Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, Eric Bodden

Harvester

Vollautomatische Extraktion von Laufzeitwerten aus obfuskierten Android-Applikationen

Mit Harvester können Sicherheitsexperten und Entwickler automatisch Laufzeitwerte aus Android-Apps extrahieren, selbst wenn diese nur verschlüsselt im Code vorliegen. Diese Werte zeigen, wie Apps mit Internetdiensten interagieren und ermöglichen, Datendiebstahl und Schwachstellen frühzeitig zu erkennen. Harvester verbessert hierdurch den Privatsphärenschutz von Milliarden von Endnutzern.

Siegfried Rasthofer

Wissenschaftlicher Mitarbeiter am Fraunhofer Institut für Sichere Informationstechnologie SIT und TU Darmstadt

E-Mail: siegfried.rasthofer@sit.fraunhofer.de

Steven Arzt

Wissenschaftlicher Mitarbeiter am Fraunhofer Institut für Sichere Informationstechnologie SIT und TU Darmstadt

E-Mail: steven.arzt@sit.fraunhofer.de

Marc Miltenberger

Wissenschaftliche Hilfskraft am Fraunhofer Institut für Sichere Informationstechnologie SIT

E-Mail: marc.miltenberger@sit.fraunhofer.de

Prof. Dr. Eric Bodden

Professor für Software Engineering am Heinz Nixdorf Institut der Universität Paderborn und Direktor für Software Engineering am Fraunhofer IEM

E-Mail: eric.bodden@uni-paderborn.de

1 Laufzeitwerte verraten, wie Apps mit der Außenwelt kommunizieren

Beim Thema Internetsicherheit denken die meisten Menschen zunächst an Verschlüsselung oder die Absicherung von Verbindungen. Zahlreiche Studien belegen jedoch, dass seit Jahren die allermeisten Cyber-Angriffe nicht über die Netze als solche, sondern über die Applikationsschicht erfolgen. Insbesondere nutzen immer mehr Anwender Internetdienste nicht mehr über klassische Webportale, sondern über Smartphone-Apps. Der konkrete Webdienst, der von der App genutzt wird, um z. B. Banktransaktionen durchzuführen, tritt immer mehr in den Hintergrund.

Insofern verwundert es nicht, dass viele Endnutzer sich um die Sicherheit und Vertrauenswürdigkeit der von ihnen auf dem Gerät eingesetzten Apps sorgen. Ein Endnutzer kann die Vertrauenswürdigkeit von Apps jedoch im Allgemeinen nicht selbst beurteilen, sondern muss sich auf das Urteil von Experten verlassen. Solche Experten arbeiten z. B. bei den Betreibern von App Stores, den Anbietern von Antiviren-Lösungen, oder teils auch im eigenen Unternehmen, sofern dies über eine eigene IT-Sicherheitsabteilung verfügt. Auch für diese Experten ist die Einschätzung einer gegebenen App jedoch oftmals schwierig und zeitaufwendig. Manche Einrichtungen bekommen mehrere tausende Apps pro Tag zur Prüfung übergeben.

Der meist wichtigste Bestandteil der Prüfung einer App auf Schwachstellen oder Schadcode ist die Ermittlung und Bewertung von Laufzeitwerten. Eine Banking-App, welche mit dem korrekten Zielserver der vorgesehenen Bank kommuniziert, ist anders zu bewerten als eine vorgebliche Banking-App, die Kontodaten an unbekannte Serveradressen versendet. Eine unverschlüsselte HTTP-Verbindung hingegen kann ein wichtiger Hinweis auf eine mögliche Schwachstelle sein. Es ist also für die Einschätzung der Vertrauenswürdigkeit einer App nahezu unabdingbar, bestimmte Laufzeitwerte der App (aufgerufene URLs, gesendete Daten etc.) zu kennen.

