

# Kryptografisch gesicherte Datenübertragung bei NIS und NFS durch SSH – Tunneling

Prof. Dr. Christian Müller

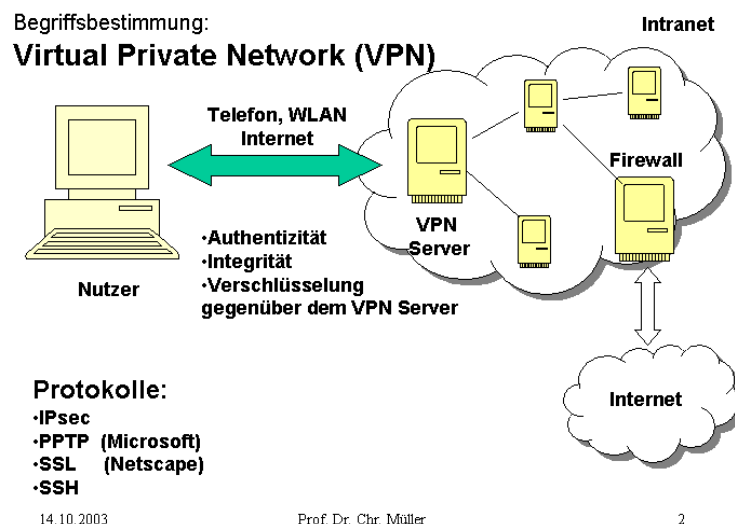
TFH Wildau im Oktober 2003

1. Zusammenfassung
2. Begriffsbestimmung und Einleitung
3. Der Tunneling Ansatz von Trapp für Remote Procedure Calls
4. Das Secure Network File System (SNFS)
5. Das Secure Network Information System (SNIS)
6. Alternativen zu SNFS und SNIS
7. Literatur

## 1. Zusammenfassung:

In UNIX basierten Netzwerken werden häufig das Network File System (NFS) als verteiltes Dateisystem und das Network Information System (NIS) als Verzeichnisdienst eingesetzt. Beide Dienste nutzen Remote Procedure Calls. Die Daten dieser Dienste werden unverschlüsselt im Netzwerk übertragen und können damit das Ziel vielfältiger Angriffe in Netzwerken sein. Besonders kritisch ist die Situation bei dem Network Information System, da dort Passwörter ungeschützt im Netzwerk übertragen werden [Hess 92]. Aufbauend auf einer Arbeit von Trapp [Trapp 96] entwickelte Bowman [Bowman 02] aus dem Network File System (NFS) das Secure Network File System (SNFS). Mit ähnlichen Techniken entwickelte der Autor [Müller 03] aus dem Network Information System (NIS) das Secure Network Information System (SNIS). SNFS und SNIS sind unter der GPL verfügbar [Bowman 03] und werden zur Zeit von Debian evaluiert [Deb-dev 03].

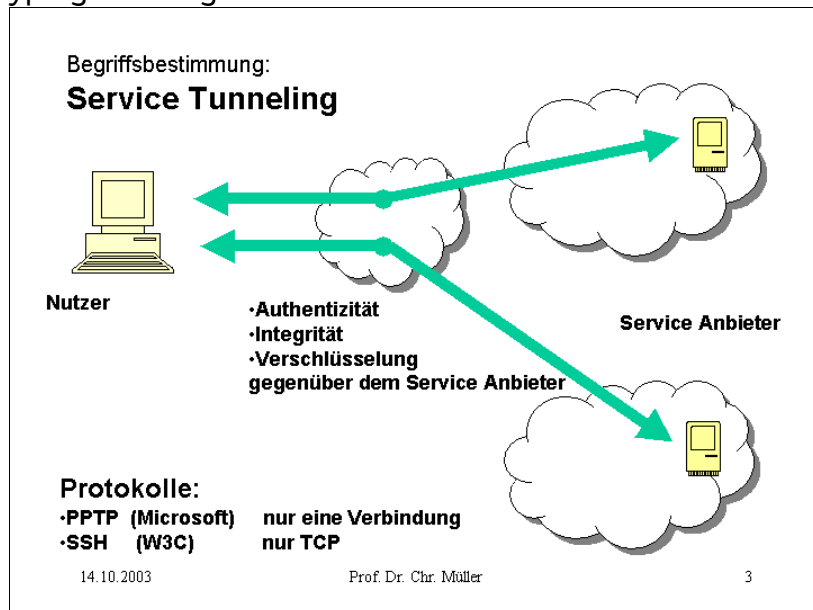
## 2. Begriffsbestimmungen und Einleitung:



Viele Unternehmen möchten ihre Unternehmensnetzwerke (Intranets) gegen externe Netze absichern. Diese Aufgabe übernehmen Firewall Rechner, welche die Zulässigkeit aller Verbindungswünsche zwischen internen und externen Rechnern kontrollieren. Eine Möglichkeit mehrere Intranets über ein öffentliches Netz zu verbinden bieten „Virtual Private Networks“ (VPN). [Bird 03] Dabei werden

alle Daten, die über öffentliche Leitungen übertragen werden, verschlüsselt. Mit den Verschlüsselungsverfahren wird gleichzeitig die Authentizität und Integrität der Kommunikationspartner sichergestellt. Das gleiche gilt, wenn einzelne externe Rechner über ein öffentliches Netz in das Unternehmensnetz integriert werden sollen. Möchte ein Nutzer an einem externen Rechner einen Service innerhalb des Intranets nutzen, so muss er eine kryptografisch gesicherte VPN Verbindung zwischen seinem Rechner und dem VPN Server des Intranets aufbauen. Dadurch wird der Nutzerrechner zu einem Teil des Intranets. Der VPN Ansatz ist ein hierarchischer Ansatz, da zu einem Zeitpunkt ein Rechner nur zu einem VPN gehören kann.

Mit dem Service Tunneling wird ein weniger restriktiver Weg beschritten. Hier werden kryptografisch gesicherte Kommunikationstunnel zwischen Client- und



Serveranwendungen erstellt. Für jeden zusichernden Kommunikationsweg muss ein solcher Tunnel erstellt werden. Die Authentizität wird bei dem Tunnelaufbau durch Passwörter oder Public Key Verfahren geprüft. Mit der Verschlüsselung wird die Integrität der im Tunnel übertragenen Daten sichergestellt.

Schlüssellängen für symmetrische Verfahren

Kosten in Dollar	40 Bit	56 Bit	64 Bit	80 Bit	112 Bit	128 Bit
1.000	4 min	145 Tg.	100 J.	7 Mio. J.	$10^{15}$ J.	$10^{21}$ J.
10.000	20 sec	14 Tg.	10 J.	700.000 J.	$10^{15}$ J.	$10^{20}$ J.
100.000	2 s	35 h	1 J.	70.000 J.	$10^{14}$ J.	$10^{19}$ J.
1 Mio.	200 ms	3,5 h	37 Tg.	7.000 J.	$10^{13}$ J.	$10^{18}$ J.
10 Mio.	20 ms	21 min	4 Tg.	700 J.	$10^{12}$ J.	$10^{17}$ J.
100 Mio.	2 ms	2 min	9 h	70 J.	$10^{11}$ J.	$10^{16}$ J.
1 Mrd.	200 $\mu$ s	13 s	1 h	7 J.	$10^{10}$ J.	$10^{15}$ J.
10 Mrd.	20 $\mu$ s	1 s	5,4 min	245 Tg.	$10^9$ J.	$10^{14}$ J.
100 Mrd.	2 $\mu$ s	100 ms	32 s	24 Tg.	100 Mio. J.	$10^{13}$ J.
1 Bio.	200 ns	10 ms	3 s	2,4 Tg.	10 Mio. J.	$10^{12}$ J.
10 Bio.	20 ns	1 ms	300 ms	6 h	1 Mio. J.	$10^{11}$ J.

Jahresbudget der National Security Agency (NSA)

Alter des Universums:  $10^{10}$  Jahre

Quelle: <http://www.rsasecurity.com/>

14.10.2003

Prof. Dr. Chr. Müller

4

Zur Übertragung von Massendaten, wie sie in einem VPN und beim Service Tunneling auftreten, werden normalerweise symmetrische Verschlüsselungsverfahren verwendet. Dabei ist beiden Kommunikationspartnern ein

gemeinsamer Schlüsselwert (Key) bekannt, der zum Ver- und Entschlüsseln verwendet wird. Die Sicherheit des Verfahrens ergibt sich aus dem Aufwand, der zum Entschlüsseln des Keys notwendig ist. Dies passiert durch das systematische Austesten aller zulässigen Schlüsselwerte. Der Aufwand steigt exponentiell mit der Schlüssellänge. Gemessen wird er durch die notwendige Rechenzeit auf Rechnern bestimmter Leistungsklassen. Die Grafik [RSA 00] zeigt den Zusammenhang zwischen Schlüssellänge, Leistungsstärke des Rechners zum Dechiffrieren (Kosten) und der für die Entschlüsselung notwendigen Zeit. Danach können z.Z. Schlüssellängen von mehr als 112 Bits als sicher betrachtet werden, da das Entschlüsseln auf aktuellen Hochleistungsrechnern mehr als  $10^{10}$  Jahren dauern würde. Dies entspricht in etwa dem Alter des Universums.

