Exercise 2

Aufgabe 2

Secure Design

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Security Champion Training

Trainingsgruppe 1 Office Exercise 2

Abgabe bis spätestens:

Freitag, 18.10.2019, 18:00 Uhr

Hinweise zur Bearbeitung und Abgabe (1/2)

Zeitlicher Rahmen: Die Übungszettel werden jeweils am Freitag einer Schulungswoche veröffentlicht. Die Bearbeitung eines Übungszettels ist ausgelegt auf maximal 16 Stunden pro Person, verteilt über 2 Wochen. Nachfragen: Anfallende Fragen zu den Aufgabenstellungen können via Mail gestellt werden (bitte immer an: stefan.dziwok@iem.fraunhofer.de und jan-niclas.struewer@iem.fraunhofer.de). Wir empfehlen zu Beginn die Aufgabenstellung einmal komplett zu lesen, um Fragen zu den Aufgaben frühzeitig zu klären da eine Antwort der Trainer auch mehr als einen Werktag dauern kann. Unbewertete Aufgaben: Nicht alle Aufgaben des Zettels fließen in die Bewertung ein. Dennoch empfehlen wir, diese Aufgaben mit der gleichen Gewissenhaftigkeit zu bearbeiten und die Antworten oder Berichte zu diesen Aufgaben in die Abgabe mit einzuschließen. Abgabe: Die Abgabe erfolgt in den am 27.08.2019 definierten Gruppen zu 4 Teilnehmern. Die Abgabe erfolgt zwei Wochen nach Ausgabe des Übungszettels durch Einchecken einer gemeinsamen ZIP-Datei in das gruppen-eigene Axa-GitHub. Bitte achtet darauf, dass alle Trainer und Organisatoren darauf Zugriff haben. Diese ZIP-Datei ist wie folgt zu benennen: Abgabe-Team<Teamnummer>-OfficeExercise<Exercisenummer>.zip Innerhalb der ZIP-Datei ist eine Textdatei mitglieder.txt mit einer Liste der Namen der Gruppenmitglieder, eine PDF-Datei loesungen.pdf mit den von der Gruppe verfassten Lösungen (Texte und Bilder) zu den bewerteten Aufgaben des Übungszettels (siehe Aufgabenunterschrift), sowie ggf. weitere explizit geforderte Dateien wie Quellcode und Modelle, abzulegen. Weiterhin sind die folgenden Information auf jeder Seite zu vermerken: Teamnummer sowie Teamname (in der Kopfzeile) und die Seitenzahl (in der Fußzeile). Abgaben, die gegen dieses Format verstoßen, werden nicht bewertet.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 1

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Hinweise zur Bearbeitung und Abgabe (2/2)

Bewertung: Zur Zulassung zur Abschlussprüfung müssen mindestens 60% der in allen Office Exercises erreichbaren Punkte erlangt werden. Die Korrektur erfolgt durch Mitarbeiter des Fraunhofer IEM. Die Gruppen erhalten in der Woche nach der Abgabe Feedback zu ihren Lösungen in einer Skype-Telefonkonferenz. Die Gruppen sind aufgefordert eigenständig in der Woche nach der Abgabe einen Termin mit den Trainern am Donnerstag oder Freitag zu vereinbaren. Die Dauer des Termins soll 30 Minuten betragen. Reichen mehrere Gruppen (teilweise) identische Abgaben ein, werden die betroffenen Aufgaben aller betroffenen Gruppen mit 0 Punkten bewertet.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 2

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Aufgabe 1: Wiederholung

Unbewertete Aufgabe Die folgenden Fragestellungen sollen euch bei der Nacharbeitung der Inhalte der Schulungstage und der weiterführenden Literatur unterstützen. Zudem bereiten sie euch auf die Abschlussprüfung vor.

1. Erläutert die in der Schulung vorgestellten Secure Design-Prinzipien in eigenen Worten.

2. Welchen Zweck erfüllen die Secure Design-Prinzipien im Allgemeinen?

3. Diskutiert die folgende Aussage: Eine konsequente Umsetzung des Prinzips Security by Obscurity reicht aus, um die Sicherheit einer Anwendung zu gewährleisten.

4. Beschreibt das Konzept der Distrustful Decomposition. Welche Design-Prinzipien werden umgesetzt? Wie können Prozesse untereinander kommunizieren?

