Firewall Testing with NAT

Semesterarbeit Wintersemester 2005/06

Adrian Schüpbach
Betreuerin: Diana Senn

Information Security

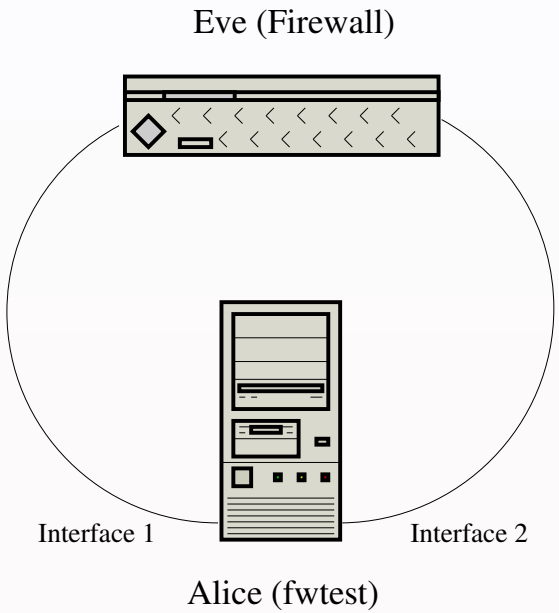
ETH Zürich

7. Februar 2006

Ziele der Arbeit

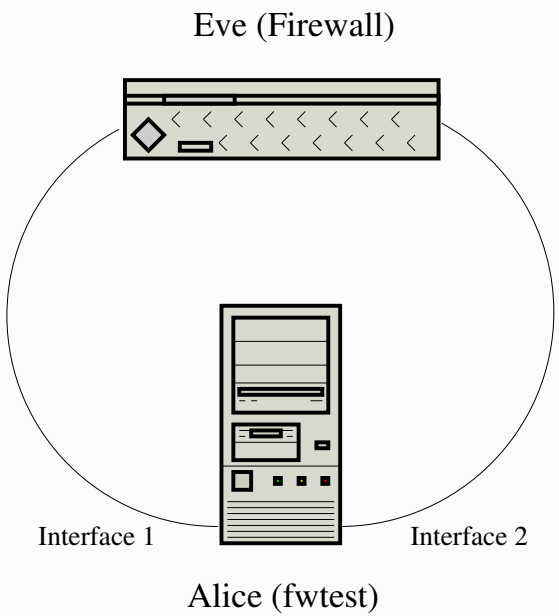
- Eine Instanz von **fwtest**
- Spezifikation von Testfällen, statt einzelner Testpakete
- Unterstützung von NAT

Eine Instanz von **fwtest**



- zwei Netzwerkschnittstellen unterstützen
- keine Synchronisationsprobleme
- globale Sicht über gesendete und empfangene Pakete
- Timeout für gesendete und erwartete Pakete nötig

Eine Instanz von **fwtest**
Datenstrukturanpassung



- jedes zu sendende und erwartete Paket als Send- und Receive-Event in Datenstruktur
- bisher: zwei Datenstrukturen
- neu: Send- und Receive-Event in gleicher Datenstuktur

- Pakete für aktuellen Zeitpunkt generieren
- Pakete senden und Timer starten
- Pakete löschen
- empfangene Pakete mit erwarteten vergleichen
- loggen
  - Pakete, die ankommen sollten, aber nicht gekommen sind
  - Pakete, die ankommen, obwohl sie nicht sollten
- zum nächsten Zeitpunkt übergehen

Bisher

- einzelne Pakete im TP-File

Neu

- Pakete gehören zu Testfall
- Testfall hat eindeutige Nummer
- Paketnummern **innerhalb** des Testfalles eindeutig

