Prüfungsprotokoll

Diplomprüfung im Vertiefungsgebiet: VVS1, VVS2, Datenkommunikation und Database Nation (Buch)

Datum: 27.10.2006
Prüfer: Professor Freiling
Beisitzer: Thorsten Holz
Prüfling: Christian Gorecki
Dauer: 45 min.
Note: 1.0

Die Atmosphäre während der Prüfung war sehr angenehm und auch ich kann Professor Freiling als Prüfer nur empfehlen.
Der Datenkommunikationsteil erscheint beim Betrachten meines Protokolls recht kurz, allerdings habe ich auch schon während des VVS1–Teils immer wieder versucht mein Wissen über Datenkommunikation an geeigneten Stellen einfließen zu lassen, so dass ich unter anderem auch Themen wie Acknowledge– und Sequenz–Nummern und ähnliches von alleine angesprochen haben. Ob das Einfluss auf die Länge des eigentlichen Datenkommunikationsteils hatte, kann ich nicht beurteilen.
Der Form halber möchte ich darauf hinweisen, dass es sich bei dem nachfolgendem Protokoll um ein Gedächtnisprotokoll handelt und ich somit weder Vollständigkeit noch Korrektheit garantieren kann. Während ich das Protokoll geschrieben habe war ich mir bei einigen Fragen sogar unsicher, ob mir die nun von Professor Freiling gestellt wurden, oder ob mir diese nur bei der Prüfungsvorbereitung untergekommen sind. So oder so sollte das nachfolgende Gedächtnisprotokoll bei der Vorbereitung auf eine entsprechende Prüfung hoffentlich hilfreich sein. Viel Erfolg!

Womit möchten Sie denn gerne anfangen?
*Am liebsten mit CHT oder FLP aus **VVS 2**.*

Und welchen von beiden würden Sie lieber vorführen?
Mmhh, mir eigentlich egal, vielleicht FLP? Da CHT ja schon in der vorangegangenen Prüfung vorgeführt wurde, wäre das doch eine nette Abwechslung.

Ok, dann FLP
Aber nur wenn der nicht als leichter gilt (im Vergleich zu CHT), sonst mach ich

lieber CHT.

Nein, die sind beide gleich schwer (lacht). Ok, nun haben wir in der Vorlesung FLP ja nicht direkt bewiesen, sondern...
In der Vorlesung haben wir das Völzer Theorem bewiesen, welches besagt, dass es für jeden Pseudo–Consensus Algorithmus einen unendlichen Lauf gibt (keine Termination). Beim Pseudo–Consensus ist das nicht weiter tragisch, da der ja nur aus jeder vorkommenden Konfiguration heraus terminierbar sein muss (man könnte wohl auch sagen: wir können jederzeit eingreifen und somit den Algorithmus terminieren lassen). Da aber jeder Algorithmus, der Consensus löst, auch Pseudo–Consensus löst, haben wir genau diesen, im Beweis zum Völzer Theorem gefundenen, nicht terminierenden Lauf auch beim Consensus Algorithmus. Daraus folgt dann FLP.
Also beweise ich nun das Völzer Theorem. (Beweis des Völzer Theorems und Eingehen auf die Zwischenfrage von Professor Freiling, wo denn beim CHT-Play so etwas wie eine non–uniforme Anfangskonfiguration vorkommt. Hier habe ich das Tagging und die Stabilisierung erläutert und habe auch noch versucht die beiden Fälle: Übergang von 0–uniform zu 1–uniform und 0–uniform zu bi–uniform zu erläutern. In letzterem Fall habe ich dann nur erwähnt, dass man das mit einem Algorithmus grafisch lösen kann, indem man nach Hooks und Forks sucht, und dass man zeigen kann, dass dieser Algorithmus terminiert und auch, dass wenn er terminiert, dass dann ein Hook oder ein Fork gefunden wurde.)

