Тема 14. Основні положення захисту для мобільних пристроїв

При розробці мобільного додатка слід враховувати, що дані, якими оперує ця програму, можуть представляти певний інтерес для третіх осіб. Ступінь цінності цих даних варіюється в широких межах, проте навіть найпростіша приватна інформація, наприклад, пароль входу в додаток, вимагає опрацювання її захисту. Особливо це важливо у світлі поширення мобільних додатків на всі сфери електронних послуг, включаючи фінансові, банківські операції, зберігання та передачу особистих даних тощо.

Основні види атак на мобільний додаток:

- Декомпіляція файлу програми (.ipa-файли для Apple iOS та .apk-файли для Google Android) та розбір локально збережених даних. Захист цього, найважливішого нині, рівня лежить на плечах мобільного розробника.
- Перехоплення даних, що передаються через мережу (MITM-атаки). Більшість мобільних додатків є клієнт-серверними, отже, постійно передають та приймають великі обсяги інформації. І хоча сучасна мобільна та веб-розробка активно завершують перехід на HTTPS-протокол спілкування, проте не варто покладатися на єдиний рубіж захисту у вигляді захищеного каналу зв'язку.
- Рутування пристрою та атака на додаток та алгоритми, що застосовуються в ньому, через зовнішні налагоджувальні інструменти.

Розглянемо вразливість загального характеру, без прив'язки до конкретної платформи. Тут і надалі використовується абревіатура КВД – критично важливі дані користувачів. До КВД належать будь-які дані, які не повинні бути доступні третій стороні, це стосується як персональних даних користувача (дата народження, адреса проживання, особисте листування), так і його приватних даних (паролі, дані кредитних карток, номери банківських рахунків, номери замовлень і т.д.).

Перелік основних вразливостей:

- Використання незахищених локальних сховищ.
  - Небезпека: Дуже висока.
  - Коментар: Зустрічається всюди, виражається у зберіганні КВД у незахищених або слабо захищених локальних сховищах, специфічних для конкретної платформи. Злам третьою стороною – елементарний, і, як правило, не потрібна наявність спеціальних навичок в атакуючого.
  - Захист: Зберігати КВД можна лише у захищених сховищах платформи.
- Зберігання КВС у коді.
  - Небезпека: Висока.
  - Коментар: Вразливість стосується зберігання КВД всередині коду (у статичних константних рядках, у ресурсах додатку тощо). Яскраві приклади: зберігання солі для пароля (password salt) у константі або макросі, яка застосовується по всьому коді для шифрування паролів; зберігання приватного ключа для асиметричних алгоритмів; зберігання паролів та логінів для серверних вузлів або баз даних. Легко розкривається третьою стороною за наявності базових навичок декомпіляції.
  - Захист: Не зберігати жодні КВД у коді або ресурсах програми.
- Застосування алгоритмів із зберіганням приватного ключа.
  - Небезпека: Висока.
  - Коментар: Вразливість є актуальною у випадку, якщо приватна інформація алгоритму (приватний ключ) вимушено зберігається в коді або ресурсах мобільного додатку (найчастіше так і буває). Легко розкривається шляхом декомпіляції.
  - Захист: У мобільній розробці бажано застосовувати тільки сучасні симетричні алгоритми з генерованим випадковим одноразовим ключем, що мають високу стійкість із злому методом грубої сили,

або виводити асиметричний приватний ключ за межі програми, або персоналізувати цей ключ (як приклад - приватним ключем може виступати користувач, збережений у зашифрованому вигляді у захищеному сховищі операційної системи).

- Використання асиметричного алгоритму з приватним ключем відомим серверу.
  - Небезпека: Залежить від ступеня безпеки сервера.
  - Коментар: Вразливість має подвійний характер. Зберігання приватного ключа дозволяє розшифровувати дані користувача на стороні сервера. По-перше, це некоректно з точки зору безпеки (якщо сервер буде зламаний - атакуючий також отримає доступ до приватних даних користувачів), а по-друге, це порушує приватність персональних даних. Користувач завжди має бути впевненим, що його персональна інформація не відома нікому, крім нього самого (тільки якщо він явно не дав дозвіл на її публікацію). Часто додатки позиціонують себе як захищені, але насправді такими не є, оскільки містять у собі засоби для розшифрування персональної інформації.
  - Захист: Без явної потреби та явного дозволу користувача (найчастіше через ліцензійну угоду) ні програма, ні сервер не повинні мати жодної можливості розшифрувати приватні дані користувача. Найпростіший приклад - пароль користувача повинен йти на сервер вже у вигляді хеша, і перевірятися повинен хеш, а не вихідний пароль (серверу абсолютно нема чого знати пароль користувача; якщо ж користувач його забув - для такої ситуації існує давно налагоджений механізм відновлення пароля, в тому числі з двофакторною авторизацією клієнта для підвищеної безпеки процедури відновлення).
- Використання самописних алгоритмів шифрування та захисту.
  - Небезпека: Середня.