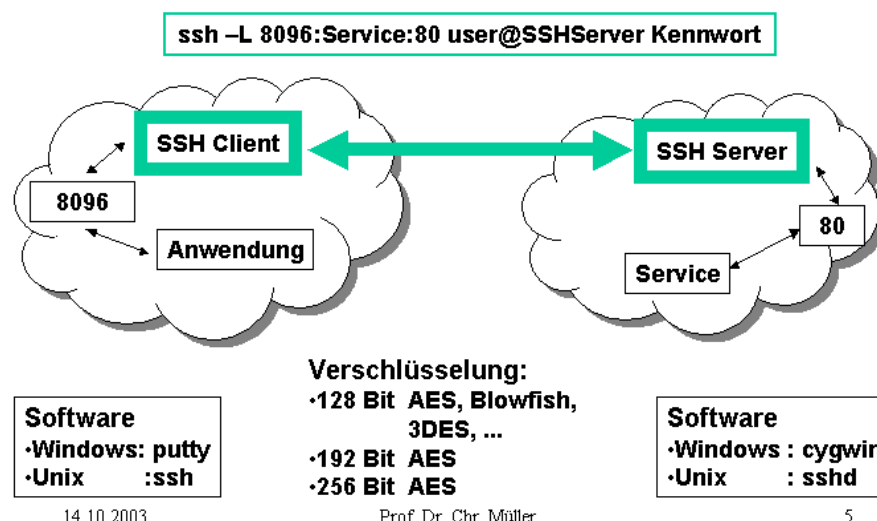
Die bei VPN und Service Tunneling verwendeten Verschlüsselungsverfahren sind in den verwendeten Protokollen festgelegt. Die Schlüssellängen können in einem gewissen Rahmen mit Parametern eingestellt werden.

Im weiteren soll ein SSH basiertes Porttunneling betrachtet werden. SSH steht für Secure Shell und ist ein in Unix Systemen standardmäßig enthaltender Client - Server Dienst. Mit Putty [Putty 03] und Cygwin [Cygwin 03] liegen inzwischen auch Portierungen für das Windows Betriebssystem vor. Das SSH Verfahren [Barret 02], [OpenSSH 03] unterstützt neben einem TCP Port Tunneling auch:

- die Arbeit mit einer entfernten Kommandoshell,
- die entfernte Kommandoausführung und
- den Filetransfer zwischen entfernten Rechnern.

Als Verschlüsselungsverfahren stehen zur Zeit 128 bis 256 Bit Verfahren zur Verfügung.

## SSH TCP-Port-Tunneling



14.10.2003

Prof. Dr. Chr. Müller

5

Mit dem SSH Client wird eine gesicherte Verbindung zu einem entfernten SSH Server aufgebaut. Außerdem kann der SSH Client einen Anwendungsdienst auf dem Client Rechner, der über einen freigewählten TCP Port erreichbar ist, simulieren. Alle Anfragen an diesen Port des Client Rechners werden von dem SSH Client an den SSH Server über einen gesicherten Kanal weitergeleitet. Der SSH Server übermittelt die Anfragen nun dem Zieldienst. Auf diese Weise