5. Was ist Defensive Coding? In welcher Beziehung steht es zu Secure Design und zur Risikoanalyse?

6. Erläutert den Satz: Complexity is the enemy of security.

7. Warum müssen Eingaben validiert werden? Welche Arten der Validierung existieren?

8. Welches sind die 10 größten Sicherheitsrisiken für Webanwendungen 2017 laut der OWASP?

9. Kryptographie:

a) Definiert die CIA-Sicherheitsziele im Kontext der Kryptographie.

b) Was ist Kerckhoffis' Prinzip? Warum ist es sinnvoll?

c) Was ist "gute" Zufälligkeit?

d) Was sind Blockchiffiren? Wie unterscheiden sie sich von Stromchiffiren?

e) Wie unterscheiden sich symmetrische und asymmetrische Verschlüsselung?

f) Erläutert das Problem der Schlüsselverteilung.

g) Was ist hybride Verschlüsselung?

h) Was ist Hashing? Wie kann es erweitert werden, um die Sicherheitsziele Vertraulichkeit und Integrität zu erreichen?

i) Erläutert das Prinzip der digitalen Signaturen.

j) Skizziert die Funktionsweise des Diffie-Hellman-Algorithmus. Welches grundlegende Problem wird hierdurch gelöst?

10. Wieso reichen Tests nicht aus, um die Sicherheit einer Anwendung zu gewährleisten? Wie sorgen Code Reviews für einen höheren Sicherheitsstandard?

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 3

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

11. Wer nimmt an einem manuellen Code Review teil? In welche Phasen gliedert es sich?

12. Was ist Static Analysis? Welche Vor- und Nachteile bergen automatisierte Static Analysis-Tools gegenüber manuellen Code Reviews?

13. Erläutert kurz den prinzipiellen Ablauf einer automatischen Static Analysis.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 4

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Aufgabe 2: Secure Design & Defensive Coding

Bewertete Aufgabe (28 Punkte)

1. Secure Design-Prinzipien in der Theorie (2+2+2+2=8 Punkte) Die folgenden Szenarien beschreiben die Folgen nicht näher beschriebener Vulnerabilities des InsecureDMNManagers, den ihr bereits in der Office Exercise 1 kennengelernt habt.

a) Ein Angreifer kann dem Frontend Management, ohne zuvor erfolgte Authentifizierung, Dateien schicken. Wenn es sich hierbei um DMN-Dateien handelt, überschreibt das Frontend Management bereits vorhandene DMN-Dateien mit demselben Namen. Welches Secure Design-Prinzip wird in diesem Szenario verletzt? Sollten mehrere Prinzipien verletzt werden, so nennt bitte nur eines. Begründet eure Antwort bitte in maximal 2 Sätzen.

**Don‘t trust:** in diesem Fall wird jedem geglaubt, der eine Datei im richtigen Format liefert, dass er die Berechtigung hat auf das System zuzugreifen und auch dazu berechtigt ist, Dateien zu liefern. Informationen werden knallhart überschrieben (so impliziert die Beschreibung), wenn entsprechende Dateien bereits vorhanden sind, ohne darauf hinzuweisen, dass hier bereits eine Datei gleichen Namens vorhanden ist, bzw. ein Überschreiben zu verweigern.

b) Der AuthFilter erhält einen ungültigen Token und antwortet auf die Anfrage mit einer detaillierten Fehlermeldung. Das Frontend Management leitet diese Fehlermeldung unverändert an den Ursprung der Authentifizierungsanfrage weiter. Welches Secure Design-Prinzip wird in diesem Szenario verletzt? Sollten mehrere Prinzipien verletzt werden, so nennt bitte nur eines. Begründet eure Antwort bitte in maximal 2 Sätzen.

**Fail securely:** Es wird eine detaillierte Fehlermeldung über Trust boundaries hinweg weitergegeben, wodurch nicht sichergestellt werden kann, dass diese Fehlermeldung angezeigt wird. Informationen aus der Fehlermeldung können einem Angreifer Daten offenlegen oder Informationen über Infrastruktur, Funktionsweise von Programm und System liefern, die er für weitergehende Aktionen nutzen kann.

c) Ein Angreifer verändert die Anfrage Request DMN Diagram List, sodass diese zusätzliche Datenbankanfragen enthält. Das Backend Management prüft die Anfrage nicht vollständig, sodass alle Anfragen durchgeführt und ihre Ergebnisse in der Antwort enthalten sind. Welches Secure Design-Prinzip wird in diesem Szenario verletzt? Sollten mehrere Prinzipien verletzt werden, so privelege nennt bitte nur eines. Begründet eure Antwort bitte in maximal 2 Sätzen.