- Synchronisation mit absoluter Zeit funktioniert mit einer Instanz
- relative Zeit reicht jetzt aus
- relative Zeit nötig wegen Paketreihenfolge
- Paketreihenfolge wichtig
  - zum Beispiel TCP-Verbindungsaufbau
    1. SYN
    2. SYN, ACK
    3. ACK
- relative Zeit durch Paketnummern
  - Paketnummer jedes Pakets eindeutig **innerhalb** eines Testfalles

- Paket mit kleinerer Nummer vor Paket mit grösserer Nummer abarbeiten
  - generieren
  - senden
  - empfangen und vergleichen
- zeitliche Abstände von Paketen nur vom Timeout abhängig
  - für verschiedene Tests mit verschiedenen zeitlichen Abständen nur ein TP-File nötig
  - absolute Zeit weder für Synchronisation noch für zeitliche Abstände nötig
- somit relative Zeit erreicht

Beispiel-TP-File

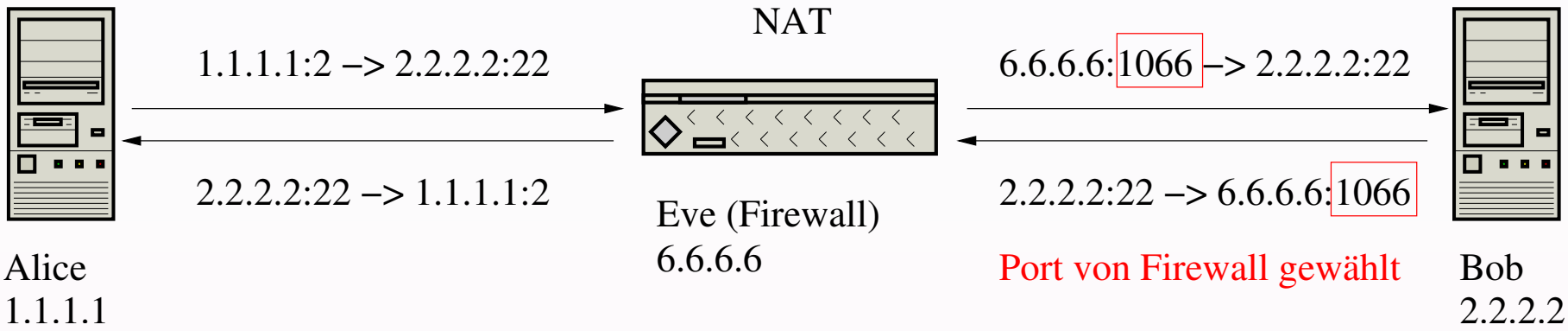
```
testcase 0{
    packet 0{...}
    packet 4{...}
}
testcase 1{
    packet 0{...}
    packet 2{...}
}
```

- Pakete mit Nummer 0 gleichzeitig gesendet
- danach Paket 2 aus Testcase 1
- danach Paket 4 aus Testcase 0

NAT-Unterstützung
wichtige Punkte

- Beschreibung des zu sendenden **und** des erwarteten Paketes
- Variablen für IP- und Port-Nummern einführen
- Paket-Identifikation
- Pakete erst kurz vor dem Senden generieren

NAT-Unterstützung
Variablen



- Source-Port von Firewall beliebig gewählt bei NAT
- gesendetes Paket sieht somit beim Empfang auf anderer Firewallseite anders aus
- Variablen für Portnummern nötig
- Variablen für IP-Nummern praktisch
- Variable **innerhalb** eines Testfalles gültig

