist 내 주요 데이터 확인 | 주요 정보가 Plist파일에 평문으로 저장해 정보가 노출된다. |
| Device Log를 통한 데이터 유출 | 사용자 정보 저장 시 평문으로 데이터가 로그에 노출된다. |
| 스크린샷 파일을 통한 데이터 유출 | 사용자가 입력하는 데이터를 스크린샷 파일로 저장하여 데이터가 노출된다. |
| 클립보드 내 데이터 유출 | 클립보드 내에 저장된 데이터가 노출된다. |
| 키보드 로그를 통한 데이터 유출 | 키보드 로그가 저장되어 데이터가 노출된다 |
| Cross Site Scripting | 악성 스크립트 입력을 통해 각종 악의적인 행동을 취할 수 있다. |
| URL Scheme을 통한 호출 | URL Scheme를 통한 호출로 설정 값 변경, 민감한 기능이 동작될 수 있다. |
| 리버스 엔지니어링 | 이미 제작 완료되어 배포된 시스템을 거꾸로 추적하여 원본문서나 소스코드, 알고리즘, 설계 기술 등의 정보를 얻어 낼 수 있다. |

1. 상세결과
   1. 취약점 진단
      1. 결과요약

[표 6] 진단 결과

|  |  |  |
| --- | --- | --- |
| **유형** | **진단 항목** | **진단결과** |
| 불충분한 전송계층 보호 | 비암호화 통신(http) | **X** |
| 안전하지 않은  데이터 저장 | 데이터베이스 내 주요정보 저장(CoreData) | **X** |
| Keychain 내 주요 테이터 저장 | **X** |
| NSUserDefaults로 주요 데이터 저장 | **X** |
| Plist 내 주요 데이터 확인 | **X** |
| 의도치 않은  데이터 유출 | Device Log를 통한 데이터 유출 | **X** |
| 스크린샷 파일을 통한 데이터 유출 | **X** |
| 클립보드 내 데이터 유출 | **X** |
| 키보드 로그를 통한 데이터 유출 | **X** |
| 클라이언트 측 공격 | Cross Site Scripting | **X** |
| 신뢰할 수 없는 입력을 통한 보안 결정 | URL Scheme을 통한 호출 | **X** |
| 리버스 엔지니어링 | 리버스 엔지니어링 | **X** |

* + 1. 불충분한 전송 계층 보호
       1. 비암호화 통신 (HTTP)

|  |  |
| --- | --- |
| **비암호화 통신(HTTP)** | |
| **개 요** | HTTP프로토콜을 이용해 주요정보를 평문으로 전송하거, GET방식을 통해 외부자에게 유출될 수 있는 취약점으로, 네트워크 스니핑 또는 중간자 공격을 통해 주요정보가 제 3자에게 노출될 수 있는 취약점이다. |

* + - * 1. Server Communication

macOS에서의 웹 프록시(HTTP ) 설정

|  |
| --- |
| [그림 1] macOs에서의 웹 프록시 설정 |

Burp Suite설정을 통한 HTTP Proxy 설정

|  |
| --- |
| [그림 2] Burp Suite설정을 통한 HTTP Proxy 설정 |

비암호화 통신 실습

|  |  |
| --- | --- |
|  |  |

[그림 3] 정보가 평문으로 노출

* + - * 1. Remote Authentication

macOS에서의 웹 프록시(HTTP ) 설정

|  |
| --- |
| [그림 4] macOs에서의 웹 프록시 설정 |

Burp Suite설정을 통한 HTTP Proxy 설정

|  |
| --- |
| [그림 5]Burp Suite설정을 통한 HTTP Proxy 설정 |

비암호화 통신 실습

|  |  |
| --- | --- |
|  |  |

[그림 6] 주요 정보가 평문으로 노출

* + 1. 안전하지 않은 데이터 저장

|  |  |
| --- | --- |
| **안전하지 않은 데이터 저장** | |
| **개 요** | 주요정보를 평문으로 저자하거나, 별도의 접근제한을 걸지 않아 악의적인 목적을 가진 사용자 또는 어플리케이션이 접근 시 주요정보를 확인할 수 있는 취약점이다. 스마트폰 분실 혹은 공격자에 의한 스마트폰 권한 탈취 시 해당되는 주요정보들이 공격자에게 노출이 된다. |

