Zusammenfassung Information Security FS2018

Alex Neher

June 8, 2018

Inhalt

Abbildungsverzeichnis

1 Teil Pouly

1.1 Nach welchen Kriterien lassen sich Hacker und ihre Opfer klassifizieren?

Hacker

- Hacktivisten
- Cyberterroristen
- Staatliche Organisationen
- Skript Kiddies
- Organisiertes Verbrechen
- Wissenschaftler / Krypto-Analytiker

Opfer

- Einzelpersonen (gezielt)
- Einzelpersonen (zufaellig)
- Die grosse Masse (zufaellig)
- Organisationen / Firmen (gezielt)

1.2 Wleche Angriffstrategien werden typischerweise gegen welche Opfer eingesetzt?

- Einzelpersonen (gezielt)

  Identitätsdiebstahl

  Informationsdiebstahl

  Diskreditierung

- Einzelpersonen (zufällig)

  Phishing

  Skimming

- Die grosse Masse (zufällig)

  Viren-Angriffe auf bestimmte Betriebssysteme

- Firmen und Organisationen (gezielt)

  DDoS

1.3 Beschreiben Sie zwei unterschiedliche Geschäftsmodelle von Hacking Organisationen

- Erpressung
- DDoS As a Service

1.4 Was versteht man unter Identitätsdiebstahl?

1. Ein Hacker stielt das E-Mail Passwort seines Vorgesetzten.
2. Ein Hacker stielt das Login Passwort zum PC eines Mitarbeiters.
3. Ein Hacker stielt ein Facebook Passwort.

In den USA ist Identitaetsdiebstahl ein grosses Problem, weil es dort – anders als in der Schweiz – mit der Sozialversicherungsnummer ein personenbezogenes Merkmal gibt, das hinreichend ist, um Steuern zurueckzufordern und Kreditkar- tenvertraege abzuschliessen. [...] Allein 2011 belief sich der finanzielle Schaden infolge von Identitaetsdiebstahl auf 3.6 Mrd. USD. NZZ, 20.01.2014

1.5 Beschreiben Sie mögliche Konsequenzen von Identitätsdiebstahl für die Opfer

- Finanzielle Folgen (Bestellungen auf meinen Namen etc)
- Rechtliche Folgen (Kinderpornografie etc)
- Gesellschaftliche Folgen (Postings von meinem FB Account)

1.6 Analysieren Sie die Authentifikationsmethoden 1-4 auf Stärken und Schwächen

1. Telnet and Remote Shell

Abbildung 1: Authentifizierung 1

2. Mailinglist, Wikis, Foren

Abbildung 2: Authentifizierung 2

3. Windows Login

Abbildung 3: Authentifizierung 3

4. UNIX Login

Abbildung 4: Authentifizierung 4

1.7 Wie schützen Betriebssysteme typischerweise unsere Identitäten und wie kann dieser Schutz mit Maschinenzugriff ausgehebelt werden?

Windows
Windows Betriebssysteme (seit NT) speichern Passwortinformationen in der Re- gistry Datei Security Account Manager (SAM). Windows Kernel sichert SAM, so dass Datei nicht kopiert werden kann.

So geht es trotzdem... Dumping File Sectors directly from Disk using Logical Offsets → Cain
Wenn SysKey aktiviert ist, verschluesselt Windows die SAM Datei partiell.
Wenn SysKey aktiviert ist, holt bkhive den Schluessel aus der Registry.