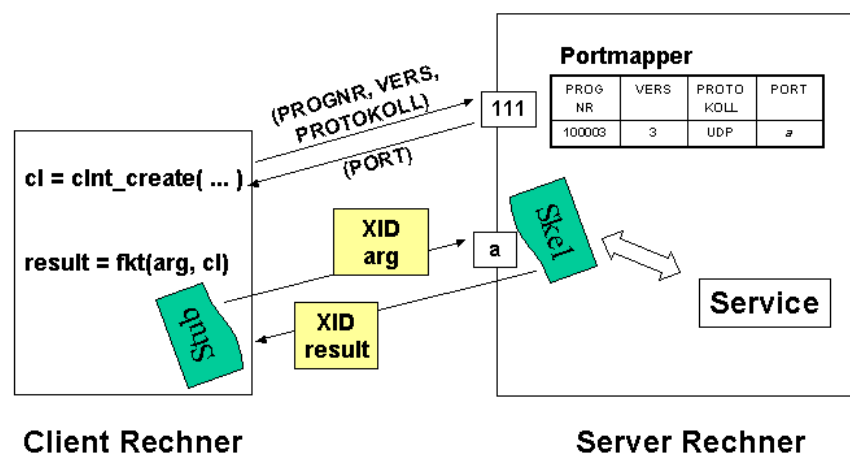
beschränkt sich jede ungesicherte Kommunikation lediglich auf den Client-, bzw. auf den Serverrechner. Die verwendeten Ports lassen sich durch eine geeignete Konfiguration vor einem unberechtigtem Zugriff von anderen Rechnern schützen.

Der SSH Standard unterstützt nur ein Porttunneling für das TCP Protokoll. Damit ist dieser Ansatz für einen gesicherten Zugriff auf WWW und Mail Dienste geeignet. Damit er auch für UDP Dienste wie NFS und NIS genutzt werden kann, sind einige Modifikationen nötig. Dabei wird ausgenutzt, dass sowohl NFS als auch NIS mit UDP Remote Procedure Calls arbeitet.

### 3. Der Tunneling Ansatz von Trapp für Remote Procedur Calls

Dienste wie NFS und NIS basieren auf Remote Procedure Calls (RPC) [Lisiecki 00]. Damit wird ein Verfahren bereitgestellt, welches Prozeduraufrufe auf einem Clientrechner ermöglicht, die auf einem entfernten Server Rechner ausgeführt werden. Ein Service stellt eine Familie von Funktionalitäten bereit, die ein Server als RPC's zur Verfügung stellt. Jedem Service ist eine Programmnummer sowie eine Versionsnummer zugeordnet. Ein Registrierungsdienst (Portmapper) ordnet jedem Service eine Portnummer zu. Die Clientanwendung erfragt von dem Portmapper des Serverrechners die Portnummer der benötigten Serviceanwendung. Mit dieser Portnummer erreicht die Clientanwendung den gewünschten Service auf dem entfernten Rechner.

#### Unix Remote Procedure Calls (RPC)



14.10.2003

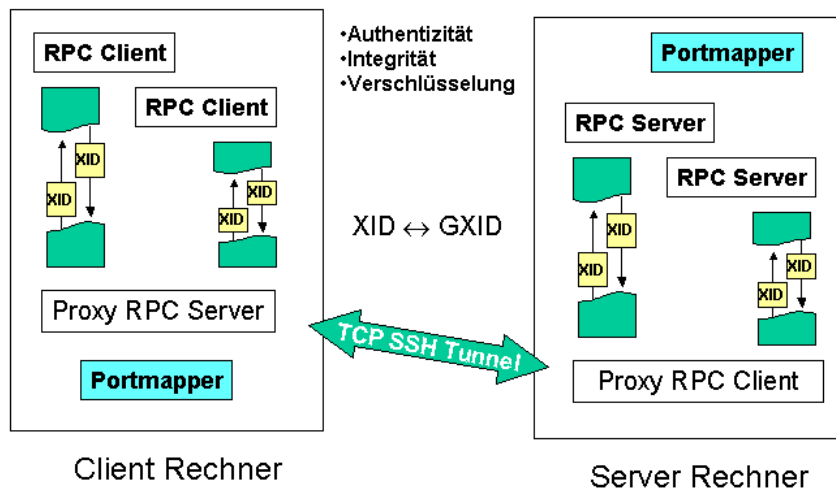
Prof. Dr. Chr. Müller

7

Die eigentliche Kommunikation für den entfernten Prozeduraufruf wird durch ein Stub auf der Clientseite und durch einen Skeleton auf dem Server realisiert. Anstelle der entfernten Prozedur ruft der Client den Stub auf. Der Stub sendet die Argumente des Prozeduraufrufs als ein UDP Paket zu dem Skeleton auf dem Serverrechner. Dieses befragt den eigentlichen Service und sendet die Antwort in Form eines UDP Pakets zurück zu dem Stub, der seinerseits die Antwort der Clientanwendung zur Verfügung stellt. Die UDP Pakete enthalten eine Identifikationsnummer (XID) mit der jeder Anfrage die passende Antwort zugeordnet wird. Die beteiligten Datenpakete sind unverschlüsselt. Auch eine Authentifizierungs- bzw. Integritätsprüfung ist nicht vorgesehen. In einer Reihe von Arbeiten wurden Ansätze zur beliebigen Manipulation solcher Dienste vorgestellt [Hess 92].