Die zu untersuchenden Apps liegen in der Regel nur als Bytecode vor. Sicherheitsanalysten können mit bisherigen Verfahren sowohl statische als auch dynamische Verfahren anwenden, um Laufzeitwerte automatisiert aus dem Bytecode zu extrahieren. Statische Analyseverfahren inspizieren den Code direkt, ohne ihn auszufüh-

ren. Sie sind daher darauf angewiesen, die entsprechenden Laufzeitwerte im Bytecode eingebettet vorzufinden. Konkrete Laufzeitwerte wie z. B. die genaue Adresse eines entfernten Servers sind oftmals jedoch nicht als lesbarer Text in der App hinterlegt, sondern verschleiert (in Abschnitt 2 geben wir ein Beispiel). Die zur Verschleierung eingesetzten Verfahren werden hierbei stetig fortschrittlicher. Beschränkten sich Schadcode-Autoren vor einigen Jahren noch darauf, Methoden-, Variablen-, Klassennamen etc. im Programmcode umzubenennen, so verschleiern aktuelle Verfahren wichtige Aufrufe mittels Reflection oder laden Programmcode dynamisch nach. Automatisierte Werkzeuge, sogenannte Packer, übernehmen diese Aufgabe dabei vollautomatisch.

Nicht nur Schadcode, sondern auch gutartige Apps nutzen solche Packer, beispielsweise um Geschäftsgeheimnisse zu schützen, oder das Ausnutzen potentieller Sicherheitslücken zu erschweren. Rein statische Analyseverfahren sind derzeit nicht in der Lage, aus aufwendig verschleierten Schadcode-Apps die benötigten Laufzeitwerte zu extrahieren, und werden dies auch in Zukunft nicht leisten können, da sich Code immer auf eine Art verschleiern lässt, die das jeweilige Analysewerkzeug nicht unterstützt.

Um eine solche App dennoch bewerten zu können, kann der Sicherheitsanalyst daher versuchen, das konkrete Verhalten der App zur Laufzeit zu beobachten. Dies ist das Ziel dynamischer Analyseverfahren. Sie führen die jeweilige App in einer geschützten Umgebung aus und versuchen, die entsprechenden Laufzeitwerte an den passenden Kommunikationsschnittstellen abzugreifen. Viele App-Entwickler, besonders solche von Schadcode, benutzen jedoch neben den oben genannten statischen Verschleierungstechniken auch noch sogenannte Anti-Analysetechniken wie „Timing Bombs" oder „Logic Bombs", die das Ziel haben, eine dynamische Analyse der App zu unterbinden. So wird in vielen Apps schadhaftes Verhalten beispielsweise erst nach einem Neustart des Geräts aktiv. Während dies auf einem realen Mobiltelefon lediglich zu einer verzögerten Ausführung des Schadcodes führt, bedeutet es gleichsam, dass eine dynamische Analyseumgebung einen solchen Neustart simulieren muss, um das schadhafte Verhalten zu stimulieren und beobachten zu können.

Andere Applikationen warten mit der Ausführung des Schadcodes auf das Betätigen bestimmter Elemente in der Benutzerschnittstelle. Ein dynamisches Analysewerkzeug muss, wenn es das schadhafte Verhalten auslösen möchte, zur Laufzeit diese Elemente betätigen. Hierfür gibt es jedoch im Allgemeinen unendlich viele mögliche Sequenzen. Dieses sogenannte Pfadabdeckungs-Problem stellt die Wissenschaft schon seit Jahren vor große Herausforderungen. Selbst aktuelle Werkzeuge erreichen in der Regel nur ca. die Hälfte aller Programmstatements. Malware-Autoren können sich daher derzeit hinter den Unzulänglichkeiten dieser Ansätze gut verstecken.

Wieder andere Schadcode-Apps implementieren so genannte Command-and-Control-Bots. Hierbei wird der Schadcode erst aktiv, wenn die „Bot-App" ein entsprechendes Startkommando empfängt, z. B. per E-Mail, SMS oder durch das Setzen eines Flags auf einem Webserver. Um den Schadcode observieren zu können, muss ein dynamisches Analysewerkzeug nun genau jenes Kommando an die jeweilige App senden. Solche Fuzzing-Ansätze haben ihre eigenen Limitierungen. Black-Box Fuzzing skaliert generell schlecht. White-Box oder Grey-Box Fuzzing hingegen basiert auf statischer Analyse, die durch die o. g. Verschleierung ausgehebelt wird.