* + - 1. 데이터베이스 내 주요정보 저장

개발자가 주요한 데이터를 데이터베이스 내에 저장함으로써 발생하는 취약점

* + - * 1. CoreData

현재 실행되는 프로세스의 경로 확인

|  |
| --- |
| [그림 7] 현재 실행되는 프로세스의 경로 확인 |

어플리케이션 데이터베이스 파일 찾기(1/2)

|  |
| --- |
| [그림 8] find를 사용하여 데이터베이스 파일 경로 확인 |

어플리케이션 데이터베이스 파일 찾기(2/2)

|  |
| --- |
| [그림 9] 데이터 베이스 파일이 존재하는 경로 |

SQL Query를 이용해 데이터베이스 정보 확인

|  |
| --- |
| [그림 10]SQL Query를 이용해 데이터베이스 정보 확인 |

찾은 정보를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 11] 찾은 데이터를 문제에 입력

* + - * 1. Local Data Storage

주요 정보를 입력

|  |
| --- |
|  |

[그림 12] 주요 정보 입력

저장된 데이터를 확인

|  |
| --- |
| [그림 13] 저장된 데이터의 경로 및 데이터 확인 |

* + - * 1. CouchBase Storage

데이터베이스에 정보 삽입

|  |
| --- |
| [그림 14]데이터베이스 내 정보 확인 |

찾은 정보를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 15] 찾은 정보를 문제에 입력

* + - * 1. Yap Storage

YapDatabase의 경로

|  |
| --- |
|  |

[그림 16] YapDatabase의 경로

DB browser를 통해 email, password 확인

|  |
| --- |
|  |
|  |

[그림 17] DB browser를 통해 email, password 확인

찾은 데이터를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 18]찾은 데이터를 문제에 입력

* + - * 1. Webkit Cache

Cache의 경로

|  |
| --- |
|  |

[그림 19]cache의 경로

찾은 데이터를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 20] 찾은 데이터를 문제에 입력

* + - 1. Cookie에 주요 데이터 저장
         1. Cookie 데이터가 존재하는 경로와 파일 확인

|  |
| --- |
|  |

[그림 21] Cookie 데이터의 경로와 내용 확인

* + - * 1. 찾은 데이터를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 22] 찾은 데이터를 문제에 입력