**Don‘t trust:** Es wird jedem Datenbank Request getraut, sodass ein ungeprüfter SQL-Request einem Angreifer die Möglichkeit bietet, per SQL-Injection Informationen aus der Datenbank abzufragen, die er für weitere Aktionen seines Angriffs nutzen kann.

d) Das Frontend Management führt keine Logs über Anmeldeversuche und Anmeldungen am InsecureDMNManager. Einem Angreifer ist es so möglich, sich mithilfe eines Brute Force-Angriffes einzuloggen und im Namen des gehackten Benutzers Änderungen an den DMN-Diagrammen durchzuführen. Welches Secure Design-Prinzip wird in diesem Szenario verletzt? Sollten mehrere Prinzipien verletzt werden, so nennt bitte nur eines. Begründet eure Antwort bitte in maximal 2 Sätzen.

**Detect and record**: Der durch unzählige Versuche erkennbare Brute Force Angriff wird im Log nicht festgehalten und ist so ist im Nachhinein nicht mehr nachvollziehbar. Auch der Nachweis, dass unerwünschte Änderungen an einem DMN-Diagramm tatsächlich von einem bestimmten Benutzer durchgeführt **nicht selbst** wurden, ist so nicht mehr erbringbar.

2. Defensive Coding (5+5+5+5=20 Punkte) In dieser Aufgabe betrachten wir unterschiedliche Java 8-Programme, die ihr ebenfalls im SecurityChampionTraining Git finden könnt. Diese Programme erreichen die in der jeweiligen Aufgabe genannten Ziele, die Programmierer haben jedoch die Richtlinien des Defensive Coding, die ihr an Präsenztag 3 kennengelernt habt, nicht bedacht. Eure Aufgabe ist es, die Richtlinienverstöße zu identifizieren, zu erläutern, die entstandene Vulnerability exemplarisch auszunutzen und den Verstoß zu beheben.

Konkret sollt ihr jeweils folgende Fragen beantworten bzw. Aktionen durchführen:

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 5

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

1# Welche Defensive Coding-Richtlinie wird in diesem Beispiel verletzt? Sollten mehrere Richtlinien verletzt werden, so nennt bitte nur eine.

2# Erläutert die Schwachstelle in maximal 4 Sätzen anhand der Stelle(n) an der die Verletzung auftritt.

3# Gebt bitte ein Programm bzw. einen Aufruf an, der den Exploit durchführt. Alternativ könnt ihr in in maximal 3 Sätzen beschreiben wie ein Exploit aussieht, der die entstandene Vulnerability ausnutzt.

4# Behebt bitte den Fehler anschließend und gebt den Code im PDF und als extra Datei ab.

Bitte fügt alle eure Programmierartefakte in den Zip-Container. Bitte fügt zudem euren Code auch in das Abgabe-PDF ein.

a) Das folgende Programm ermöglicht das Auslesen von Kundeninformationen aus einer Datenbank.

Ad 1# Missachtete Defensive Code Richtlinie: Input Validation, Immutability  
Ad 2# Vulnerability: SQL Injection  
**|**Ad 3# Exploit 1: in Zeile 8 – **costumerName =** **b‘ or \*; --**  
Ad 4# Behebung: prepared Statement - ersetzt Zeile 8:  
PreparedStatement p = null;  
p = conn.preparedStatement(“SELECT \* FROM Customers WHERE Name = ?“)  
p.setString(1, customerName);  
Resultset res = p.executeQuery();

1 public class CustomerManagement {

2 public ResultSet getCustomerDetail ( String customerName) {

3 if (customerName != null && ! customerName.equals ("") ) {

4 Connection conn = this.establishConnection () ;

5 if (conn != null ) {

6 try \*

7 Statement stmt = conn.createStatement () ;

8 ResultSet result = stmt.executeQuery ("SELECT \* FROM Customers WHERE Name = '

" + customerName + " '") ;

9 if (! result.isBeforeFirst () ) {

10 return null ;

11 }

12 return result ;

13 } catch(SQLException e) {

14 System.out.println ("Could not obtain customer detail ! ") ;

15 } finally {

16 conn.close () ;

17 }

18 return null ;

19 }

20 }

21 }

22 public Connection establishConnection () {

23 Connection conn = null ;

24 try {

25 Class.forName("org.h2.Driver").newInstance () ;

26 conn = DriverManager.getConnection ("jdbc : h2 :mem: db1" , "" , "") ;

27 } catch( InstantiationException | IllegalAccessException | ClassNotFoundException |

SQLException e) {

28 System.out.println ("Could not establish database connection ! ") ;

29 }

30 return conn ;

31 }

32 }

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 7

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

b) Das folgende Programm ermöglicht den Inhalt einer Datei auf der Konsole auszugeben:

Ad 1# Missachtete Defensive Code Richtlinie: Exception Handling (Cleaning up)  
Ad 2# Vulnerability: dead Objects  
Ad 3# Exploit: Zeile 10 in der Methode read(.) nicht abgesichertes Öffnen eines FileReaders. Im Fehlerfall bleibt das Objekt offen und kann zum Missbrauch genutzt werden.  
Ad 4# Behebung: Verwendung von try/catch-Surrouding mit FileReader.close in einem finally-Block, kein „throws“ der FileNotFoundException im Methodenheader

try {  
 Scanner fileReader = new Scanner(file)  
 …  
} catch( FileNotFoundException e){  
 //welches Errorhandling auch immer  
} finally {  
 fileReader.close();  
}

1 public class FileReader {

2 public static void printContent ( String fileName ) throws IOException {

3 System.out.println ("There is nothing you can hide ! ") ;

4 for ( String line : read ( getFile ( fileName ) ) ) {

5 System.out.println ( line ) ;

6 }

7 }

8 public static List<String> read ( File f i l e ) throws FileNotFoundException {

9 List<String> lines = new ArrayList<String >() ;

10 Scanner fileReader = new Scanner ( f i l e ) ;

11 while( fileReader.hasNextLine () ) {

12 lines.add( fileReader.nextLine () ) ;

13 }

14 fileReader.close () ;

15 return lines ;

16 }

17 public static File getFile ( String name) throws NullPointerException {

18 return new File (name) ;

19 }

20 }

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 8

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

c) Das folgende Programm überprüft, ob eine beliebige Zahl prim ist.

Ad 1# Missachtete Defensive Code Richtlinie: Dead Store Removal  
Ad 2# Vulnerability: Der als Class Parameter im Konstruktor bereits initialisierte Array primes wird in der Methode isPrime(.) erneut initialisiert. Der bereits allocierte Array wartet nun aufs Abräumen durch den GarbageCollector und ist mit potentiell sensiblen Daten gefüllt.   
Ad 3# Exploit: Ein Angreifer könnte diese dead stores über Hybernation aus dem RAM dumpen und so an die im dead store liegenden Informationen gelangen.  
**|**Ad 4# Behebung: überflüssige Initialisierung in isPrime(.) (Zeile 23,24), die Initialisierung erfolgte bereits **über den Konstruktor.**

1 public class Sieve {

2 private boolean [ ] primes ;

3 public Sieve ( int n) {

4 primes = new boolean [n+1];

5 initArray () ;

6 }

7 private void initArray () {

8 for ( int i =2; i < primes.length ; i++) {

9 primes [ i ] = true ;

10 }

11 }

12 private void sieve () {

13 for ( int i =2; i < Math. sqrt ( primes.length ) ; i++) {

14 for ( int k=i +1; k < primes.length ; k++) {

15 if (k % i == 0) {

16 primes [ k ] = false ;

17 }

18 }

19 }

20 }

21 public boolean isPrime ( int i ) {

22 if ( i >= numbers.length ) {

23 primes = new boolean [ i +100];

24 initArray () ;

25 sieve () ;

26 } else if ( i < 2) {

27 return false ;

28 } else {

29 return primes [ i ] ;

30 }

31 }

32 public void setPrimes (boolean [ ] primes ) {

33 this.primes = primes ;

34 }

35 public boolean [ ] getPrimes () {

36 return primes ;

37 }

38 }

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 9

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

d) Das folgende Programm simuliert zwei Autoren, die sich einen Block zum Niederschreiben ihrer Ideen teilen. Behebt die hier enthaltene Schwachstelle bitte so, dass erst ein Autor 10 Ideen niederschreibt bevor der zweite Autor 10 Ideen niederschreibt.

Ad 1# Missachtete Defensive Code Richtlinie: Concurrency  
Ad 2# Vulnerability: Multithreading hier nicht erforderlich → KIS-Prinzip  
Ad 3# Exploit: Bei Abbruch bleibt der Thread offen und kann für einen Angriff genutzt werden.  
Ad 4# Behebung: Start der Writer mit try/Catch/finally → obligatorisches Close der Writer im finally Block. Kein Multithreading sondern sequenzielles schreiben.