- Коментар: Це пряме порушення принципу Керкгоффса. Виражається у спробі розробника винайти "свій особистий, не відомий нікому, а тому суперзахищений алгоритм шифрування". Будь-яке відхилення від існуючих, багаторазово перевірених та вивчених, математично доведених алгоритмів шифрування у 99% випадків обертається швидким зломом подібного "захисту". Потребує наявності середньо-високих навичок у атакуючого.
- Захист: Слід підбирати відповідний алгоритм лише з налагоджених та актуальних загальновідомих криптографічних алгоритмів.

– Передача КВС у зовнішнє середовище у відкритому вигляді.
  - Небезпека: Середня.
  - Коментар: Виражається в передачі КВД без застосування шифрування по будь-якому доступному каналу зв'язку із зовнішнім середовищем, будь то передача даних сторонньому додатку або передача в мережу. Може бути розкрито опосередковано шляхом розтину не додатка, яке сховища, або цільового додатка. Злам вимогливий до наявності навичок у атакуючого, за умови, що сховище є захищеним.
  - Захист: Будь-які КВД перед виходом за межі програми повинні бути зашифровані. Локальні сховища платформи не є областю програми, вони також повинні отримувати лише зашифровані дані.
– Ігнорування факту наявності рутованих чи заражених пристроїв.
  - Небезпека: Середня.
  - Коментар: Рутовані пристрої - це девайси, де виконана модифікація для отримання прав суперкористувача на будь-які операції, заборонені виробником операційної системи. Виконується користувачем на своєму пристрої самостійно і не обов'язково добровільно (клієнт може бути не в курсі, що пристрій зламано). Установка програми на рутованний девайс нівелює всі штатні засоби захисту операційної системи.

- Захист: Якщо це технічно можливо для платформи, то бажано забороняти роботу програми, якщо вдалося зрозуміти, що запуск здійснюється на рутованому пристрої, або хоча б попереджати про це користувача.

– Зберігання КВД у захищених сховищах, але у відкритому вигляді.
  - Небезпека: Середня.
  - Коментар: Розробники часто схильні зберігати КВД у захищені системні сховища без додаткового захисту, оскільки системні механізми добре опираються злому. Однак рівень їх стійкості падає до мінімуму у випадку, якщо пристрій рутоване.
  - Захист: КВД не повинні використовуватись у додатку без додаткового шифрування. Щойно потреба у "відкритих" КВС відпала — вони негайно мають бути або зашифровані, або знищені.
– Переклад частини функціоналу у вбудовані веб-движки.
  - Небезпека: Середня.
  - Коментар: Найчастіше виглядає як передача КВД у вбудований браузер, де завантажується зовнішня веб-сторінка, яка виконує свою частину функціоналу. Рівень захисту в цьому випадку різко знижується, особливо для рутованих пристроїв.
  - Захист: Не використовувати вбудований браузер та вбудований веб-рух в операціях з КВД. На крайній випадок – шифрувати КВД перед передачею.
– Реверсивна інженерія алгоритмів, що становлять інтелектуальну цінність.
  - Небезпека: Низька залежить від цінності алгоритму.
  - Коментар: Якщо розробки додатку всередині компанії використовуються деякі власні алгоритми, які можуть становити високу цінність потенційних конкурентів чи зломщиків, ці алгоритми мають бути захищені від стороннього доступу.
  - Захист: Автоматичний або ручний обфускація коду.

Є кілька загальних для всіх мобільних платформ моментів, яких слід дотримуватись при розробці.

Захист кодом користувача

Якщо програма захищена паролем користувача (PIN-кодом, сканом відбитка пальця, графічним паролем тощо), то при відході програми у фон ("згортання") вона повинна негайно відображати вікно введення цього захисного коду, перекриваючи собою весь екран програми. Це виключає можливість для зловмисника отримати приватну інформацію у разі крадіжки пристрою, поки програма все ще запущена та перебуває в сплячому режимі.