Es lässt sich also zusammenfassen, dass aufgrund der Obfuskations- und Anti-Analyse-Techniken, die in aktueller Schadsoftware eingesetzt werden, die momentanen Ansätze der statischen und dynamischen Analyse allesamt an ihre Grenzen stoßen. Im Ergebnis bleiben derzeit tausende Apps unzureichend geprüft und Milliarden von Endnutzern unzureichend geschützt. Somit ist die Internetsicherheit für mobile Geräte nicht gewährleistet. Da die Internetnutzung häufiger über mobile Geräte erfolgt als über den PC [1], stellt dies ein großes Problem dar.

2 Harvester extrahiert Laufzeitwerte

Listing 1 zeigt beispielhaft anhand von Java-Code, wie Obfuskations- und Anti-Analyse-Techniken aktuelle statische und dynamische Codeanalysen unbrauchbar machen. Das Codebeispiel ist aus einer aktuellen Android-Malware entnommen, wurde allerdings vereinfacht, um die Funktionsweise von Harvester beschreiben zu können.

Listing 1 | Vereinfachtes Code-Beispiel für Obfuskations- und Anti-Analyse-Techniken

```
1  public void onBootCompleted() {
2    if(isEmulator()) return;
3    String information = null;
4    if(simCountryIso().equals("US"))
5      information = decryptInformation1();
6    else
7      information = decryptInformation2();
8    String url = decrypt("iv498aIlPnRQccpRR");
9    doSomethingElse();
10   executePostRequest(url, information);
11 }
```

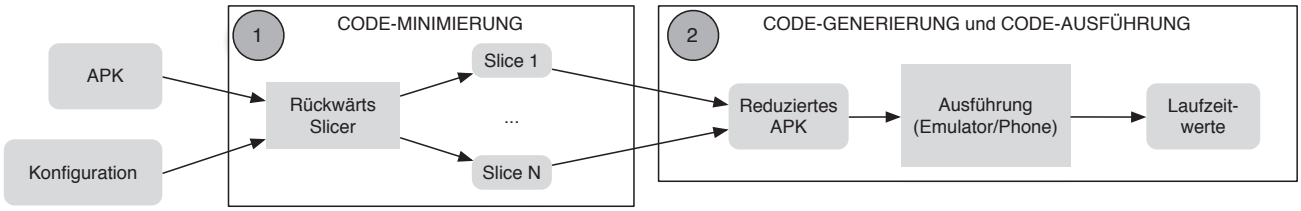
Zeile 10 zeigt eine Anweisung, in der Daten per HTTP Post an einen entfernten Server übertragen werden. Als App-Analyst möchte man wissen, welche Daten wohin gesendet werden. Nur mit diesen Informationen lässt sich einschätzen, ob eine sichere und gewollte Nutzung eines Webdiensts vorliegt (bspw. per HTTP oder HTTPS), oder ob unautorisiert persönliche Daten des Benutzers versandt werden. Die Adresse des Zielservers, an den die Daten gesendet werden, wird jedoch erst zur Laufzeit entschlüsselt (siehe Zeile 8).

Das Codebeispiel zeigt außerdem, dass die übermittelten Informationen abhängig vom Land sind (USA oder nicht, Zeile 4). Auch diese Daten liegen im Quelltext nicht im Klartext vor, sondern werden erst zur Laufzeit entschlüsselt. Diese Verschlüsselung kann beliebig komplex sein und von Fall zu Fall variieren; sie ist daher weder für den Analysten noch für ein statisches Analysewerkzeug leicht umkehrbar.

Warum also nicht eine dynamische Programmanalyse benutzen, welche den Programmcode während seiner Ausführung beobachtet? Leider wird durch das hier gegebene Codebeispiel auch eine dynamische Analyse effektiv verhindert. Zeile 1 birgt die Signatur einer sogenannten Callback-Methode, welche nur ausgeführt wird, wenn das Smartphone komplett neu gestartet wurde. Dies zeigt eine sehr gängige Methode in aktueller Android-Malware, um das schadhafte Verhalten nicht direkt auszuführen, sondern erst nach bestimmten Ereignissen. Das Prinzip kann durch beliebig komplexe Ereignis-Sequenzen erweitert werden, z. B. 24 Stunden Wartezeit, Button-Klicks, Antworten vom Command-And-Control-Server etc.