* + - 1. Realm에 주요 데이터 저장
         1. Realm데이터가 존재하는 경로

|  |
| --- |
|  |

[그림 23] Realm 데이터가 존재하는 경로

* + - * 1. 010editor로 realm의 내용 확인

|  |
| --- |
|  |

[그림 24] 010editor로 realm의 내용 확인

* + - * 1. 찾은 데이터를 문제에 입력

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 25] 찾은 데이터를 문제에 입력

* + - 1. NSUserDefaults로 주요 데이터 저장

개발자가 주요한 데이터를 NSUserDefaults함수로 저장해 경로가 노출되고, 데이터를 확인할 수 있는 취약점

* + - * 1. NSUserDefault로 저장된 파일의 경로

|  |
| --- |
|  |

[그림 26]NSUserDefault로 저장된 파일의 경로

* + - * 1. 파일의 내용 확인

|  |
| --- |
|  |

[그림 27] 파일의 내용 확인

* + - * 1. 찾은 데이터를 문제에 입력

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 28] 찾은 데이터를 문제에 입력

* + - 1. Plist 내 주요 데이터 확인

개발자가 주요 데이터를 Plist파일에 저장해 발생하는 취약점

* + - * 1. Plist Storage

Plist 파일이 존재하는 경로

|  |
| --- |
|  |

[그림 29] Plist 파일이 존재하는 경로

파일 내용을 확인

|  |
| --- |
|  |

[그림 30] 파일 내용 확인

찾은 데이터를 문제에 입력

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 31] 찾은 데이터를 문제에 입력

* + - * 1. Social Login

Hopper를 통해 파일의 이름과 경로를 찾음

|  |
| --- |
|  |

[그림 32]Hopper를 통해 파일의 이름과 경로를 찾음

파일이 위치하는 경로

|  |
| --- |
|  |

[그림 33] 파일이 위치하는 경로

파일의 내용을 확인

|  |
| --- |
|  |

[그림 34] 파일의 내용 확인

찾은 데이터를 문제에 입력

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 35] 찾은 데이터를 문제에 입력

* + 1. 의도치 않은 데이터 유출

|  |  |
| --- | --- |
| **의도치 않은 데이터 유출** | |
| **개 요** | 타 어플리케이션에서 접근 가능한 데이터영역에 민감한 정보를 저장 시 발생할 수 있는 취약점이다. 개발자 문서에서는 쉽게 확인할 수 없는 OS 또는 프레임워크 내부의 Cache 방식, Key 저장 방식, 메모리 버퍼, 로깅 등의 로직 등이 포함된다. Side-Channel Attack으로도 불린다. |

* + - 1. Log를 통한 데이터 유출

개발자가 디버그 메시지를 남겨 발생하는 취약점으로, 각종 툴을 통해 로그 확인이 가능

* + - * 1. Device Log

데이터를 입력하고 제출

|  |  |
| --- | --- |
|  |  |

[그림 36] 데이터를 입력하고 제출

log파일이 존재하는 경로

|  |
| --- |
|  |

[그림 37] log파일이 존재하는 경로

디바이스 내의 로그에서 입력한 데이터 노출 확인

|  |
| --- |
|  |

[그림 38] 디바이스 내의 로그에서 입력한 데이터 노출 확인

* + - * 1. Random Key Generation

log파일이 존재하는 경로

|  |
| --- |
|  |

[그림 39] log파일이 존재하는 경로

로그 내에 암호화된 키의 정보가 노출

|  |
| --- |
|  |

[그림 40] 로그 내에 암호화된 키의 정보가 노출

찾은 데이터를 문제에 입력

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 41] 찾은 데이터를 문제에 입력

* + - 1. 스크린샷 파일을 통한 데이터 유출

개발자가 화면정보 저장(캐싱)을 위해 특정 행동 시, 스크린샷 파일을 앱 내부에 저장하고 있을 시 발생하는 취약점(주요정보가 노출되어야 함)