Ok, Sie haben ja gerade auch Pseudo–Consensus angesprochen, und gesagt, dass es da einen unendlichen Lauf geben kann. Wir hatten ja in der Vorlesung einen Algorithmus, der Pseudo–Consensus löst, besprochen. Könnten Sie diesen Algorithmus einmal erläutern und auch zeigen, wie es da zu einem unendlichen Lauf kommen kann?
Klar! (Algorithmus erklärt und gezeigt wie es zu einem unendlichen Lauf kommen kann)

Kann man dies mit einem Perfect–Failure Detector verhindern, also Terminierung garantieren?
Ja! Dafür muss man dann allerdings den Algorithmus ein wenig ändern. (Notwendige Änderungen erklärt. Idee: Nicht einfach auf eine Change–Anweisung eines Nachbarn reagieren, wenn der andere Nachbar noch am leben ist [an dieser Stelle benutzt man den Perfect–Failure Detector], sondern erst abwarten, was dieser für einen Wert vorschlägt)

Reicht für diesen Algorithmus auch der Failure Detector “Omega” ?
Ja, denn Omega ist äquivalent zum “eventually Strong” Failure–Detector und dafür kann man recht einfach (wenn man sich vorher schon überlegt hat, wie das geht ;-) zeigen, dass das geht.

Ja, das ist die einfache Variante. Aber man könnte das auch direkt mit Omega implementieren.

Ok, dann versuche ich das mal, und wenn es nicht klappt, dann zeige ich es über die Äquivalenz zum "eventually strong" Failure–Detector. (Mit ein bisschen Überlegen und Denkanstößen von Professor Freiling bin ich dann auf einen Algorithmus gekommen, der mit "Omega" als Failure–Detector auskommt.)

Ok, kommen wir dann zu **VVS1**. Wenn man nun ein System schützen möchte, wie geht man da zum Beispiel vor.
Zunächst einmal muss man sich überlegen, gegen wen man sich schützen möchte, ...

Da hatten wir so eine Klassifizierung von Angreifern...
(Hier habe ich die Skizze zur Klassifizierung von Spionen, Crackern, Hackern und Vandalen aufgemalt und versucht die Relationen zu erklären, bin dann aber nicht auf den Grund gekommen, warum Spione gefährlicher sind als Cracker [Der Grund ist: Spione haben wesentlich größere Resourcen, da sie oft für Geheimdienste arbeiten])

Was kann man denn gegen die Verwendung schwacher Passwörter tun?
Eine entsprechende Passwort–Policy erstellen und nur sichere Passwörter zulassen. Dabei können gewählte Passwörter mit entsprechenden Tools überprüft werden.

Welche Mechanismen gibt es denn, um eine gesicherte Authentifizierung zu gewärleisten?
Passwörter sollten nicht im Klartext abgespeichert werden. Statt dessen kann man einen Hash über diese bilden, zum Beispiel durch Verschlüsseln mit einer abgewandelten Form des DES. Das entsprechende Kryptogramm wird dann auf der Festplatte (bei Linux zum Beispiel in der /etc/shadow) abgelegt. Bei der Benutzeranmeldung wird dann das gleiche mit dem vom Benutzer eingegebenen Passwort gemacht und das Resultat dann mit dem auf gespeicherten Kryptogramm verglichen. Bei einer Übereinstimmung ist der Benutzer erfolgreich authentifiziert.

Nun würden aber zwei gleich Passwörter das gleiche Kryptogramm ergeben ...
Darum verwendet man bei dem Beschriebenen Verfahren ein sogenanntes Salt, welches auch bei gleichen Passwörtern zu einem unterschiedlichen Kryptogramm führt. Auf diese Weise wird einem Angriff mit Hilfe der Rainbow Tables (welche vorberechnete Hashes von Passwörtern erhalten und somit das Nachschlagen der zugehörigen Passwörter zu einem gegebenen Hash–Wert ermöglichen) vorgebeugt.