Zeile 2 enthält einen so genannten Emulator-Check: Die App selbst prüft, ob sie gerade in einer emulierten Umgebung, also

Abbildung 1 | Ablaufdiagram von Harvester



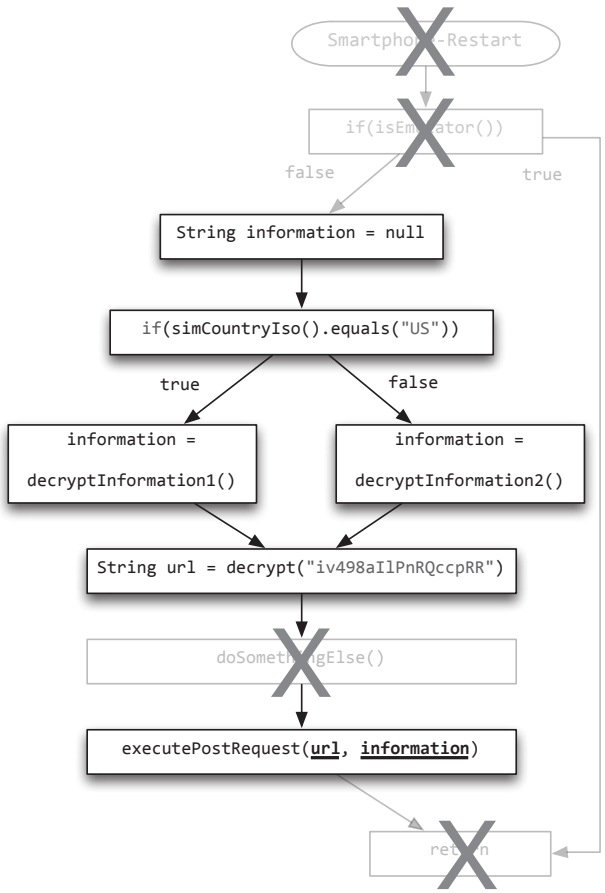
beispielsweise in einem Analysewerkzeug, ausgeführt wird. Sollte dies der Fall sein, werden keine Daten versendet. Der Datenversand erfolgt nur, wenn die App auf einem realen Smartphone ausgeführt wird. Um diesen beobachten zu können, müsste ein Analyst sämtliche Überprüfungen erkennen und dann den entsprechenden Systemzustand simulieren – ein sehr aufwändiger Prozess. Selbst wenn der Analyst diese Überprüfungen durch geschickte Konfiguration seiner Analyseumgebung umgehen kann, erhält er kein vollständiges Bild des Verhaltens der App. Abhängig vom Land (USA oder nicht) wird jeweils nur ein Datensatz versendet. Möchte der Analyst also alle möglichen Werte extrahieren, so muss er alle möglichen Ausführungsvarianten identifizieren (in diesem Fall nur zwei, sonst häufig mehr) und entsprechend viele Smartphones oder Analyseumgebungen verwenden – gerade bei größeren Apps mit einer großen Anzahl an Ausführungsvarianten ein sehr teures und langwieriges Unterfangen.

Die Hauptfunktion des Harvester-Werkzeugs ist es, die für die Sicherheitseinstufung einer App entscheidungsrelevanten Laufzeitwerte effektiv und vollautomatisiert zu extrahieren [2]. Dies ist auch dann möglich, wenn die App Obfuskations- und Anti-Analyse-Techniken einsetzt. Nehmen wir an, wir wollen im gegebenen Codebeispiel sowohl die URL als auch Informationen über die versendeten Daten erhalten.

