Prüfungsprotokoll
Praktische Informatik
Faecher: VVS1, VVS2, Datenkommunikation, Buch:Database Nation
Pruefling: Franziska Roloff
Pruefer: Professor Freiling geb. Gärtner
27.10.2006
Note: 1,0

Womit wollen Sie anfangen?
Am besten Standardprogramm, VVS2, CHT

Gut, dann erklären Sie doch mal, worum es da geht. Zuerst vielleicht was consensus ist und was ein failure detector ist und was da dann bei CHT gemacht wird.
(Erklärt, was consensus ist, was ein failure detector ist, dass man damit Synchronisation „angeben" kann. Dann CHT erklärt, mit allem, was dazu gehört. Auch hooks und forks vom Appendix angesprochen, aber nichts Genaues dazu gesagt, nur, dass man auch im Fall der Bivalenz des Zustands mit dem kritischen Index ein korrekter Zustand identifiziert werden kann, nur dass es dann ein anderer ist usw.)

Ok, wir hatten ja auch noch eine andere Art von consensus, nämlich Pseudo-Consensus. Erklären sie doch mal, was das ist, und was wir da in der Vorlesung für ein Beispiel hatten.
Erklärt, was Pseudo-Consensus ist, das Problem mit den drei Prozessen und dann es eben zu einem unendlichen Lauf kommen kann bei ungünstigem scheduling, weil dann einer seinen Wert ändert auf Grund eines „change"-Signals von links, aber selber ein change wegschickt und so weiter.

Was für einen failure detector braucht man denn für Pseudo-Consensus?
Gar keinen. Man kann ja jeden endlichen Lauf so erweitern, dass Terminierung erfüllt ist, nämlich indem man einen Prozess „einfriert".

Gut, und wie würde man das machen, wenn man einen perfect failure detector hätte?
Das Problem ist ja nicht, wenn ein Prozess abstürzt, sondern dass man nicht weiß, ob einer abgestürzt ist. Wenn man das also weiß mit Hilfe des perfect failure detector, dann kann man, wenn der rechte Nachbar noch lebt, auf seinen Wert warten, bevor man seinen eigenen Wert ändert. Dann hätte man hier auch consensus gelöst mit dem Algorithmus.

Ok, reicht denn dann auch ein eventually perfect failure detector?
Ja, der reicht auch. Irgendwann werden ja die korrekten nicht mehr verdächtigt, das heißt dann, dass irgendwann auf den Wert vom rechten Nachbarn gewartet wird und dann Terminiert der Algorithmus.

Reicht dann vielleicht auch ein eventually strong failure detector?
Ja, der reicht sogar auch. Dann wird ja einer, der „holy guy", irgendwann nicht mehr falsch verdächtigt. Das heißt, dann aus dessen Wert gewartet wird. Und dann terminiert der Algorithmus.

Sehr gut, dann gehen wir mal zu VVS1 über.
Da gibt es ja verschiedene Möglichkeiten, so ein System zu kompromittieren. Zum Beispiel die sogenannten Race Conditions. Was war denn das und wozu haben wir das benutzt?
(Erklärt, was das ist, dann mit dem Beispiel aus der Vorlesung vertan. Professor Freiling hat dann erzählt, was das Beispiel in der Vorlesung war: passwd öffnen.)

Was kann man denn dann machen, wenn man in die passwd mit Schreibrechten reinkommt?
Benutzer anlegen…

Ok, dann hatten wir ja auch noch buffer overflow. Was macht man denn da und wie kann man dadurch was erreichen?
Relativ allgemein erklär, was das ist, Rücksprungadresse überschreiben, neue raten, noop-Landebahn, eingeschleusten code ausführen.

(malt code auf) Ok, also wie genau? Hier haben wir ein Stück c-code, was wird denn wo in den Speicher geschrieben?
(kurz erklärt)

(malt den stack auf) Also, welchen Speicher muss man denn jetzt überschreiben?
Den buffer, da, wo die lokalen Variablen drin stehen. Nicht den, wo die Parameter drin stehen. Dadurch überschreibt man von unten nach oben die Rücksprungadresse, springt „unten“ in den buffer und führt den code aus, mit dem man die Rücksprungadresse überschrieben hat. Damit man die Rücksprungadresse nicht ganz genau raten muss, macht man sich eine sogenannte „noop-Landebahn“, damit man da irgendwoe reinspringen kann.