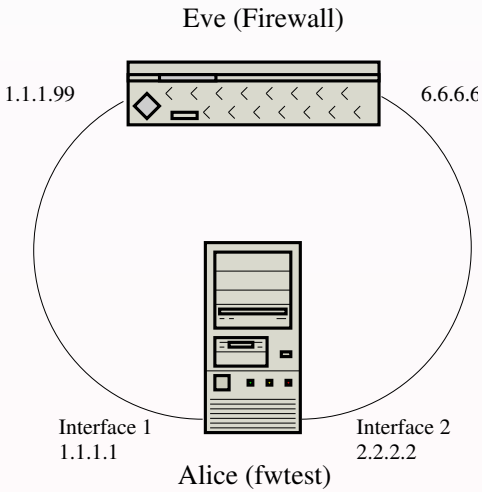
- bisher alle Felder (IP's, Port's,...) verglichen
  - ginge immernoch mit Variablen
  - unschön
- neu eindeutige Paketnummer (Paket-ID) mitgesendet
- Paket-ID in IP-ID-Feld mitsendbar (standard)
  - für TCP, UDP und ICMP möglich
- Paket-ID als Daten mitsenden (alternativ)
  - nur für TCP und UDP möglich
  - vom Benützer wählbar mit Kommandozeilen-Option
  - für ICMP-Pakete weiterhin im IP-ID-Feld

- empfangenes Paket anhand Paket-ID identifizieren
- ist empfangenes Paket wie erwartet?
  - Überprüfung aller Felder auf Gleichheit mit erwarteten Werten
    - IP's
    - Port's
    - SYN- und ACK-Nummern
    - ...
  - Variablen im erwarteten Paket überprüfen

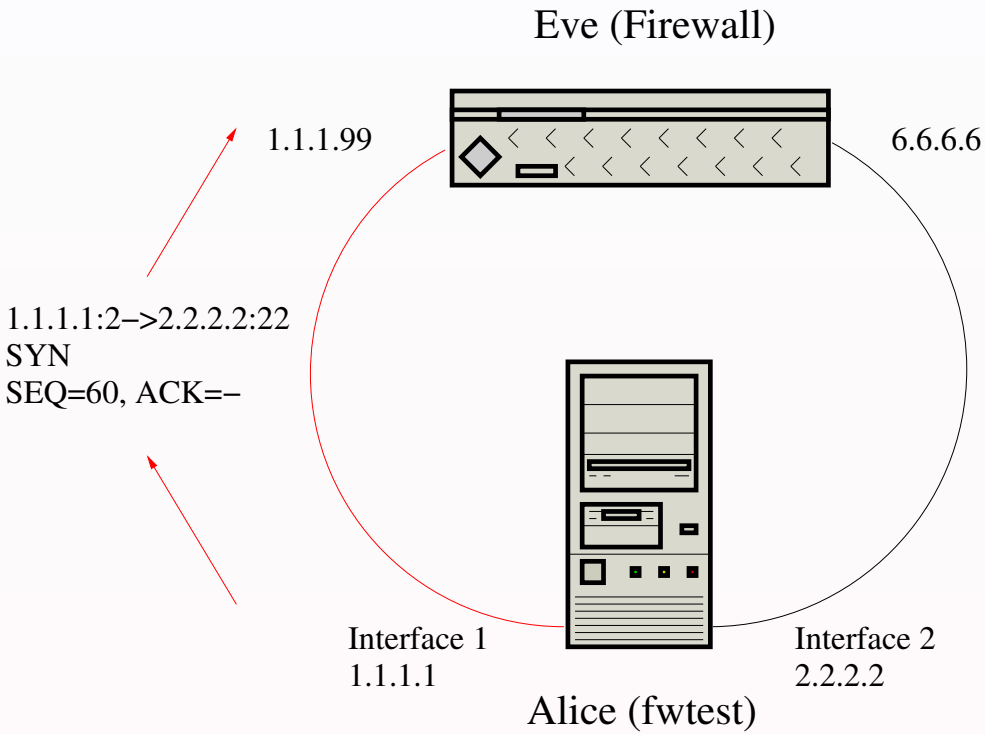
- Wertzuweisung an Variablen in erwarteten Paketen
  - Wert des Feldes des ersten erwarteten Paketes zuweisen
  - für weitere Pakete gültig
- Wertüberprüfung von Variablen
  - Spätere Pakete: Gleicher Wert wie Variablenwert im entsprechenden Feld zwingend