Mit dem RPC Tunneling Ansatz von Holger Trapp [Trapp 96] steht eine Möglichkeit für sichere RPC Calls zur Verfügung. Trapp nutzt dabei das SSH Verfahren, welches die benötigten Sicherheitseigenschaften bereitstellt. Dazu wird auf dem Client Rechner ein Proxy RPC Server installiert, der den gewünschten Dienst auf dem Clientrechner bereitstellt. Die Clientanwendung muss nun so konfiguriert werden, dass sie den Service direkt von dem Proxy RPC Server bezieht. So wird eine unsichere Kommunikation, die den Clientrechner verlässt vermieden.

### RPC Tunneling (Holger Trapp, TU-Chemnitz 1996)



14.10.2003

Prof. Dr. Chr. Müller

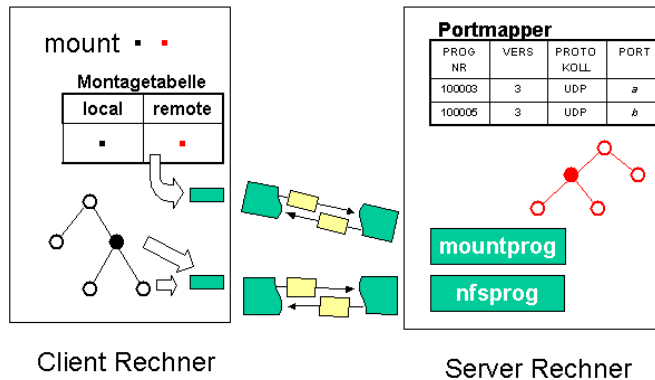
8

Der Proxy RPC Server leitet die Datenpakete über einen sicheren TCP SSH Tunnel zu einer Proxy RPC Clientanwendung auf dem entfernten Serverrechner weiter. Der Proxy Client auf dem Server Rechner kommuniziert nun auf klassischer Weise mit dem gewünschten RPC Server auf dem gleichen Rechner. Somit wird auch hier die ungeschützte RPC Kommunikation auf den Server Rechner beschränkt. Die Netzwerkfunktionen beider Rechner sind durch lokale Firewalls abzusichern. Auf diese Weise wird die Authentizität und Integrität der Netzwerkkommunikation durch das SSH Verfahren garantiert. Weiterhin muss allerdings die Vertrauenswürdigkeit der Anwendungen auf dem Client- und Server Rechner sichergestellt sein.

#### 4. Das Secure Network File System (SNFS)

Bei dem Netzwerk File System (NFS) [Kirch 00] handelt es sich um ein in UNIX Umgebungen verbreitetes Dateisystem, welches über mehrere Rechner verteilt ist. Auf einem Server Rechner werden Teile des lokalen Dateisystems zum Zugriff von entfernten Rechnern freigegeben. Der Client Rechner bindet einen solchen freigegebenen Teilbaum des Serverrechners virtuell in seinen lokalen Dateibaum ein. Dazu ist die Wurzel des freizugebenen Teilbaums einem lokalen Verzeichnis (Mountpoint) in der Montagetabelle zuzuordnen. Die Zulässigkeit dieser Operation wird mit einer RPC Anfrage beim Mount Programm des Server Rechners erfragt.