1 public class Pad {

2 private List<String> ideas = new ArrayList<String >() ;

3 public static void main( String [ ] args ) {

4 Pad pad = new Pad() ;

5 Writer w1 = new Writer (0 , 50 , pad) ;

6 Writer w2 = new Writer (50 , 50 , pad) ;

7 w1. start () ;

8 w2. start () ;

9 }

10 public void writeDown( String idea ) {

11 ideas.add( idea ) ;

12 System.out.println ("Written down: " + idea ) ;

13 }

14 }

15 public class Writer extends Thread {

16 private final int lb , ub;

17 private final Pad pad ;

18 public Writer ( int lb , int ub , Pad pad) {

19 this.lb = lb ;

20 this .ub = ub;

21 this.pad = pad ;

22 }

23 @Override

24 public void run () {

25 for ( int i =0; i < 10; i++) {

26 String idea = "" + ( int ) ( lb + Math. random() #ub) ;

27 pad.writeDown( idea ) ;

28 try {

29 this.sleep (10) ;

30 } catch( InterruptedException e) {

31 e.printStackTrace () ;

32 }

33 }

34 }

35 }

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 10

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Aufgabe 3: Verschlüsselungsalgorithmen

Bewertete Aufgabe (44 Punkte)

1. One-Time Pad (2+2+2=6 Punkte) One-Time Pad (OTP) ist ein Verschlüsselungsverfahren, das 1882 durch den amerikanischen Kryptographen Frank Miller erstmal beschrieben wurde. Einen kurzen Überblick der Funktionsweise von OTP findet ihr hier.

[https://de.wikipedia.org/wiki/One-Time-Pad#Beispiel](https://de.wikipedia.org/wiki/One-Time-Pad" \l "Beispiel)

a) Beschreibt die Funktionsweise von OTP in eigenen Worten. Verwendet für eure Beschreibung nicht mehr als 4 Sätze.

Zeichenweise Verschlüsselung durch Kombination des Zeichenschlüssels mit dem Klartext meist XOR, aber auch Addition ist möglich.  
Der gesamte Schlüssel ist so lang wie der zu verschlüsselnde Text, Schlüssel wird einmalig verwendet  
Moderne Verschlüsselung über Bits und Bytes, früher Großbuchstabentabelle, erweiterbar mit Kleinbuchstaben, Zahlen und Sonderzeichen.  
Nachweislich nicht brechbar

b) Handelt es sich bei OTP um ein symmetrisches oder um ein asymmetrisches Verschlüsselungsverfahren? Begründet eure Antwort!

Symmetrische Verschlüsselung: Es liegt ein Schlüssel vor, der zur Ver- und Entschlüsselung benötigt wird. D. h. der Empfänger der Nachricht muss den Schlüssel kennen, um diese entschlüsseln zu können.

c) Welche Probleme bestehen bezüglich des Schlüssels und des Schlüsseltauschs?

Eine sichere Übermittlung des Schlüssels zur Dechiffierung zu gewährleisten.

2. Data Encryption Standard (5+2+2=9 Punkte) Alice und Bob möchten über einen sicheren Kanal miteinander kommunizieren. Sie verwenden zur Verschlüsselung ihrer Nachrichten ein an den Data Encryption Standard (DES)-Algorithmus angelegtes Verfahren. Dazu erzeugt Alice zunächst einen geheimen Schlüssel und schickt diesen Bob via E-Mail. Alle folgenden Nachrichten zwischen Alice und Bob werden mithilfe dieses Schlüssels verschlüsselt.

a) Illustriert den vollständigen Ablauf (d.h. inklusive Schlüsselerzeugung) der Kommunikation zwischen Alice und Bob anhand eines UML-Sequenzdiagramms. Benutzt dabei für die Beschriftung der Übergänge:

K Der für die Verschlüsselung verwendete Schlüssel.

generateKey() Erzeuge einen Schlüssel.

send(msg) msg wird von Alice an Bob (oder anders herum) geschickt.

decrypt(msg, key) msg wird mithilfe von key entschlüsselt.

encrypt(msg, key) msg wird mithilfe von key verschlüsselt.

→ |generateKey()|

←key← | |

→ |send(key) | →key→

**Alice** **Bob**

→ |encrypt(msg, key)|

← encMsg← | |

→ |send(encMsg) | →encMsg→

|decrypt(encMsg, key)| ←

| | →msg→

b) Handelt es sich bei DES um ein symmetrisches oder ein asymmetrisches Verschlüsselungsverfahren? Begründet eure Antwort in maximal 2 Sätzen.

Symmetrisches Verfahren. Der zur Verschlüsselung verwendete Schlüssel wird für die Entschlüsselung der Nachricht gebraucht.

c) Warum ist die beschriebene Kommunikation unsicher? Begründet eure Antwort in maximal 2 Sätzen.

