Le app colluse aggirano le misure di sicurezza applicato da sistemi operativi sandbox come Android. Ricerca nel rilevamento e nella protezione dalle app la collusione richiede una varietà di app colluse per la sperimentazione. Un sistema chiamato ApplicationCollusion Engine (ACE) per generare automaticamente combinazioni di collusione e non collusione. I moderni sistemi operativi mobili, come Android, utilizzano

sandboxing per evitare che app dannose causino danni effetti limitando l'accesso alle risorse da parte di ogni processo al di fuori del suo dominio In un ambiente sandbox, accedi a le risorse di sistema sensibili sono mediate dal sistema operativo e limitato per impostazione predefinita. Le app che richiedono l'accesso devono richiedere le autorizzazioni necessarie all'utente. Le app colluse utilizzano canali nascosti e palesi insieme eseguire operazioni dannose. Lo sviluppatore incurante può esporre involontariamente il permesso protetto alle risorse consentendo al componente di accedere a quelle risorse per comunicare con altre app tramite IAC (comunicazione tra le app). Un attaccante può trarne vantaggio di questo componente per accedere alla risorsa protetta senza dover richiedere il permesso all’utente. Le app colluse possono fornire informazioni furto, ma può anche essere utilizzato per abusare di un servizio di dispositivo o aumentare l'impatto di un attacco all'interno di un sistema. Il principale obiettivo degli attacchi di collusione è evitare le restrizioni imposte da ambienti sandbox, come Android, per effettuare un attacco più difficile da rilevare. I ricercatori sono spesso concentrati sugli accessi alle risorse che si danno ad una applicazione per stabilire il livello di rischio. Non è necessario richiedere l'accesso a una risorsa protetta. Lo farà basta chiederlo a un'app collusa che ha già accedervi. Va notato che, mentre Android richiede all'utente di concedere le autorizzazioni per essere utilizzato da un'app, esso non impone alcun controllo sul modo in cui le app scambiano informazioni.

le tecniche per rilevare malware Android sono prontamente disponibili [17,30]. Questi possono essere classificati in due gruppi principali: statici e dinamico. Nell'analisi statica, alcune funzionalità del file binario dell'app vengono estratti e analizzati utilizzando diversi approcci come come tecniche di apprendimento automatico. L'analisi dinamica rileva il malware in fase di esecuzione. Distribuisce monitor adeguati sui sistemi Android per registrare tracce e funzionalità utilizzate per cercare dannosi comportamenti. A differenza del rilevamento di malware, rilevamento di app colluse implica non solo l'ottenimento di funzionalità che mostrano se un'app svolge una minaccia alla sicurezza, ma rivela anche se la comunicazione tra più app si verifica durante l'attacco.

Le altre app colluse disponibili in letteratura possono essere classificati in due gruppi: quelli sviluppati per testare il rilevamento e metodi di protezione e quelli sviluppati per esplorare il diversi covert channel disponibili su Android. Covert canali in Android, come in altri sistemi, sono limitati dalla quantità di risorse condivise, side channel che possono essere torvati.

Le app colluse possono eseguire qualsiasi attacco simile a quelli posti dalle singole app affette da malware. (Dos, furto di informazioni….). Sebbene la natura di ciascuno di questi attacchi sia diversa, essi sono tutti basati sull'esecuzione di una serie di azioni in un ordine specifico. Nel nostro caso, ogni azione definita viene implementata con un file frammento di codice specifico. In questo modo, creando un'app collusa impostato per eseguire una minaccia richiede di dividere, tra diverse app, i diversi frammenti di codice (azioni) necessari per eseguirlo. Il canale di comunicazione utilizzato per eseguire l'attacco è non rilevante per l'obiettivo dell'attacco. Finché il codice selezionato gli snippet consentono alle app di scambiare informazioni in qualche modo, il file il set generato sarà in grado di colludere. Questo fatto ci permette di farlo creare molti set di app collusi diversi anche per lo stesso minaccia scambiando diversi frammenti di codice di comunicazione. Gli snippet di codice sono piccole porzioni di codice che eseguono uno specifico funzione. Questi vengono utilizzati per popolare i componenti dell'app nei punti di iniezione del codice. I file di descrizione dell'app specificano il modo in cui gli snippet di codice vengono iniettati nel componente.   
Il Colluding Set Engine ha il compito di strutturarli in modo tale che le app risultanti siano in grado di comunicare e colludere.

I file di descrizione dell'app specificano quali componenti dell'app verranno utilizzati all'interno di un'app e come saranno i loro punti di iniezione.

Immagine che contiene tavolo

Descrizione generata automaticamenteUn esempio di questo processo per una collusione set di app che estrae l'elenco dei contatti dal dispositivo dell'utente. Il set è composto da tre app. La prima app legge i contatti e li invia alla seconda app tramite un intent. Il secondo app inoltra le informazioni alla terza app utilizzando external storage. La terza app del set invia le informazioni a un file server in remoto. Tutte le informazioni lette dagli snippet vengono concatenate in una stringa e quindi inviate all'altra app per l'estrazione. Due diversi canali di comunicazioni utilizzati dalle app sono:

INTENT : un'app avvia un intent con un'estensione azione generata casualmente. L'app ricevente registra un broadcast receiver con la stessa azione. Altri canali di comunicazione basati su intent (ad es. intenti espliciti lancio di acticity e servizi) potrebbero essere facilmente aggiunti incorporando i frammenti di codice corrispondenti.

SHARED PREFERENCES: Sebbene non siano intese per inter-app comunicazione, le app possono utilizzare coppie chiave-valore per lo scambio informazioni se vengono definiti i flag appropriati (WORLD\_READABLE o WORLD\_WRITABLE) durante l'accesso e la memorizzazione dei dati. Nel i nostri set, una delle app salva i dati in un mondo leggibile file delle preferenze condivise. L'app ricevente accede lo stesso file per leggere le informazioni. Informazioni inviate a un server web esterno (controllato da noi) è stato registrato sul server e sul dispositivo. Per assicurati che la collusione sia realmente avvenuta abbiamo verificato il file log del dispositivo e le richieste HTTP POST ricevute da server.

Per fare la detection di questo tipo di attacco che utilizza covert channel, è possibile combinare due app in modo tale che i componenti che comunicano sono ora all'interno della stessa app). In questo modo, strumenti come FlowDroid o Amandroid potranno tracciare direttamente le informazioni sensibili che fluiscono attraverso le due componenti (che ora sono insieme nella stessa app). Tuttavia, l'app risultante è ancora valida ai fini dell'analisi statica e della trasformazione dei canali di comunicazione tra le app in canali di comunicazione tra i componenti all'interno della stessa app combinata, anche se potrebbe non funzionare correttamente l’istallazione su un dispositivo. APKCombiner unisce tutti i componenti dell'app (attività, services, ecc.) in un unico APK, risolvendo i conflitti di denominazionee unendo entrambi i file manifest dell'app.