Die grundlegende Idee von Harvester besteht nun darin, aus einer App nur genau den Programmcode auszuschneiden, der diese beiden Werte berechnet. Genau dieser Code wird dann in Isolation ausgeführt. Etwaige Wartezeiten oder Überprüfungen, ob die App in einer Analyseumgebung ausgeführt wird, verändern den Variablenwert im Normalfall nicht und werden daher entfernt. Somit werden solche Obfuskationen effektiv umgangen. Der Harvester-Algorithmus führt hierzu folgende drei Schritte durch, und zwar vollautomatisch:

- **Code-Minimierung:** Abbildung 2 zeigt den Kontrollflussgraphen für das Code-Beispiel von Listing 1. Der Graph verdeutlicht, dass zur Bestimmung der Laufzeitwerte für *url* und *information* in Zeile 10 nicht alle Code-Statements relevant sind. Dies ist durch ein Kreuz gekennzeichnet. Harvester nutzt eine spezielle Form des sogenannten statischen Slicings [3], um solche Mengen von Statements zu identifizieren, die für die Bestimmung bestimmter Laufzeitwerte notwendig sind, sogenannte Slices. Im Beispiel bestimmt Harvester beispielsweise all diejenigen Programmstatements, die die Werte von *url* und *information* beeinflussen. In Abbildung 2 sind diese Statements hervorgehoben, alle anderen Statements sind ausgegraut. Durch eine empirische Studie mit aktueller Android-Malware, unterstützt durch McAfee, konnten wir belegen, dass die Anzahl der Programmstatements, die für die Berechnung solcher Werte notwendig ist, normalerweise sehr klein ist. Nachdem Harvester ein Slice bestimmt hat, wird dieses aus dem Code extrahiert.

Abbildung 2 | Grafische Darstellung des Code-Beispiels



- **Code-Generierung:** Aus dem im ersten Schritt ermittelten Slice wird nun automatisch eine neue App generiert, die den Slice-Code direkt ausführt. Dadurch, dass nur der extrahierte Code ausgeführt wird, wird im Beispiel insbesondere die entschlüsselte Zieladresse des Servers sofort berechnet, auch wenn dies in der ursprünglichen App erst nach einer Wartezeit oder einem Neustart des Smartphones geschehen wäre. Hängt der berechnete Wert von der Ausführungsumgebung ab (z. B. dem Land des Benutzers wie im Beispiel), wird der Slice parametrisiert, d. h., es werden alle möglichen Ergebnisse solcher Umgebungsprüfungen aufgezählt. Auf Codeebene werden solche Bedingungen durch eine von Harvester steuerbare Variable ersetzt, die einmal auf *true* und einmal auf *false* gesetzt wird. Harvester fügt dem generierten Code außerdem Logging-Anweisungen hinzu, welche die zu extrahierenden Werte an den richtigen Programmstellen ausgeben.

- **Code-Ausführung:** Im dritten Schritt werden die neu generierten Methoden nacheinander ausgeführt. Da Harvester ausschließlich die relevanten Code-Teile der App ausführt, reduziert Harvester ein in vielen Fällen eigentlich aussichtslos komplexes dynamisches Analyseproblem auf eine handhabbare Größe. In den meisten Fällen ist dadurch eine vollständige Extraktion der für Sicherheitsexperten relevanten Laufzeitwerte einer App in unter einer Minute möglich.

Abbildung 1 zeigt den vollständigen Ablauf von Harvester. Als Eingabe erwartet Harvester eine oder mehrere binäre Apps (APK) und eine Konfigurationsdatei. In der Konfigurationsdatei wird definiert, welche Arten von Laufzeitwerten extrahiert werden sollen (z. B. URLs). Dann werden für jede App die Schritte 1 und 2 in Abbildung 1 durchgeführt, welche bereits oben beschrieben wurden. Die Ausgabe von Harvester sind Laufzeitwerte an den in der Konfigurationsdatei bestimmten Programmpositionen (siehe 2). Weitere konkrete Details zu dem hier vorgestellten Ansatz können der Konferenzpublikation der Autoren entnommen werden [4].

3 Anwendungsgebiete

Exemplarisch betrachten wir nun verschiedene Nutzergruppen und Anwendungsgebiete von Harvester.

3.1 Private Endnutzer

Apps übertragen oftmals private Daten der Nutzer über das Internet. Harvester kann Auskunft darüber geben, mit welchen Internetdiensten eine App wie genau kommuniziert, bevor der Nutzer diese installiert. Manche Benutzer möchten beispielsweise nicht, dass ihre Daten außerhalb Deutschlands oder der EU verarbeitet werden. Ob dies der Fall ist, lässt sich für den Endnutzer jedoch normalerweise kaum feststellen. Selbst wenn der Entwickler der App entsprechende Angaben macht, muss der Benutzer diesen bislang blind vertrauen und kann sie nicht prüfen.