## Network File System (NFS)



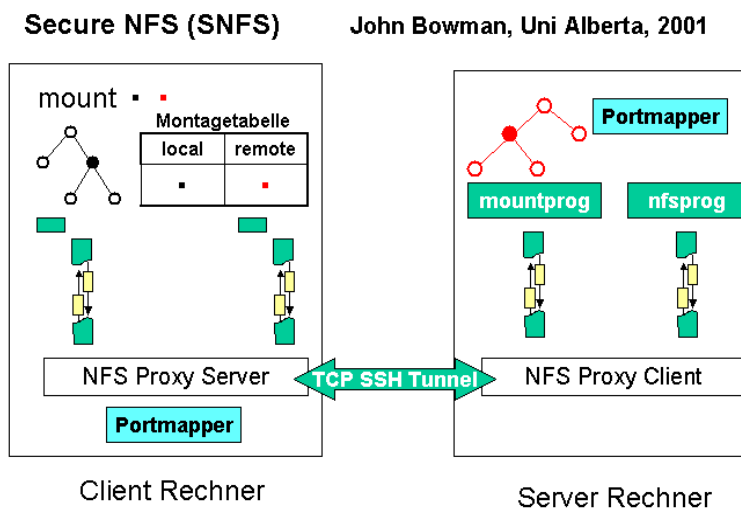
14.10.2003

Prof. Dr. Chr. Müller

9

Wenn der Client Rechner auf die entfernten Teile seines Dateisystems zugreift, dann werden die benötigten Informationen mit RPC Anfragen an das NFS Programm des Server Rechners beschafft. Auch diese Kommunikation findet standardmäßig vollkommen ungeschützt statt.

Dieses Sicherheitsloch beseitigte John Bowman [Bowman 02] indem er durch eine geschickte Konfiguration den RPC Tunneling Ansatz von Trapp mit dem klassischen NFS Verfahren kombinierte. Diese Kombination nannte er **Secure Network File System (SNFS)**. Dazu wird auf dem Client Rechner wieder ein Proxy Server installiert, der die lokalen NFS Anfragen entgegennimmt, um sie getunnelt zu einem Proxy Client auf dem Server Rechner weiterzuleiten. Bei der NFS Konfiguration ist darauf zu achten, dass die NFS Datenpakete ungefähr die gleiche Größe wie die UDP Pakete haben, damit durch das Tunneling nur geringe Performanceverluste eintreten. Solche Einstellungen sind allerdings erst seit NFS Version 3 möglich.



14.10.2003

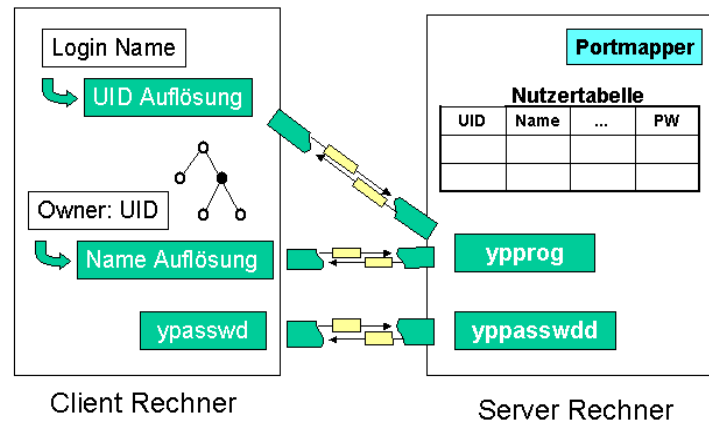
Prof. Dr. Chr. Müller

10

## 5. Das Secure Network Information System (SNIS)

In Unix Systemen werden zur Identifikation von Nutzern bzw. Nutzergruppen ID's in Form von Zahlen (UID bzw. GID) verwendet. In den lokalen Dateisystemen beschreiben diese ID's die Zugriffsberechtigten.

### Network Information System (NIS)



14.10.2003

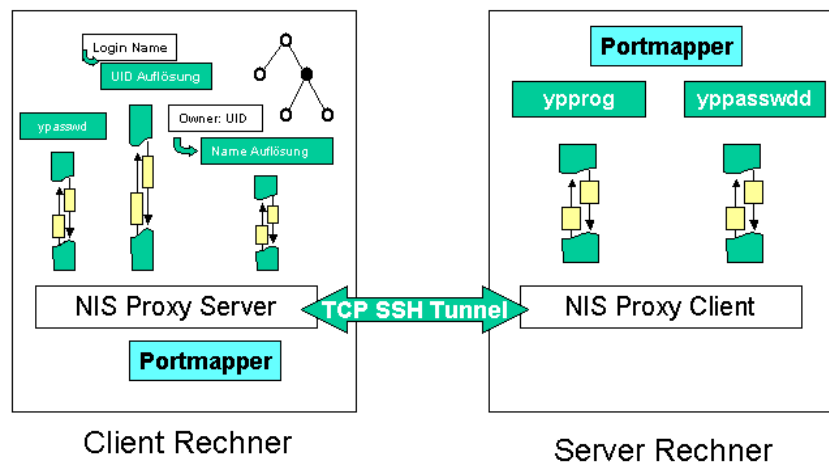
Prof. Dr. Chr. Müller

11

Das NFS System nutzt diese ID's auch auf den Client Rechnern. Dies ist jedoch nur sinnvoll, wenn auf allen Client Rechnern die gleichen ID's verwendet werden. Das Network Information System (NIS) [Kirch 00] ist ein RPC basierter Dienst, der alle benötigten Nutzerinformationen auf den Client Rechnern bereitstellt.