Schlüsselversand erfolgte via eMail. Ein eMail-Versand gilt als nicht sicher und kann deshalb als allenfalls als Security by Obscurity angesehen werden. Durch eine gezielte Man in the Middle-Attacke könnte der Schlüssel mitgelesen werden und so nachfolgende verschlüsselte Kommunikation ebenfalls.

3. Pretty Good Privacy (10+2+2=14 Punkte) Alice und Bob haben erkannt, dass ihre Kommunikation nicht sicher ist. Daher wollen sie ihr Verschlüsselungsverfahren nun an Pretty Good Privacy (PGP) anlehnen. In diesem generieren sowohl Alice als auch Bob einen öffentlichen und einen privaten Schlüssel. Ihre öffentlichen Schlüssel teilen sie miteinander

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 11

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

per E-Mail. Bob verschlüsselt Nachrichten an Alice mit Alice' öffentlichem Schlüssel. Alice entschlüsselt Nachrichten von Bob mit ihrem privaten Schlüssel und umgekehrt.

a) Illustriert den vollständigen Ablauf (d.h. inklusive Schlüsselerzeugung) der Kommunikation zwischen Alice und Bob anhand eines UML-Sequenzdiagramms. Benutzt dabei für die Beschriftung der Übergänge:

PKAlice bzw. PKBob Die öffentlichen Schlüssel von Alice bzw. Bob.

SKAlice bzw. SKBob Die privaten Schlüssel von Alice bzw. Bob.

generateKeyPair() Erzeuge ein Schlüsselpaar.

send(msg) msg wird von Alice an Bob (oder anders herum) geschickt.

decrypt(msg, key) msg wird mithilfe von key entschlüsselt.

encrypt(msg, key) msg wird mithilfe von key verschlüsselt.

→ |generateKeyPair()|

←SKPK←| |

→ |generateKeyPair()|

| | →SKPK→

**Alice**  **Bob**

→ |encrypt(msg, PKBob) |

← encMsg← | |

→ |send(encMsg) | →encMsg→

|decrypt(encMsg, SKBob) | ←

| | →msg→

|encrypt(msg2, PKAlice) | ←

| | →encMsg2→

**|** **←encMsg2←** |send(encMsg2) | ←

→ |decrypt(encMsg2, SKAlice)|

←msg2← | |

b) Handelt es sich bei PGP um ein symmetrisches oder ein asymmetrisches Verschlüsselungsverfahren? Begründet eure Antwort in maximal 2 Sätzen.

Asymmetrisches Verfahren. Pretty Good Privacy beruht auf dem RSA-Algorithmus, bei dem ein privater und ein öffentlicher Schlüssel erzeugt werden. Die Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel, den nur der Schlüsselersteller kennt, entschlüsselt werden.

c) Wie werden die Probleme der vorherigen Kommunikation gelöst? Formuliert eure Antwort in maximal 4 Sätzen.

Da der öffentliche Schlüssel nur für die Verschlüsselung zu gebrauchen ist, kann diesen jeder kennen. Er kann ohne Sicherheitsbedenken versendet werden, auch per Mail. Eine Nachricht ist nur mit dem privaten Schlüssel zu entschlüsseln, der beim Schlüsselerzeuger bleibt und daher sicher ist. Der kritische Faktor des Schlüsselversands entfällt damit.

4. Transport Layer Security (TLS) (11+2+2=15 Punkte) Alice und Bob sind froh darüber, dass ihre Kommunikation nun deutlich sicherer ist als zuvor. Es stört sie jedoch, dass die Ver- und Entschlüsselung deutlich mehr Zeit in Anspruch nimmt als zuvor. Daher wollen sie nun ihre Verschlüsselung nach dem Beispiel von Transport Layer Security (TLS) erweitern. Nach wie vor müssen beide jeweils einen öffentlichen und einen privaten Schlüssel generieren und ihre öffentlichen Schlüssel miteinander teilen. Um eine Unterhaltung mit Alice zu beginnen, wählt Bob ein symmetrisches Verschlüsselungsverfahren aus und generiert einen geheimen Schlüssel für dieses Verfahren. Nun nutzt er Alice' öffentlichen Schlüssel, um eine kurze Nachricht zu verschlüsseln. Diese Nachricht enthält das verwendete symmetrische Verschlüsselungsverfahren sowie den zugehörigen Schlüssel. Diese Informationsnachricht schickt er an Alice. Alice entschlüsselt die Informationen über das symmetrische Verschlüsselungsverfahren mit ihrem privaten Schlüssel. Alle weiteren Nachrichten zwischen Bob und Alice werden mithilfe des symmetrischen Verschlüsselungsverfahrens und dem zugehörigen Schlüssel verbzw. entschlüsselt.

a) Illustriert den vollständigden Ablauf (d.h. inklusive Schlüsselerzeugung) der Kommunikation zwischen Alice und Bob anhand eines UML-Sequenzdiagramms. Benutzt dabei für die Beschriftung der Übergänge:

PKAlice bzw. PKBob Die öffentlichen Schlüssel von Alice bzw. Bob.