Harvester hingegen ermöglicht dem Nutzer, die verwendeten URLs und Servernamen aus der App zu extrahieren. Diese lassen sich leicht mittels Geo-IP-Diensten, die kostenlos im Internet verfügbar sind, auf physikalische Standorte abbilden. Des Weiteren können durch zusätzliche Analysen, wie einer durch Laufzeitwerte verbesserten Datenflussanalyse, dem Benutzer gezielt Informationen über Datenleaks in Applikationen aufgezeigt werden. Dies ist insbesondere in Bereichen, in denen rein rechtliche Vorgaben nicht hinreichend oder effektiv sind, ein wichtiges Hilfsmittel. Gerade private App-Entwickler und Kleinfirmen veröffentlichen ihre Apps international auf den verschiedenen App Stores, ohne die Datenschutzvorschriften und Informationsvorgaben sämtlicher Länder zu kennen. Der Benutzer kann sich daher z. B. nicht auf die Einhaltung europäischer Datenschutzstandards verlassen. Eine eigene Kontrollmöglichkeit ist daher anzuraten.

3.2 App-Entwickler

Kontrollmöglichkeiten im Bereich des Datenschutzes sind nicht nur für Endnutzer wichtig, sondern auch für die App-Entwickler selbst. Diese greifen oftmals auf Softwarebibliotheken von Drittanbietern zurück (z. B. für die Anzeige von Werbung oder das Speichern von Daten), deren genaue Funktionalität sie nicht kennen. Hierdurch kann der Entwickler unbewusst die Privatsphäre der Nutzer gefährden oder im schlimmsten Fall sogar ungewollt Schadcode in seine App integrieren. Mit Harvester kann der Entwickler diese Bibliotheken vor der Einbindung in seine App prüfen und bewerten.

3.3 App Store-Betreiber

Vor kurzem haben Amazon und Google begonnen, Schwachstellen-Scans in ihre App Stores zu integrieren. Bevor eine Applikation in den jeweiligen App Store aufgenommen wird, wird diese zunächst auf Sicherheitslücken untersucht. Sollte die Applikation eine Schwachstelle beinhalten, wird der Entwickler aufgefordert, diese zu beheben, bevor sie in den App Store aufgenommen wird. Beispiele hierfür sind falsche oder schwache kryptographische Implementierungen, eingebettete Authentifizierungsdaten (z. B. Server-Passwörter) oder falsche Autorisierungsrechte für Dateizugriffe. Für alle diese Beispiele sind konkrete Laufzeitwerte notwendig.

In einer aktuellen Studie konnten wir zeigen, dass viele App-Entwickler Zugangsdaten für die Serverkommunikation fest in ihre Apps integrieren. Die meisten der Entwickler haben das Sicherheitsprinzip einer geschützten Server-Komponente nicht verstanden und gingen davon aus, dass eine Obfuskation der Zugangsdaten in der App ausreichend Schutz bietet. Wie bereits erwähnt sind Obfuskationstechniken in der Regel jedoch kein Problem für Harvester, und wir konnten zeigen, dass mehr als 56 Millionen sensitive Datensätze (E-Mail-Adressen, Gesundheitsdaten, Daten zu Verkehrsunfällen etc.) offen zugänglich im Internet verfügbar waren. Wir informierten die App Store-Betreiber Google und Amazon über unsere mit Harvester erzielten Ergebnisse. Die in den betreffenden App Stores nun integrierten Schwachstellenscans sind u. a. auch ein Resultat dieser Forschung. IT-Sicherheitsabteilungen von Unternehmen stehen vor ähnlichen Herausforderungen, wenn diese eigene interne App Stores mit für die Verwendung auf Dienstgeräten zugelassenen Apps betreiben oder entsprechende Mobile Device Management-Lösungen konfigurieren.

3.4 Antivirus-Anbieter

Virenscanner auf Kundensystemen funktionieren, indem sie Apps gegen eine „schwarze Liste" von Signaturen abgleichen, um herauszufinden, ob eine App Schadcode enthält und daher nicht installiert werden sollte. Hierfür müssen die Sicherheitsexperten des Herstellers des Antivirus-Scanners täglich mehrere tausend neue Apps beurteilen und im Falle von Schadcode entsprechende Signaturen erstellen.