Mit diesem Dienst besorgt sich der Client Rechner die benötigten Nutzerinformationen bei dem Server Rechner. Da die einmal erfragten Daten im Client Rechner zwischengespeichert werden, sind diese Anfragen normalerweise nicht zeitkritisch. Passwörter werden allerdings immer auf dem Server überprüft und nicht auf dem Client gespeichert. Zur Änderung von Passwörtern gibt es einen separaten RPC Dienst. Die Daten im NIS werden unverschlüsselt übertragen. Dies ist als kritisch zu betrachten, da bei jeder Systemanmeldung und bei jeder Passwortänderung Kennwörter über das Netz übertragen werden. Auch die bei NIS+ verwendete 64 Bit Verschlüsselung kann heute nicht mehr als zeitgemäß betrachtet werden.

Mit einer Kombination des NIS Verfahrens mit dem RPC Tunneling Ansatz von Trapp konnte ich [Müller 03] diese Sicherheitslücke beseitigen. Neben einer geeigneten Konfiguration mussten hier auch der Quellcode des Passwortänderungsprogramms „ypasswd“ angepasst werden. Bei den Quellcode Anpassungen wurde darauf geachtet, dass die Anpassungen auf die Client-Funktionalitäten beschränkt blieben, damit die Kompatibilität zu bestehenden NIS Server Installationen gewährleistet ist. Das resultierende Verfahren nennt sich **Secure Network Information System (SNIS)**.



14.10.2003

Prof. Dr. Chr. Müller

12

Auch hier wurde ein NIS Proxy Server auf dem Client Rechner installiert, an den alle NIS Anfragen des Clients gesendet werden. Der Proxy Server sendet die Anfragen durch einen SSH Tunnel an einen NIS Proxy Client auf dem Server Rechner. Dieser kommuniziert über RPC mit den realen NIS Diensten ypprog und yppasswd auf dem Server Rechner. Auf diese Weise werden unsichere und rechnerübergreifenden RPC Aufrufe vermieden.

Bei Benchmark Untersuchungen [Bowman 03] konnte festgestellt werden, daß mit SNFS etwa 80% der Transferrate von NFS Version 3 erreicht wird. Bei SNIS sind die Geschwindigkeitsverluste praktisch nicht messbar, da die NIS Daten auf den Clients zwischengespeichert werden und damit nur einmal erfragt werden müssen.

SNFS und SNIS sind unter der Gnu Public Licence (GPL) verfügbar [Bowman 03] und werden zur Zeit von Debian evaluiert [Deb-dev 03].

## 6. Alternativen zu SNFS und SNIS

Alternativen zum Network Information System (NIS) ist mit Verzeichnisdiensten wie z.B. X.500 [Chadwick 96] oder LDAP [Smith 03] gegeben. Für diese Verzeichnisdienste existieren Implementierungen für alle gängigen Betriebssysteme, die sich auch systemübergreifend nutzen lassen. Zur Verwaltung von Passwörtern können die Verzeichnisdienste mit Ticketdiensten wie Kerberos kombiniert werden. Bevor eine Clientanwendung einen Serverdienst nutzen kann, muss sie sich bei einem Kerberos Server authentifizieren. Dafür erhält sie von Kerberos ein Ticket in dem die Rechte der Anwendung codiert sind. Bei jeder Nutzung eines Serverdienstes muss der Client dieses Ticket vorweisen. Anhand des Tickets entscheidet der Server, ob der gewünschte Service gewährt wird oder nicht. Die bei Kerberos verwendeten Passwörter und Tickets werden mit geeigneten kryptografischen Verfahren gesichert.

Das Andrew File System (AFS) [OpenAFS 03] ist ein verteiltes Dateisystem, welches mit Kerberos zusammenarbeitet. Das System wurde von der IBM

entwickelt und ist heute unter der GPL verfügbar. Es unterstützt eine optionale Verschlüsselung der übertragenen Daten. Das Management von AFS wird meist als aufwendig erachtet und ist deshalb nur bei großen Organisationen sinnvoll. Anstelle der Standard UNIX Rechteverwaltung unterstützt es Access Control Lists (ACL) für Verzeichnisse. Da sich diese Art der Rechteverwaltung von dem Unix Standard abweicht, hat dies zur Folge, dass viele grafische Anwendungen mit AFS nur begrenzt sinnvoll zusammenarbeiten.