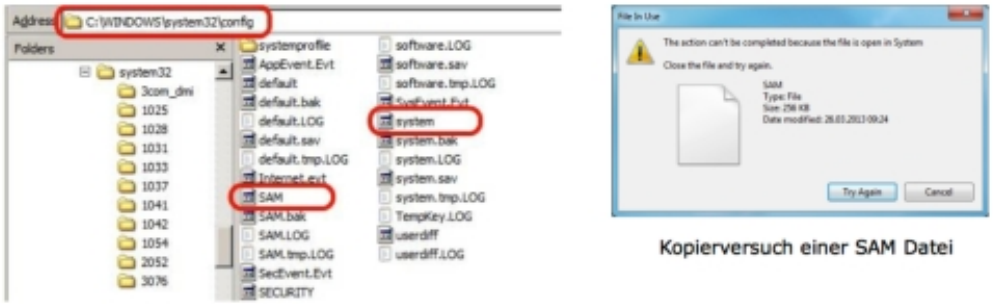

Abbildung 5: Password Windows

Linux
Unix Betriebssysteme (seit 1990) speichern Passwortinformationen in Shadow Dateien. Nur Superuser koennen diese Dateien lesen.

John the Ripper beinhaltet `unshadow` Befehl

```
> umask 077
> unshadow /etc/passwd /etc/shadow > mypasswd
```

Speicherort von Passwortdateien:

Alte Unix Versionen (vor 1990)	→ `/etc/passwd`
Moderne Unix Versionen	→ `/etc/shadow`
BSD Unix	→ `/etc/master.passwd`
Mac OS (10.2 und früher)	→ `/private/var/db/netinfo/local.nidb/`
Mac OS (10.3 – 10.6)	→ `/var/db/shadow/hash/`
Mac OS (10.7 und später)	→ `/var/db/dslocal/nodes/Default/users/`

Abbildung 6: Password Linux

Fazit
• In Klartext gespeicherte (z.B. FileZilla) oder gecachte (z.B. Windows- Domaenen Anmeldung) Passwoerter koennen mit Maschinenzugriff leicht geklaut werden.
• Betriebssysteme schuetzen Passwortdateien, jedoch kann der Schutz leicht umgangen werden.
• Anwendungsprogramme und Betriebssysteme verschluesseln Passwoerter bevor sie in einer Passwortdatei abgelegt werden.

1.8 WelcheAngriffsstrategien auf gestohlene Passwort-Hashes hat ein Hacker zur Verfügung?

1. Passwort-Hashfunktion knacken
2. Brute-Force Attacke

   Alle möglichen Kombinationen werden systematisch durchprobiert

   Jedes mögliche Passwort wird verschlüsselt und in der Passwort-Datei gesucht

   Irgendeinmal wird jedes System so geknackt → Vollständige Methode
3. Dictionary Attacke

   Nur Wörter aus einem vorgegebenen Wörterbuch werden durchprobiert

   Beruht auf der Annahme, dass sinnvolle Wörter als Passwort verwendet werden, oder dass das Passwort auf einem Algorithmus beruht (z.B. Palindrom)

   Knackt das System nur, falls sich das Passwort im Wörterbuch befindet → Unvollständige Methode
4. Lookup Tabelle

   Wörterbuch mit Wort und dazugehörigem Hash

   Wörterbuch mit 1 Million Wörtern verlangt 1 Million Hashberechnungen

   Es muss nun nur noch nach dem Hashwert gesucht werden

   Es wird mehr Memory benötigt, dafür wird Zeit gespart

   Gleich wi¡e Dictionary-Attack: Knackt nur Passwörter in der Tabelle → Unvollständige Methode

1.9 Sie können den Aufwand von Brute-Force Attacken vorausbereichnen

Passwörter bestehen aus alphanumerischen Symbolen (a-z,A-Z,0-9) und 8 möglichen Zusatzzeichen (26 + 26 + 10 + 8 = 70 Zeichen). Das CMS Joomla verschlüsselt die Passwörter mit MD5. Angenommen, man verwendet die Software oclHashcat und 2 GPUs mit 880Mhz getaktet und 1250Mhz getaktete RAM. Die Benchmark zeigt 23083.9 M/s an → ca. $23 * 10^9$ Hashes/s

Passwort der Länge 7

$$\frac{70^7 Hashes}{23 * 10^9 Hashes/sek} = 358sek = 5.96Min$$

Passwort der Länge 9

$$\frac{70^9 Hashes}{23 * 10^9 Hashes/sek} = 1754504sek = 487.3h = 20.3Tage$$

Passwort der Länge 11

$$\frac{70^{11} Hashes}{23 * 10^9 Hashes/sek} = 8597072795sek = 99503TAge = 272.6Jahre$$