NAT-Unterstützung
Beispiel: TP-File und Testaufbau

```
testcase 0{
    packet 0{
    TCP
    send {1.1.1.1 2.2.2.2 2 22 S 60 -}
    receive {6.6.6.6 2.2.2.2 VarA 22 S 60 -}
    }
    packet 1{
    TCP
    send {2.2.2.2 6.6.6.6 22 VarA SA 70 61}
    receive {2.2.2.2 1.1.1.1 22 2 SA 70 61}
    }
    packet 2{
    TCP
    send {1.1.1.1 2.2.2.2 2 22 A 61 71}
    receive {6.6.6.6 2.2.2.2 VarA 22 A 61 71}
    }
}
```

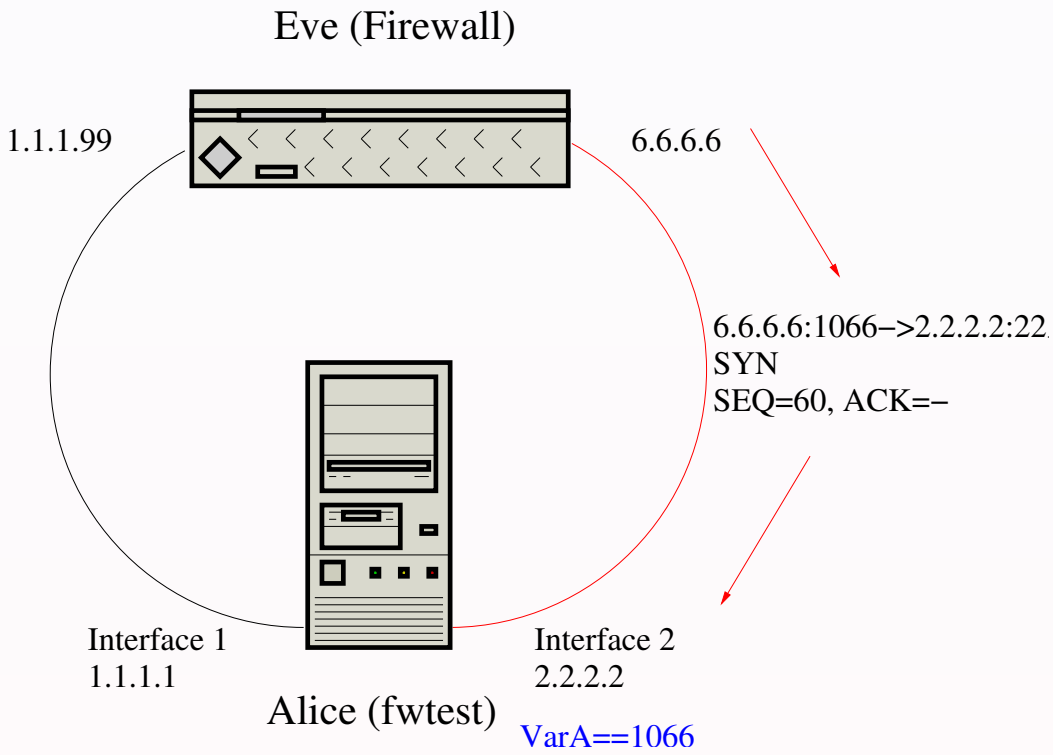


NAT-Unterstützung
Beispiel – erstes Paket zu Firewall senden



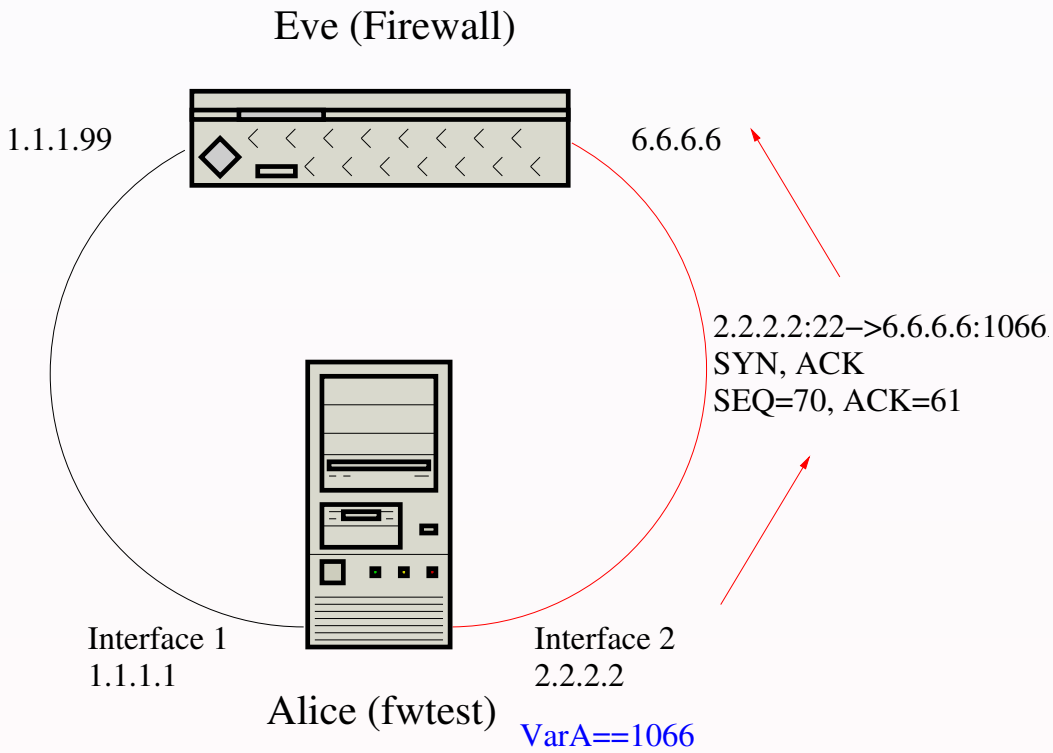
- sende erstes Paket zur Firewall

NAT-Unterstützung
Beispiel – erstes Paket zum Ziel



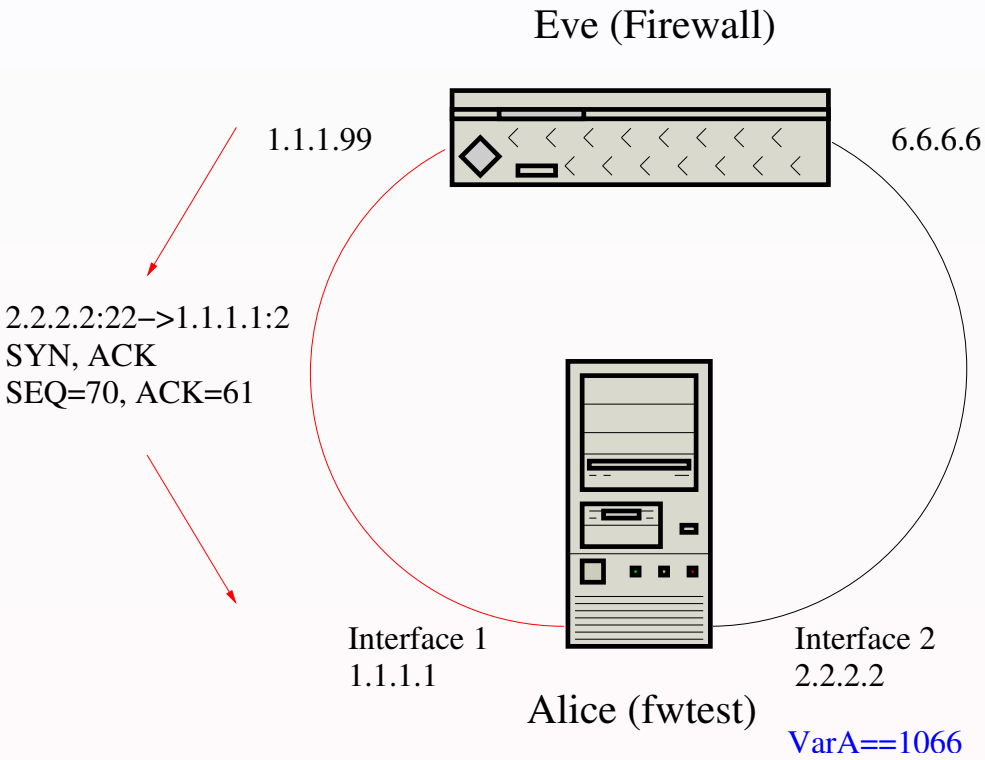
- Firewall wählt Source-Port 1066
- sendet Paket mit gewähltem Port und eigener IP-Nummer weiter
- erstes Vorkommen von VarA → **fwtest** setzt VarA=1066