Viele Apps kommunizieren als normaler Funktionsbestandteil mit dem Internet. Um zu ermitteln, ob diese Kommunikation erwünscht ist oder einen Datenschutzverstoß darstellt, muss man daher ihre Endpunkte kennen, z. B. die URL des entfernten Servers. Diese Laufzeitwerte können mit Harvester extrahiert werden. Sendet eine App bspw. Daten an einen bekannten Server auf einer schwarzen Liste, kann die App automatisiert sofort als Schadsoftware aussortiert werden. Selbst bei nicht eindeutigen Fällen, die weiterhin eine manuelle Prüfung erfordern, sind die von Harvester extrahierten Daten eine wichtige Hilfestellung für den menschlichen Experten.

3.5 Strafverfolgungsbehörden

Um Cyberkriminelle für Datendiebstähle oder Betrugsdelikte strafrechtlich verfolgen zu können, ist es von zentraler Bedeutung, Hinweise auf die Identität der Kriminellen zu erhalten. Hierbei kann Harvester die Mitarbeiter von Strafverfolgungsbehörden unterstützen. Extrahierte Laufzeitwerte wie z. B. IP-Adressen oder URLs von Servern, mit denen kommuniziert wird, können als erster Anhaltspunkt dienen, um dann bspw. zu ermitteln, welche Person die entsprechenden Domains registriert hat. Ein Banken-Trojaner, den wir in Zusammenarbeit mit McAfee analysiert haben und der in Südkorea über 20.000 Geräte infizierte, nutzte unter anderem E-Mail als Kommunikationsmedium. Hier waren wir mit Hilfe von Harvester in der Lage, die E-Mail-Adresse aus dem Schadcode zu extrahieren, obwohl sie nur verschleiert vorlag. Sobald die Adresse bekannt ist, kann ein entsprechendes Auskunftsersuchen an den E-Mail-Provider gerichtet werden.

Damit der Schadcode das E-Mail-Konto nutzen kann, muss zudem das zugehörige Passwort darin enthalten sein, welches sich auch mit Harvester extrahieren lässt. Dies gibt Strafverfolgern Zugang zum entsprechenden Account, auch wenn der E-Mail-Provider z. B. aus dem Ausland heraus agiert und dort aufgrund fehlender Rechtshilfevereinbarungen nicht zur Kooperation verpflichtet werden kann. In diesem Fall kann der E-Mail-Account dennoch überwacht werden, um weitere Informationen über die Urheber des Angriffs zu erfahren oder neue Angriffswellen frühzeitig zu erkennen. Beim o. g. Banken-Trojaner war dies der Fall. Auch kann die E-Mail-Adresse von Entwicklern von Schutzsoftware auf eine Sperrliste aufgenommen werden, um weiteren Datendiebstahl zu vermeiden. Eine Sicherheits-App kann beispielsweise auf dem jeweiligen Gerät sämtliche ausgehenden E-Mails an diese Adresse blockieren. Auch eine Kooperation mit Internetdiensteanbietern, E-Mails an diese Adresse nicht mehr über ihre Server weiterzuleiten, ist denkbar.

Viele schädliche Anwendungen handeln nicht autark, sondern benötigen eine Verbindung zu einem Kontrollserver. Diese Apps werden von Kriminellen über spezielle Befehle ferngesteuert, um z. B. erst zu einem bestimmten Zeitpunkt aktiv zu werden. Das Ziel der Angreifer ist es, möglichst viel Schadcode zu verteilen. Dabei soll die App nicht vorzeitig durch ihr Verhalten auffällig werden. Erst wenn eine hinreichende Infizierungsquote erreicht ist, werden alle diese Installationen gleichzeitig aktiviert, um bis zur Entdeckung möglichst viele Einnahmen erzielen zu können. Extrahiert ein Analyst die entsprechenden Steuerbefehle aus der App, was mit Harvester möglich ist, können sie durch Sicherheitssoftware blockiert werden, sodass sie ihr Ziel nicht mehr erreichen. In den meisten Ländern ist eine Strafverfolgung zudem erst ab diesem Aktivierungszeitpunkt möglich, da vorher formal keine Straftat begangen wurde. Ist der Befehl bekannt, kann er aber direkt als Auslöser für ein Verfahren dienen, sobald er versendet wurde, selbst wenn er blockiert wird. Einen konkreten Fall bearbeiteten wir gemeinsam mit McAfee. Hierbei haben wir erfolgreich mögliche Command-and-Control-Kommunikations-Befehle und Zielserver aus der Applikation mit Hilfe von Harvester extrahiert und warten nun, bis wir diese Befehle in einer Netzwerkverbindung sehen. Im Anschluss werden die maliziösen Server durch die Behörden abgeschaltet.