1.10 Nennen Sie den Unterschied zwischen Dictionaries und Lookup-Tabellen

Lookup-Tabellen sind vorgefertigte Tabellen, die einen Hashwert und das dazugehörige Klartext-Passwort enthalten. Die Hashes sind bereits berechnet. Man muss nur noch anhand des Hashes, welches z.B. im shadows File gefunden werden kann, das Passwort heraussuchen.

Ein Dictionary ist im Gegensatz zur Lookup-Tabelle nur eine Sammlung von möglichen Passwörtern im Klartext. Man kann z.B. einen einfachen Duden als Dictionary verwenden. Anschliessend wird für jedes Wort den Hash berechnet und mit dem gewünschten Hash verglichen.

1.11 Sie können die Funktionsweise von Rainbow-Tables anhand eines Beispiels erklären

Zur Vereinfachung wird bei diesem Beispiel keine Hash-Methode sondern eine einfache multiplikative Methode (Formel 1) verwendet. Ebenfalls besteht das Passwort ausschliesslich aus Ziffern und ist zweistellig (also 00-99).

$$h(k) = \lfloor m * (kA \bmod 1) \rfloor \tag{1}$$

Für das Passwort k = 78, Multiplikator m = 2000 und A = Goldener Schnitt = 0.618 gilt also:

$$h(k) = \lfloor 2000 * (78 * 0.618 \bmod 1 \rfloor) = \lfloor 2000 * 0.204 \rfloor = \lfloor 408 \rfloor \tag{2}$$

Da unser Hashwert aus vier Stellen besteht, wird daraus 0408. Wiederholen wir das Ganze für jedes mögliche Passwort, erhalten wir eine Tabelle mit dem Passwort und dem dazugehörigen Hashwert:

Passwort	Hash
NULL	NULL
01	1236
02	0472
03	1708
[...]	[...]
78	0408
[...]	[...]
99	0364

Um diese normale Tabelle in eine Regenbogentabelle umzuwandeln benötigen wir Reduktionsfuntktionen. Diese Funktionen reduzieren den Hashwert und sparen so Speicherplatz. Bei einem zweistelligen Passwort mag das unsinnig erscheinen, aber wenn man 10+ stellige alphanumerische Passwörter hat, dann können solche Tabellen schnell mal sehr gross werden. Eine sehr einfache Reduktionsfunktion wäre zum Beispiel nur die letzten beiden Ziffern unseres vierstelligen Hashes zu speichern. Aus 0408 wird nun also 08. Nun hasht man diesen reduzierten Hashwert wieder mit der vorhin verwendeten Methode (Formel 1). Es kann frei gewählt werden, wie oft man das ganze Spiel wiederholt. Je mehr Wiederholungen man macht, desto grösser wird jedoch die Tabelle und nimmt somit auch mehr Speicherplatz ein.

P1	H1	P2	H2	P3	H3	P4	H4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
01	1236	36	0496	96	0656	56	1215
02	0472	72	0992	92	1712	12	0832
03	1708	08	1888	88	0768	68	0048
04	0944	44	0384	84	1824	24	1664
05	0180	80	0879	79	1644	44	0384
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]

Da man die Tabelle jedoch so klein wie möglich halten will, behält man nur den ersten und den letzten Wert der Zeile:

P1	H4
NULL	NULL
01	1215
02	0832
03	0048
04	1664
05	0384
[...]	[...]