Warum ist es denn bei Windows oft recht einfach Passwörter zu brechen?
(Hier habe ich dann die unsicheren LM–Hashes erwähnt und dass diese auch bei neueren Windows Versionen mit sichererem NT–Hash nachwievor, der Abwärtskompatibilität halber, verwendet werden. Auf die Frage warum die LM–Hashes soviel unsicherer sind habe ich in dem Moment keine Antwort gewusst. Als er dann die Konvertierung in Großbuchstaben erwähnte, konnte ich mich da auch wieder dran erinnern. Das aber beim LM–Hash auch das Passwort in zwei Teile aufgeteilt wird und dann von beiden Teilen seperat der Hash berechnet wird, muss ich wohl beim lernen überlesen haben.)

Was ist mit Spoofing?
Da hatten wir DNS–Spoofing, ARP–Spoofing und IP–Spoofing. (Hier habe ich alle drei Spoofing Arten erklärt und erwähnt, dass ARP–Spoofing nur in einem lokalen Netzwerk Sinn macht, wohingegen man Netzwerkübergreifend eher auf IP–Spoofing zurückgreift. Auch bin ich hier auf die Aushebelung der Sicherheitsvorteile eines Switches gegenüber eines Hubs durch ARP–Spoofing eingegangen. Ein wenig Datenkommunikation kam hier auch schon vor.)

Können Sie ein Beispiel geben, wofür man Spoofing verwenden kann?
Zum Beispiel um eine bestehende Verbindung zu beenden oder zu übernehmen (Hijacking). Dabei könnte man sich zunächst mittels ARP–Spoofing zwischen die Kommunikation zweier Rechner schalten dann die Sequenznummern beobachten und somit dann ein Reset mit richtig "geratener" Sequenznummer an den Client schicken und gegebenenfalls weitere Pakete mit entsprechenden Sequenznummern an den Server senden, wenn man die Verbindung übernehmen möchte.

Wir hatten ja auch vorgestellt, wie man Spoofing verwenden kann, um einen Port–Scan zu verschleiern ...
Ja, das war beim Decoy–Scan ... (Hier habe ich dann zunächst den einfachen Port–Scan und dann Decoy– und Idle– Scan erklärt. So kamen wir dann auf das IP–ID Feld im IP–Header zu sprechen und hatten damit den Übergang zu ***Datenkommunikation****.)*

Wofür ist denn das IP–ID Feld eigentlich gedacht?
Um bei Fragmentierung erkennen zu können, welche IP–Pakete Fragmente eines Payloads bzw. unterschiedliche Payloads enthalten. (Das habe ich dann noch ein bisschen ausführlicher erklärt)

(Wenn ich mich recht erinner hat Professor Freiling noch ein paar andere Fragen zu Datenkommunikation gestellt, aber da kann ich mich leider nicht mehr so genau dran erinnern)

Ok, dann kommen wir zum Abschluss noch zu **Database Nation**: Was kann man denn jemandem entgegnen, der sagt, er hätte nichts zu verbergen und somit keine Einwände gegen die tägliche Observation der wir ausgesetzt sind?
Nun ja, viele Informationen, die über uns gesammelt werden, scheinen in der Tat zunächst einmal von keinerlei Bedeutung und es kann mir ja im Grunde egal sein, ob jemand weiß, was ich wo einkaufe. Das Problem ist, dass wir uns in der Regel nicht darüber im Klaren sind, welche Macht sich aus der Zusammenführung all dieser Informationen über uns ergeben kann. Ein gutes Beispiel hierfür ist "Social Engineering". Hierbei werden viele kleine zunächst "unwichtige" Informationen gesammelt, die dann genutzt werden, um das Vertrauen des Opfers zu gewinnen. Das Resultat: Wir vertrauen jemandem der nicht vertrauenswürdig ist (Stichworte: trusted vs. trustworthy). Dieses Vertrauen kann dann folgenschwer ausgenutzt werden.
Ein weiteres Problem ist die schnelle, nur schwer zu kontrollierende Propagierung

von Fehlern in der Inter–Kommunikation zwischen verschiedenen Datenbanken. Das Buch liefert einige Beispiele bei denen ganze Existenzen zerstört, oder zumindest nachhaltig geschädigt wurden. (An dieser Stelle ergab sich dann eine Art Diskussion über Themen, die in dem Buch behandelt wurden.)

Ok, das sollte reichen, gehen Sie dann doch bitte einmal vor die Tür