An der University of Michigan wird zur Zeit ein Prototyp von NFS Version 4 [NFSv4 03] entwickelt. Dieser soll im Sommer 2004 verfügbar sein. Diese NFS Version wird erstmals mit Kerberos Tickets arbeiten und neben Access Control Lists (ACL) auch die Standard Unix Rechteverwaltung unterstützen. Die Namen der Dateieigentümer und Nutzungsberechtigten werden uncodiert bereitgestellt. Damit kann dieses Dateisystem zusammen mit Kerberos unabhängig von einem Verzeichnisdienst betrieben werden. Mit einem Dateiattribut wird die Art der Verschlüsselung bei der Datenübertragung individuell konfigurierbar sein. NFS Version 4 ist kompatibel zu dem Windows DFS konzipiert.

Die gegenwärtig verfügbaren NFS Versionen können noch nicht mit Ticketsystemen wie Kerberos zusammenarbeiten. Solange es keine gut in die UNIX Landschaft integrierbare Alternative zu den gegenwärtigen NFS Versionen gibt, die mit AFS bzw. NFS Version 4 vergleichbare Sicherheitseigenschaften haben, wird SNFS in Kombination mit SNIS ein sinnvoller Weg zur Realisierung eines verteilten Dateisystems sein, bei dem:

- Authentizität,
- Integrität und
- Vertraulichkeit bei der Datenübertragung gewährleistet ist.

## 7. Literatur

- [Barret 02] Barret, Silverman: Secure Shell; O'Reilly; 2002
- [Bird 03] Bird: VPN Information on the World Wide Web; 2003; <http://vpn.shmoo.com/>
- [Bowman 02] Bowman: Secure NFS Via SSH Tunnelling; 2002; <http://www.math.ualberta.ca/imaging/snfs/README.NFS>
- [Bowman 03] Bowman, Müller: Secure NFS and NIS via SSH Tunnel; 2003; <http://www.math.ualberta.ca/imaging/snfs/>
- [Chadwick 96] Chadwick: Understanding X.500 - The Directory; 1996; <http://www.isi.salford.ac.uk/staff/dwc/Version.Web/Contents.htm>
- [Cygwin 03] Cygwin: GNU+Cygnus+Windows; 2003; <http://cygwin.com/>
- [Deb-dev 03] ITP: sec-rpc -- rpc-proxy that allows tunneling of nis and nfs over ssh; 2003; <http://lists.debian.org/debian-devel/2003/debian-devel-200308/msg00314.html>
- [Hess 92] Hess, Safford Pooch: A Unix Network Protocol Security Study: Network Information Service; 1992; <http://ouah.kernsh.org/a-unix-network-protocol.pdf>
- [Kirch 00] Kirch, Dawson: The Linux Network Administrator's Guide, Second Edition, 2000; <http://www.tldp.org/LDP/nag2/index.html>

- [Lisiecki 00] Lisiecki: Linux Remote Procedure Call Programmierung; 2000;  
<http://www.linuxhaven.de/dlhp/HOWTO-test/DE-RPC-Programmierung-HOWTO.html>
- [Müller 03] Müller, Bowman: Secure NIS Via SSH Tunnelling; 2003;  
<http://www.math.ualberta.ca/imaging/snfs/README.NIS>
- [NFSv4 03] Center for Information Technology; NFS Version 4 Open Source; 2003;  
<http://www.citi.umich.edu/projects/nfsv4/>
- [OpenAFS 03] OpenAFS; 2003; <http://www.openafs.org/>
- [OpenSSH 03] OpenSSH 3.7.1; 2003; <http://www.openssh.com/>
- [Putty 03] PuTTY: A Free Win32 Telnet/SSH Client; 2003;  
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>
- [RSA 00] RSA Security: Symetric key lengths; 2000; <http://www.rsasecurity.com/>
- [Smith 03] Smith: LDAP standards and documents; 2003;  
<http://www.mozilla.org/directory/standards.html>
- [Trapp 96] Trapp; Using SSH to increase the security of ONC RPC services; 1996;  
[ftp://ftp.tu-chemnitz.de/pub/Local/informatik/sec\\_rpc/README.RPC](ftp://ftp.tu-chemnitz.de/pub/Local/informatik/sec_rpc/README.RPC)