Angenommen man findet jetzt heraus, dass der Hashwert 1888 zu einem Passwort gehört, führt man folgende Schritte durch:

1. Man sucht den Wert 1888 in der Tabelle → Er ist nicht in der Tabelle
2. Man führt eine Reduktion des Hashwertes aus → 88
3. Man berechnet den Hashfunktion von 88 → 0768
4. Man sucht den Wert 0768 in der Tabelle → Er ist nicht in der Tabelle
5. Man führt eine Reduktion des Hashwertes aus → 68
6. Man berechnet den Hashwert von 68 → 0048
7. Man sucht den Wert 0048 in der Tabelle → Zeile 03
8. Man berechnet den Hashwert von 03 → 1708
9. Man führt die Reduktion des Hashwertes aus → 08
10. Man berechnet den Hashwert von 08 → 1888 → **Die gesuchte Zahl** → Das gesuchte Passwort ist ***08***

1.12 Gegen welche Angriffe hilft Salz? Begründen Sie Ihre Antwort

Das Salzen eines Passwortes hilft gegen Lookup- und Rainbow-Tables. Es hilft, sobald eine Angriffsmethode darauf basiert, dass einen Hash zu einem Passwort mappen kann. Dies ist sowohl bei Rainbow- wie auch bei Lookup-Tabellen der Fall. Wie im vorherigen Abschnitt zu sehen ist, werden in der Rainbow-Table die Hashes der ungesalzenen Passwörter verwendet.

Salzen ist jedoch ziemlich nutzlos bei Brute-Force Attacken, da diese jeden Hash zur Laufzeit berechnen und somit das Salz, wie das Betriebssystem auch, einfach in die Rechnung miteinbeziehen können.

1.13 Nennen Sie ein Betriebssystem, das seine Passwörter salzt und eines, dass sie nicht salzt

Salzen: Linux, MacOS, Solaris

Nicht salzen: Windows

1.14 Definieren Sie eine Passwort-Policy fuer eine fiktive Firma und begründen Sie die von Ihnen verlangten Regeln in Bezug auf die verschiedenen Angriffsmethoden

Passwortlänge
Min. 10 Zeichen. Verhindert / Erschwert Brute-Force Attacken

Passwortkomplexität
Min. 1x Grossbuchstaben, 1x Kleinbuchstaben, 1x Ziffer, 1x Sonderzeichen. Verhindert / Erschwert Brute-Force Attacken sowie Dictionary Attacks.

Blacklist
Gewisse Passwörter wie Pa$$w0rd oder auch solche, die in gewissen Top-Listen von beliebten Passwörtern sind, sind nicht erlaubt. Ebenso solche, die den Namen, das Geburtstag oder sonstige persönliche Informationen des Mitarbeiter und/oder der Firma enthalten. Dies verhindert/erschwert Lookup- und Rainbow-Table Angriffe, sowie auch Dictionary Attacks.

Passwortdauer
Die Passwortdauer sollte nicht zu kurz gewählt werden. Ansonsten beginnen die Mitarbeiter, sich die Passwörter aufzuschreiben, einfachere oder kürzere Passwörter zu wählen oder sie recyclen einfach ihre alten Passwörter (von Hunter2 zu Hunter3)

1.15 Was ist ein Man-in-the-Middle (MITM) Angriff?

Wie der Name das schon vermuten lässt, werden die Nachrichten auf dem Weg vom Sender zum Empfängen von einem Man in the Middle abgefangen, gelesen und evtl. manipuliert. Die Antwort des Empfängers fängt er ebenfalls ab und manipuliert sie evtl. wieder, bevor sie zurück zum Sender kommt.

Somit kann sowohl dem Sender, wie auch dem Empfänger vorgegaukelt werden, sie würden auf einem sicheren Kanal miteinander kommunizieren. Keiner der beiden weiss, dass sich noch ein dritter Teilnehmer in ihre Kommunikation eingeklinkt hat.

Es wird unterschieden zwischen Software- und Hardware-Attacken. Eine Software-Attacke kann z.B. ein Keylogger sein. Eine Hardware-Attacke ist z.B. ein Fake-Tippfeld bei einem Bankautomaten, welches über das echte Tippfeld gelegt wird, um den Karten-Chip sowie den dazugehörigen PIN auszulesen.