Neben klassischer Internetkriminalität werden Apps auch zunehmend zur Unterstützung von Terrorismus genutzt. Um unerkannt kommunizieren zu können, entwickelten Anhänger des sogenannten Islamischen Staats nach Medienberichten eine eigene App zum verschlüsselten Austausch von Nachrichten. Ziel ist offenbar, potentiellen Hintertüren in bestehenden, populären Messenger-Apps oder entsprechenden Zugriffsmöglichkeiten der Strafverfolger auf die jeweiligen Backends zu entgehen [5]. Natürlich benötigen auch solche Untergrund-Apps eine Infrastruktur im Internet, um Nachrichten zu speichern und weiterzuleiten. Mit Harvester lassen sich die entsprechenden URLs oder IP-Adressen extrahieren. Diese Systeme können dann entweder mit rechtlichen oder technischen Maßnahmen deaktiviert oder zur Informationsgewinnung von den Strafverfolgern infiltriert werden.

Insgesamt leistet Harvester daher einen wesentlichen Beitrag zur Verbesserung der Internetsicherheit und Privatsphäre von Milliarden von Smartphone-Nutzern – sowohl direkt durch die Bereitstellung von Informationen als auch indirekt durch die Unterstützung von Sicherheitsexperten, App Store-Betreibern, Antivirenanbietern und Entwicklern.

4 Zusammenfassung

In diesem Beitrag haben wir gezeigt, wie wichtig es für Datenschutz und IT-Sicherheit ist, Laufzeitwerte aus Apps extrahieren zu können. Diese Werte geben Aufschluss darüber, mit welchen Servern im Internet eine App kommuniziert, unter Nutzung welcher E-Mail-Kontos Daten versendet werden, und bieten erste Hinweise darauf, ob eine App Schadcode enthält oder nicht. In vielen Fällen liegen diese Werte jedoch nicht im Klartext in der App vor, sondern wurden bewusst verschleiert, um die App-Analyse zu erschweren. Mit dem in diesem Beitrag vorgestellten Werkzeug Harvester lassen sich die Werte auch in solchen komplexen Fällen ermitteln. Harvester funktioniert, indem es die App zuerst auf die zur Berechnung der Werte wesentlichen Anweisungen reduziert und diese Anweisungen dann isoliert ausführt. Damit vereinfacht Harvester die Arbeit von IT-Sicherheitsanalysten massiv und leistet so einen wesentlichen Beitrag zur Verbesserung der Internetsicherheit.

Literatur

[1] Jerry Dischler: *Building for the next moment*. Google AdWords Blog; 05.03.2015; https://adwords.googleblog.com/2015/05/building-for-next-moment.html

[2] Dennis Schirrmacher: *Fraunhofer patentiert Erkennung von Schläfer-Apps*. Heise Online; 12.02.2015; http://www.heise.de/security/meldung/Fraunhofer-patentiert-Erkennung-von-Schlaefer-Apps-2546240.html

[3] Frank Tip: *A Survey of Program Slicing Techniques*. Technical Report, Centre for Mathematics and Computer Science, Amsterdam, 1994.

[4] Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, Eric Bodden: *Harvesting Runtime Values in Android Applications That Feature Anti-Analysis Techniques*. In: 23rd Annual Network & Distributed System Security Symposium (NDSS), February 2016.

[5] Clay Dillow: *ISIS Has Its Own Secure Messaging App*. Fortune; 13.01.2016; http://fortune.com/2016/01/13/isis-has-its-own-secure-messaging-app/