* + - * 1. 데이터 입력 후 기기 홈 버튼을 입력

|  |
| --- |
|  |

[그림 42] 데이터 입력 후 기기 홈 버튼 입력

* + - * 1. 스크린샷 기능을 통해 파일이 저장되는 경로에 파일 존재 유무, 내용 확인

|  |
| --- |
|  |
|  |

[그림 43] 스크린샷 기능을 통해 파일이 저장되는 경로에 파일 존재 유무, 내용 확인

* + - 1. 클립보드 내 데이터 유출

클립보드 내에 주요한 데이터를 남길 수 있는 취약점으로, 잘못된 Input Box 등을 사용해 발생

* + - * 1. 데이터를 클립보드 내에 복사

|  |
| --- |
|  |

[그림 44] 데이터를 클립보드 내에 복사

* + - * 1. 실행파일에 pid와 cycript를 이용하여 데이터 확인

|  |
| --- |
|  |

[그림 45] 실행파일에 pid와 cycript를 이용하여 데이터 확인

-[UIPasteboard generalPasteboard].items로 사용자가 클립보드에 저장한 데이터 노출 여부 확인

* + - 1. 키보드 로그를 통한 데이터 유출

Ios 내부에서 키보드 로그를 확인할 수 있으며, 로그 내에 주요한 데이터가 기록될 수 있는 취약점

* + - * 1. 주요 데이터를 입력

|  |
| --- |
|  |

[그림 46] 주요 데이터 입력

* + - * 1. 키보드 캐시 파일의 위치

|  |
| --- |
|  |

[그림 47] 키보드 캐시 파일의 위치

* + - * 1. 키보드 캐시 파일의 내용 확인

|  |
| --- |
|  |

[그림 48] 키보드 캐시 파일의 내용 확인

* + 1. 클라이언트 측 공격

|  |  |
| --- | --- |
| **클라이언트 측 공격** | |
| **개 요** | SQL Injection, Local File Inclustion, XML Injection, Javascript Injectjon, Binary Injection 등 클라이언트에서 발생할 수 있는 공격입니다.  주요 정보 유출(암호, 세션 값, 개인 정보) 또는 DoS 등의 피해가 있을 수 있습니다. |

* + - 1. Cross Site Scripting

View 컴포넌트 내에서 악성 스크립트 입력을 통해 각종 악의적인 행동을 할 수 있는 취약점

* + - * 1. 텍스트 Input 시 반응 확인

Script 내용을 삽입 후 성공

|  |  |
| --- | --- |
|  |  |

[그림 49] Script 내용을 삽입 후 성공

* + - * 1. Sql Injection

Sql 코드를 주입

|  |
| --- |
|  |

[그림 50]Sql 코드를 주입

공격 전, 후 화면

|  |  |  |
| --- | --- | --- |
|  |  |  |

[그림 51] 공격 전, 후 화면

* + 1. 신뢰할 수 없는 입력을 통한 보안 결정

|  |  |
| --- | --- |
| **신뢰할 수 없는 입력을 통한 보안 결정** | |
| **개 요** | 어플리케이션이나 프로세스 간 통신 시 발생할 수 있는 취약점이다.  신뢰할 수 없는 외부 프로세스 또는 어플리케이션의 요청, 입력에 대한 민감한 의사결정을 방지해야 한다. |

* + - 1. URL Scheme을 통한 호출

URL Scheme 또는 View를 호출했을 시 설정값 변경, 민감한 기능이 동작되는 취약점

* + - * 1. 주요 데이터를 입력

|  |
| --- |
|  |

[그림 52] 주요 데이터를 입력

* + - * 1. Log를 통해 URL Scheme관련 파라미터 확인

|  |
| --- |
|  |
|  |

[그림 53] log를 통해 URL Scheme관련 파라미터 확인

* + - * 1. URL Scheme을 활용

|  |  |
| --- | --- |
|  |  |

[그림 54] URL Scheme을 활용

* + 1. 리버스 엔지니어링

|  |  |
| --- | --- |
| **역공학** | |
| **개 요** | 소스 코드, 라이브러리, 알고리즘 및 기타 자산을 결정하는 최종 핵심 바이너리 분석을 포함 IDA Pro, Hopper, otool 및 기타 이진 검사 도구와 같은 소프트웨어는 침입자가 응용 프로그램의 내부 동작을 파악할 수 있도록 한다. 이는 백엔드 서버, 암호화 상수 및 암호 및 지적 재산에 대한 정보를 공개하는 것 외에도 응용 프로그램의 초기 취약점을 악용하는 데 사용될 수 있다. |

* + - 1. Reverse Engineering

이미 제작 완료되어 배포된 시스템을 거꾸로 추적하여 원본문서나 소스코드, 알고리즘, 설계 기술 등의 정보를 얻어내는 취약점이다.