1.16 Nenn Sie je ein Beispiel von Software- und Hardware-Attacken

Software: Keylogger

Hardware: Tastatur-Overlay

1.17 Zeichnen Sie eine Phishing-Attacke auf ein E-Banking System mit TAN-Generator auf

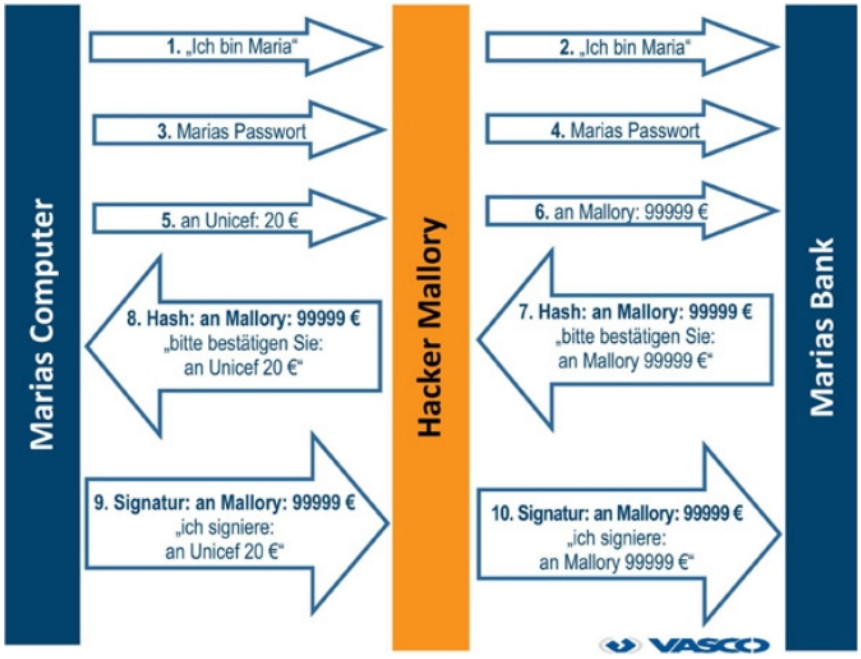

Abb. 1: Man in the Middle Attacke

1.18 Welche Vorteile hat ein TAN Generator gegenüber einer Streichliste?

Ein TAN Generator generiert einen one-time Code zur Laufzeit, während dem eine Streichliste beliebig viele Male verwendet werden kann. Somit kann eine Streichliste einfacher gestohlen werden wie ein ein TAN-Generator.

Falls der TAN-Generator zusätzlich zum Code noch weitere Informationen verschickt wie z.B. den Betrag und Empfängername oder -ID, können MITM-Attacken recht wirksam bekämpft werden.

1.19 Wie könnte eine Phishing-Attacke mittels Pharming und einem Pineapple-Device aussehen?

Pineapple Device
Der Wi-Fi Pineapple (Abb. 2) ist ein Router / WiFi Access Point, der gratis WiFi anbietet, aber für Pentesting / Auditing ausgelegt ist. Er enthält eine Software, die MITM-Attacken ausführen kann, um den Wireless-Verkehr zu sniffen, auditen und manipulieren.

Abb. 2: WiFi Pineapple Device

Pharming
Pharming beschreibt einen Angriff auf DNS-Abfragen. Wenn der Benutzer z.B. e-finance.postfinance.ch aufruft, so erhält er nicht die korrekte IP, sondern die IP einer Phishing-Site.

Um dies zu erreichen, gibt es verschiedene Methoden:

Angriff auf DNS: Mittels DNS Spoofing (DNS Cache Poisoning) kann die falsche IP direkt im DNS hinterlegt werden → Sehr schwer

Angriff auf die Host-Datei: z.B. Mittels Malware kann direkt im lokalen Hostfile, das vor dem DNS durchsucht wird, einen Verweis von e-finance.postfinance.ch auf die gewünschte IP gemacht werden → Bedingt Windows-Betriebssystem

Netzwerk-Router: Es kann im lokalen Netzwerk-Router ein entsprechender Verweis von der Domain auf die gewünschte IP gemacht werden → WiFi Pineapple Device ist der Netzwerk-Router, also einfach und plattformunabhängig.