Ok, dann hatten wir auch noch das sogenannte suid bit. Was ist denn das?
Programme, bei denen das suid-Bit gesetzt ist, werden mit den Rechten des Besitzers aufgerufen, nicht mit den Rechten des Startenden. Das kann man als Angreifer ausnutzen, indem man Programme mit gesetztem suid root bit sucht und diese ausführt, er möglichst schafft, mit Hilfe dieser Programme eine shell zu starten.

Wie kann man denn die Macht von root einschränken?
Tja, da fallen mir jetzt nur diese security level ein. Aber da kenne ich mich nicht mit aus. Ich weiß nur, dass es das gibt und dass man, um von einem ins andere security level zu kommen, neu starten müsste.

Ok, gut.
Kommen wir zum sogenannten „Spoofing“. Was ist denn das?
Das ist wenn man sich als jemand anders ausgibt. Man schickt also jemandem ein Paket mit einer gefälschten Absenderadresse.

Es gibt ja zum Beispiel das DNS-Spoofing. Wie läuft das?
Wenn jemand einen DNS-Server kontaktieren will und diesen fragt, welches die IP-Adresse des symbolischen Namens xyz ist, dann schicke ich ihm eine Antwort und leite ihn so auf einen falschen Server um.

Wo ist denn dann da genau das Spoofing?
Gute Frage, das hatte ich mit auch schon mal überlegt, aber nicht irklich zu Ende gedacht.
(malt das Szenario auf)
Ahja, wenn ich dem Anfragenden als Antwort die IP-Adresse eines falschen Servers nenne, dann ist das nur dann glaubwürdig, wenn diese von der IP-Adresse des DNS-Servers kommt. Deshalb muss ich dieses Paket mit der Absenderadresse des DNS-Servers spoofen.

Mit welchem Protokoll wird denn eigentlich der DNS-Server angesprochen?
Geht da nicht beides? TCP und UDP?

Ja, theoretisch schon. Aber wie wird es wirklich gemacht?
Mit TCP?

Naja, wir wollen ja schnell sein und haben nur ne kleine Anfrage…
Ok, dann wohl UDP ☺

Wie geht denn so ein TCP-Verbindungsaufbau?
Mit dem sogenannten 3-way-handshake…

Und wie wir dann der weitere Verkehr gehandhabt?
Da gibt es zum Beispiel Sequenznummern, Acknowledgement-Nummern und so…
Außerdem wird auch eine buffersize übertragen, die dem Sendenden angibt, wie viel er senden kann…

Wie viele Bits werden für die windowsize benutzt?
Ich rate, 16?

Mmh, das weiß ich jetzt auch nicht so…

Ok, dann noch ein paar Fragen zu Database Nation:
Was würden sie denn jemandem antworten, der sagt, er hätte nichts zu verbergen, man könne ihn ruhig abhören etc.
Social engineering, wie das mit der ssn in den USA ist und das kleine Infos oftmals viel mehr Wert sind, als man denkt…
Im weiteren noch eine kleine Unterhaltung über Beispiele im Buch, Schufa, …

Zusammenfassend muss ich sagen, dass die Prüfung wirklich sehr angenehm war. Auch wenn zwei, drei Sachen drankamen, mit denen ich nie gerechnet hätte und mit denen ich nicht besonders sicher war, habe ich nie irgendwie ein unwohles Gefühl gehabt. Professor Freiling lächelt einen eigentlich die ganze Zeit an und man hat das Gefühl, dass es ihm Spaß macht und dass er sich mit einem mehr unterhalten möchte, als dass er versucht, was zu finden, was man nicht kann oder so.
Ich kann Professor Freiling als Pruefer nur empfehlen und wünsche allen, die sich auch beim ihm prüfen lassen, viel Erfolg!