SKAlice bzw. SKBob Die privaten Schlüssel von Alice bzw. Bob.

SymKey Der

→ |generateKeyPair()|

←SKPK←| |

|generateKeyPair()| ←

| | →SKPK→

→ |generateSymmetricKey(DES) |

←SymKey | |

**|** → |encrypt(**SymKey, PKBob**)|

← encSymKey← | |

**|** → |send(encSymKey) | →**encSymKey**→

|decrypt(encSymKey, SKBob) | ←

| | →SymKey→

**Alice**  **Bob**

→ |encrypt(msg, SymKey)|

← encMsg← | |

→ |send(encMsg) | →encMsg→

|decrypt(encMsg, SymKey) | ←

| | →msg→

|encrypt(msg2, SymKey) | ←

| | →encMsg2→

← |send(encMsg2) | ←encMsg2←

→ |decrypt(encMsg2, SymKey) |

←msg2← | |

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 12

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

generateKeyPair() Erzeuge ein Schlüsselpaar.

generateSymmetricKey(method) Erzeuge einen Schlüssel für das symmetrische Verschlüsselungsverfahren method.

send(msg) msg wird von Alice an Bob (oder anders herum) geschickt.

decrypt(msg, key) msg wird mithilfe von key entschlüsselt.

encrypt(msg, key) msg wird mithilfe von key verschlüsselt.

b) Handelt es sich bei TLS um ein symmetrisches oder ein asymmetrisches Verschlüsselungsverfahren? Begründet eure Antwort in maximal 2 Sätzen.

Asymetrisches Verfahren zur sicheren Schlüsselübermittlung für die Kommunikationsverschlüsselung, Symmetrisches Verfahren zur Datenübermittlung. Der symmetrische Schlüssel ist für die gesamte Kommunikation während einer Session gültig.

c) Welches Problem asymmetrischer Verschlüsselungsverfahren mindert TLS? Formuliert eure Antwort in maximal 4 Sätzen.

**|**Zeitaufwand durch **Verwendung** asymmetrischer Verschlüsselung, da die Kommunikation über die weniger zeitintensive symmetrische Verschlüsselung erfolgt und nur der Schlüsselaustausch dafür über asynchrone Verschlüsselung erfolgt.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 13

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Aufgabe 4: CogniCrypt & SpotBugs

Bewertete Aufgabe (15 Punkte)

1. Password Manager (0+0+5+10=15 Punkte) Der Password Manager ist eine einfache Variante eines lokalen Passwortmanagers, mit dessen Hilfe ein Benutzer seine verschiedenen Passwörter verschlüsselt speichern kann. Der Benutzer kann mithilfe des Password Managers ebenfalls die Passwörter entschlüsseln und sich als Liste anzeigen lassen. Der Entwickler ist jedoch noch unerfahren im Umgang mit Javas Kryptographie-Bibliotheken und hat diese fehlerhaft verwendet.

In dieser Aufgabe sollt ihr den Code mittels zweier Eclipse-WErkzeuge analysieren und korrigieren: CogniCrypt und FindSecBugs, welches ein Plugin von SpotBugs ist.

a) Ladet das Projekt Password Manager aus eurem SecurityChampionTraining Git herunter. Es handelt sich hierbei um ein Java 8-Programm.

b) # Installiert euch Eclipse.

# Anschließend installiert CogniCrypt und SpotBugs über die Eclipse-Update-Site.

# Ladet euch anschließend das Jar von FindSecBugs herunter.

# In den Eclipse Preferences solltet ihr in der Kategorie CogniCrypt die Option Enable automated analysis when saving aktivieren.

# In den Eclipse Preferences geht ihr auf die Kategorie Java#SpotBugs. In dieser müsst ihr im Reiter Plugins and misc. Settings das FindSecBugs-Jar hinzufügen. Anschließend müsst ihr noch im Reiter Reporter Configuration bei den Reported (visible) bug categories die Kategorie Security aktivieren.

# CogniCrypt startet nun seine Analyse sobald ihr in euren Java-Files etwas ändert und anschließend speichert.

# Um die FindSecBugs-Analyse zu starten müsst ihr im Kontextmenü des Java-Projekts (rechte Maustause auf das Projekt) die Option SpotBugs#FindBugs auswählen.