Vorgehen
Man setzt einen sog. "Honey-Pot" auf. Indem man gratis und ungeschütztes Internet anbietet (z.B. mit der SSID "public"), verlockt man Leute dazu auf das Netzwerk zu verbinden. Von nun an geht jede Anfrage die der Benutzer stellt über den Netzwerk-Router (das WiFi-Pineapple), über den ich totale Kontrolle habe. So kann ich, wie bereits erwähnt, jede beliebige Anfrage auf jede beliebige IP umleiten mithilfe des Router-internen DNS'. Der Nutzer merkt davon nichts, ausser er achtet sich auf das SSL-Zertifikat oben links in der Adressliste.

1.20 Erklären Sie SQL Injection Angriffe anhand eines Beispiels

SQL-Injection bedeutet, dass der Benutzer durch unvalidierte Benutzereingaben Schadcode auf der Datenbank einschleusen ('injecten') kann. Dies geschieht dann, wenn Benutzereingaben ungeprüft direkt in die Datenbank geladen werden.

Beispiel

```
<?php
   $sql = "SELECT ID
         FROM USER
         WHERE Benutzer = '".$_POST['Benutzername']."' AND
               Passwort = '".$_POST['Passwort']."'
         ";
   $result = mysql_query($sql);
?>
```

Listing 1: PHP Code mit unvalidiertem SQL-Code

Der oben gezeigte Code ist anfällig für eine einfache SQL-Injection. Ein Benutzer könnten nun als Passwort `'' OR '1' = '1'` eingeben und könnte sich problemlos einloggen. Denn auf PHP-Level würde dieser Loginversuch so aussehen:

```
<?php
   $sql = "SELECT ID
         FROM USER
         WHERE   Benutzer = 'Admin' AND
               Passwort = '' OR '1' = '1'
         ";
   $result = mysql_query($sql);
?>
```

Somit gibt die Datenbank alle Benutzer zurück, die den Benutzernamen 'Admin' haben. Der Passwort-Check wurde komplett ausgehebelt, da nun gecheckt wird, ob 1 = 1 ist, was immer 'true' zurückgibt.

Mittles SQL-Injection kann auch fremder Code eingeschmuggelt werden. So könnte man in obenstehenden Szenario auch alle Benutzer löschen, wenn man den String `''; DROP TABLE USER` als Benutzernamen oder Passwort einsetzen würde. Denn ";" zählt als Abschluss eines SQL-Statements. Somit würde man z.B. einen leeren Benutzernamen übergeben und als nächstes, eingeschmuggeltes SQL-Statement die Tabelle 'USER' löschen.

1.21 Mit welchen Methoden können SQL-Injection Angriffe verhindert/erschwert werden?

Mittels Prepared Statements. Das SQL-Statement wird bereits vordefiniert und die Benutzereingaben werden quasi nur noch als Variablen übergeben.

```
<?php
   // Create connection
   $conn = new mysqli($servername, $username, $password, $dbname);

   // Check connection
   if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
   }

   // prepare and bind
   $stmt = $conn->prepare("SELECT ID FROM USER WHERE Benutzer=user AND
       Password=pass) VALUES (?, ?)");
   $stmt->bind_param("ss", $user, $pass);

   //set parameters to userinput
   $user = $_POST['Benutzername'];
   $pass = $_POST['Passwort']

   //execute statement
   $stmt->execute();
?>
```

Listing 2: Prepared Statement in PHP

Trotz Prepared Statement, welche SQL-Injection ziemlich ausser Gefecht setzt, sollte kein User-Input unvalidiert zur Datenbank gelangen. Es kann immer noch z.B. mit JavaScript schadhaften Code injiziert werden, der beim nächsten Aufruf der Webiste ausgeführt werden kann.

2 Teil Portmann

3 Teil Bürgler