Будь-який код користувача повинен мати обмежену кількість спроб введення (наприклад, 5 разів), потім, у разі невдачі, програма повинна автоматично розлогінуватися (або взагалі блокуватися, залежить від конкретної програми).

В даний час при використанні цифрових кодів суворо рекомендується використовувати обмеження на довжину коду мінімум 6 цифр (більше можна, менше - не можна).

Функціонування клієнт-серверної програми

Для клієнт-серверних програм дуже корисно застосовувати сесійний механізм з обмеженим часом життя сесії. Це дозволить уникнути "простування" програми в незахищеному режимі, якщо користувач просто забув закрити його і залишив пристрій у вільному доступі. Слід враховувати, що термін дії сесії та її ідентифікатор відносяться до КВС, з усіма наслідками, що звідси випливають. Одним із вдалих прикладів реалізації подібного механізму є отримання абсолютного значення часу із сервера після проходження процедури авторизації користувача (дата та час мають показувати, коли саме сесія стане неактивною). Дату та час закінчення дії сесії не слід генерувати на пристрої, це знижує безпеку та гнучкість програми.

Клієнт-серверна програма не повинна здійснювати зміну КВД у локальному режимі. Будь-яка дія, яка потребує зміни КВД, має проходити синхронізацію із сервером. Виняток із цього правила становить лише

користувальницький код входу, що задається особисто користувачем та збережений у захищеному локальному сховищі.

Робота з датами

При оперуванні важливими додатку, на кшталт часу знищення сесії, не слід спиратися на відносний час. Тобто дані, що передаються з сервера, не повинні містити дату у вигляді "плюс N секунд/годин/днів від поточного моменту". В силу наявності потенційно високих затримок у передачі даних по мережі від мобільного додатка до сервера і назад, подібний спосіб синхронізації матиме надто велику похибку. Крім того, атакуючий (або просто несумлінний користувач) може просто змінити локальний пояс на пристрої, порушивши таким чином логіку роботи обмежувальних механізмів застосування. Завжди потрібно передавати абсолютне значення часу.

Абсолютні значення слід передавати із застосуванням універсальних способів обміну подібною інформацією, без прив'язки до часового поясу конкретного пристрою. Найчастіше оптимальним варіантом є поведінка програми, при якій дані відображаються користувачеві в його локальному часовому поясі, але їх зберігання та передача здійснюється у форматі, не прив'язаному до тайм-зони. Відповідними форматами для дат і часу є універсальний UNIX timestamp, збережений у змінній 64-бітного цілого знакового типу (UNIX timestamp — це кількість секунд, що пройшла з 1 січня 1970 року), або, на крайній випадок, рядок у повному форматі ISO-8601 із нульовою тайм-зоною. Переважний саме UNIX timestamp, він дозволяє уникнути потенційних помилок та проблем з конвертацією рядків у дату та назад на різних мобільних платформах.

Додаткові рекомендації

Додаток не повинен відображати приватну користувальницьку інформацію великими, яскравими, шрифтами, що добре читаються, без явної на те необхідності і без окремого запиту користувача, щоб виключити можливість читання цих даних здалеку з екрана пристрою.

Не варто сліпо довіряти бібліотекам з відкритим вихідним кодом, які пропонують певний захист приватним даним користувачів. Виняток становлять бібліотеки, перевірені часом і які у великих проектах корпорацій (наприклад, вбудоване шифрування у відкритому движку бази даних Realm). Штатних механізмів захисту операційної системи та загальнодоступних перевірених криптографічних алгоритмів у переважній більшості випадків буде більш ніж достатньо.

Абсолютно неприпустимо використовувати криптографічні бібліотеки із закритим вихідним кодом (навіть якщо вони платні). У таких рішеннях ви не зможете перевірити, наскільки ефективна дана бібліотека, а також наскільки "чесний" у неї захист (чи немає там механізму backdoor, чи не надсилаються "захищені" дані якійсь третій стороні).

У релізних збірках додатків має бути відключено логування даних у системну консоль та незахищені файли. Специфічні логи для розробників можуть бути присутніми, але бажано в зашифрованому вигляді, щоб уникнути доступу третіх осіб до закритої службової інформації, яка може бути присутня в логах.