# Insgesamt solltet ihr nun 7 Fehler von CogniCrypt und 6 Warnungen von SpotBugs erhalten.

Wir haben zur Erstellung der Aufgabe und der Lösung Java 8 (jre1.8.0\_192), Eclipse 2019-09 R (4.13.0), CogniCrypt 1.0.0.201905151726 und SpotBugs 4.0.0.201904010749-792e955 mit dem FindSecBugsPlugin 1.9.0 verwendet.

c) Überarbeitet den Code, so dass weder CogniCrypt noch SpotBugs bzw. FindSecBugs Fehler bzw. Warnungen anzeigen. Bitte fügt eurer ZIP-Datei den korrigierten Quellcode hinzu.

d) Erläutert jeden Fehler, den ihre gefunden habt, und eure jeweils vorgenommene Korrektur in jeweils maximal 2 Sätzen.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 14

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

Aufgabe 5: Ver- & Entschlüsseln am praktischen Beispiel

Bewertete Aufgabe (22 Punkte) Das folgende Python 3-Programm implementiert einen aus der Schulung bekannten Algorithmus, um einen als Kommandozeilenparameter übergebenen Klartext zu verschlüsseln. Den Code findet ihr ebenfalls im SecurityChampionTraining git.

*Das Codebeispiel verstehe ich nicht durchgehend. Ich kenne Python nicht, vielleicht liegt es daran….*

*Ich vermute mal, dass in magic 2 ein Hashing durchgeführt wird, das dann über magic3 noch ein bisschen aufgemischt wird. Da ein Hashing nicht wieder eindeutig rückgeführt werden kann, ist die Lösung für die Entschlüsselung keine Wirkliche Entschlüsselung, sondern reines Guessing. Funktioniert auch nur, weil die Hash-Arrays so klein gehalten sind. Wir führen also eine Brute force Attacke durch: Wir müssten demnach ein Wörterbuch mit allen möglichen Hashes erstellen und die einzelnen Teile des verschlüsselten Textes dagegen vergleichen. Der daraus entstandene Text besteht dann auch noch aus sich überlappenden Arrays. … Das würde auch nur funktionieren, wenn tatsächlich ein einziger eindeutiger Klartext ermittelt werden könnte.*

*Ich glaube, Gero, hier ist Deine Expertise gefragt.*

1 import base64

2 import sys

3

4 def magic3(b) :

5 return pow(b, 65539 , 928115603367739043)

6

7 def magic2(b) :

8 res = 0

9 for c in b:

10 res = ( res << 8) + ord(c)

11 return res

12

13 def magic1(input) :

14 some\_array = [ magic2(input [ i : i +4]) for i in range(0 , len (input) , 4) ]

15 another\_array = [3735928559]

16 for b in some\_array :

17 another\_array.append(magic3(b) ^ another\_array [ 1])

18 some\_string = ""

19 for a in another\_array :

20 some\_string += str (a).z f i l l (19)

21 return base64.b64encode ( bytes ( some\_string , "utf8") )

22

23 def main() :

24 output = magic1( sys.argv [1])

25 print ( str (output , "utf 8") )

26

27 if \_\_name\_\_ == "\_\_main\_\_" :

28 main()

Ihr findet dieses Programm ebenfalls im Schulungs-Git.

1. Analyse (6+3=9 Punkte)

a) Beschreibt kurz die Schritte, die das Programm zur Verschlüsselung eines beliebigen Klartextes durchführt.

b) Welche Art von Verschlüsselungsverfahren wird hier implementiert? Handelt es sich um ein symmetrisches oder asymmetrisches Verschlüsselungsverfahren? Begründet eure Antwort in maximal 4 Sätzen.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 15

Security Champion Training Dr. Stefan Dziwok, Felix Barczewicz Fraunhofer IEM

2. Entwurf (12+1=13 Punkte)

a) Entwickelt ein Programm, das mit dem obigen Programm verschlüsselte Klartexte entschlüsseln kann. Ihr dürft selbst entscheiden, ob ihr dieses Programm in Java 8 oder Python 3 entwickelt. In jedem Fall fügt eurer ZIP-Datei bitte den Quellcode eures Programms hinzu und fügt den Code bitte auch in euer PDF ein.

b) Entschlüsselt den in enc.txt abgelegten Chiffretext und gebt den Klartext an.

Office Exercise 2 Letzte Änderung: 4. Oktober 2019, 19:58 16