* + - * 1. Hopper를 통해 정답을 찾음

|  |
| --- |
|  |

[그림 55] Hopper를 통해 정답을 찾음

* + - * 1. 찾은 데이터를 문제에 입력

|  |  |
| --- | --- |
|  |  |

[그림 56] 찾은 데이터를 문제에 입력

1. 대응방안
   1. 취약점 보안 대책
      1. 불충분한 전송계층 보호

[표 7] 불충분한 전송계층 보호 대응방안

|  |
| --- |
| ᆞSSL/TLS 전송 계층을 활용해 정보를 전송한다. (민감한 데이터, 세션 정보 또는 기타 주요 정보)  ᆞ웹뷰 또는 웹킷을 이용한 웹 앱의 경우에도 SSL을 사용한다.  ᆞ적절한 키 길이을 가진 강력한 암호화를 사용한다.  ᆞ신뢰할 수 있는 CA공급자가 서명한 인증서를 사용한다.  ᆞSSL/TLS를 적용했다면, 민감한 데이터를 다른 채널을 통해 전송하지 않는다. (SMS, MMS Push 등) |

* + 1. 안전하지 않은 데이터 저장

[표 8] 안전하지 않은 데이터 저장 대응방안

|  |
| --- |
| ᆞ계정(자격증명)을 파일 시스템 안에 저장해서는 안되며, 웹 또는 API 로그인 방식(HTTPS)을 통해 인증하고 세션 시간은 사용자의 사용 시간을 고려한 최소한의 시간으로 설정해야 한다.  ᆞ중요정보 저장 시 하드 코딩된 암호화 또는 암호화 해독 키에 의존하지 않아야 한다.  ᆞSQLite 데이터베이스를 사용할 경우 SQLCipher 모듈을 사용해 암호화해야 한다. |

* + 1. 의도치 않은 데이터 유출

[표 9]의도치 않은 데이터 유출 대응방안

|  |
| --- |
| ᆞ개발 시 아래 항목에 대해 민감한 정보 저장 여부를 확인하여, 주요 정보 저장 시 제거해야 한다.  -URL Caching(Both Request and Response)  -Keyboard Press Caching  -Copy/Paste buffer Caching  -Application backgrounding  -Logging  -HTML5 data storage  -Browser cookie objects  -Analytics data send to 3rd parties |

* + 1. 클라이언트 측 공격

[표 10] 클라이언트 측 공격 대응방안

|  |
| --- |
| ᆞ클라이언트 측 어플리케이션에서 사용자 입력 값에 대한 유효성 검사를 실시해야 하며, 입력 값에 따른 동작 로직에 따른 대응이 필요하다. |

* + 1. 신뢰할 수 없는 입력을 통한 보안 결정

[표 11] 신뢰할 수 없는 입력을 통한 보안 결정 대응방안

|  |
| --- |
| ᆞ민감한 의사 결정을 할 시, 사용자의 확인을 받는 과정이 필요하다.  ᆞ의사결정 기능 동작 시, 입력에 대한 유효성 검사 로직을 넣는다.  ᆞ민감한 정보 전달 등의 기능을 방지한다. 타 어플리케이션에서 접근 또는 기능 실행을 할 수 있는 부분을 고려해 개발한다. |

* + 1. 리버스 엔지니어링

[표 12] 리버스 엔지니어링 대응방안

|  |
| --- |
| ᆞ대부분의 앱은 리버스 엔지니어링의 영향을 받기 쉽지만 이 리스크를 완화 할지 여부를 고려할 때, 리버스 엔지니어링의 잠재적인 비즈니스 영향을 조 사하는 것이 중요하다.  ᆞ리버스 엔지니어링만으로 수행 할 수 있는 작업에 대한 간단한 샘플은 아래 예제를 참조하라. 좋은 난독화는 다음과 같은 사항을 포함한다:  -난독화 할 메소드/코드 세그먼트를 좁힌다.  -성능 영향의 균형을 맞추기 위해 난독화 정도를 조정한다.  -IDA Pro 및 Hopper와 같은 도구로 분석이 어려울 수 있다. 메소드뿐 만 아니라 문자열 테이블을 난독화 한다. |