NAT-Unterstützung
Beispiel – sende Antwort zur Firewall

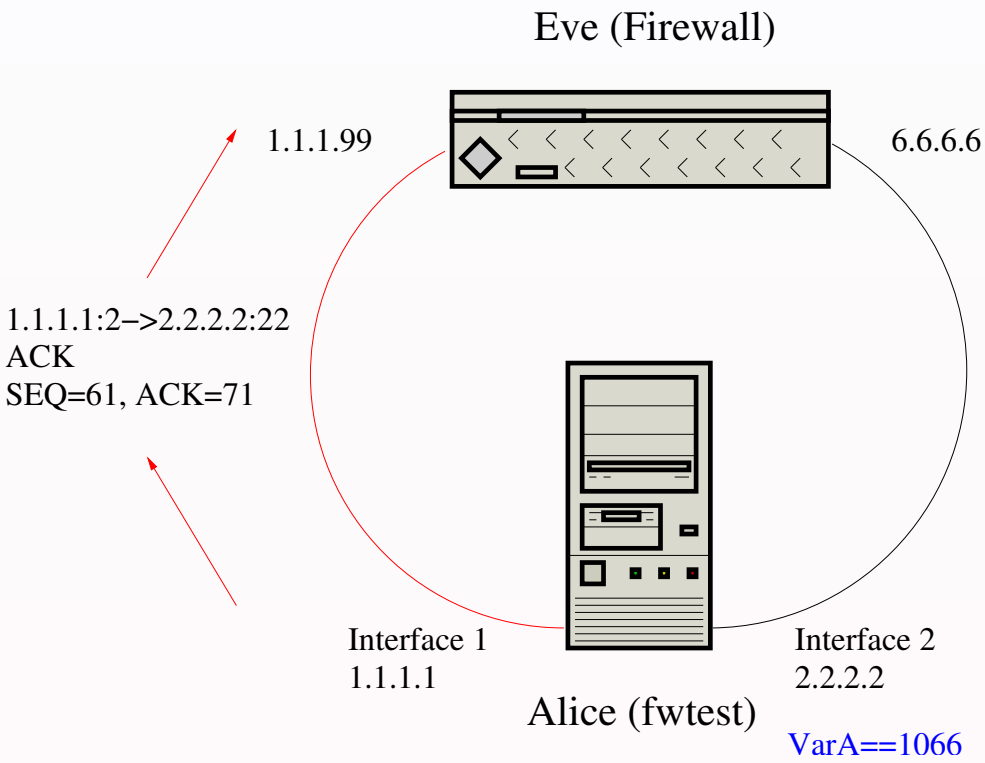


- Paket mit VarA=1066 generieren
- zur Firewall senden

NAT-Unterstützung
Beispiel – Firewall sendet Antwort weiter

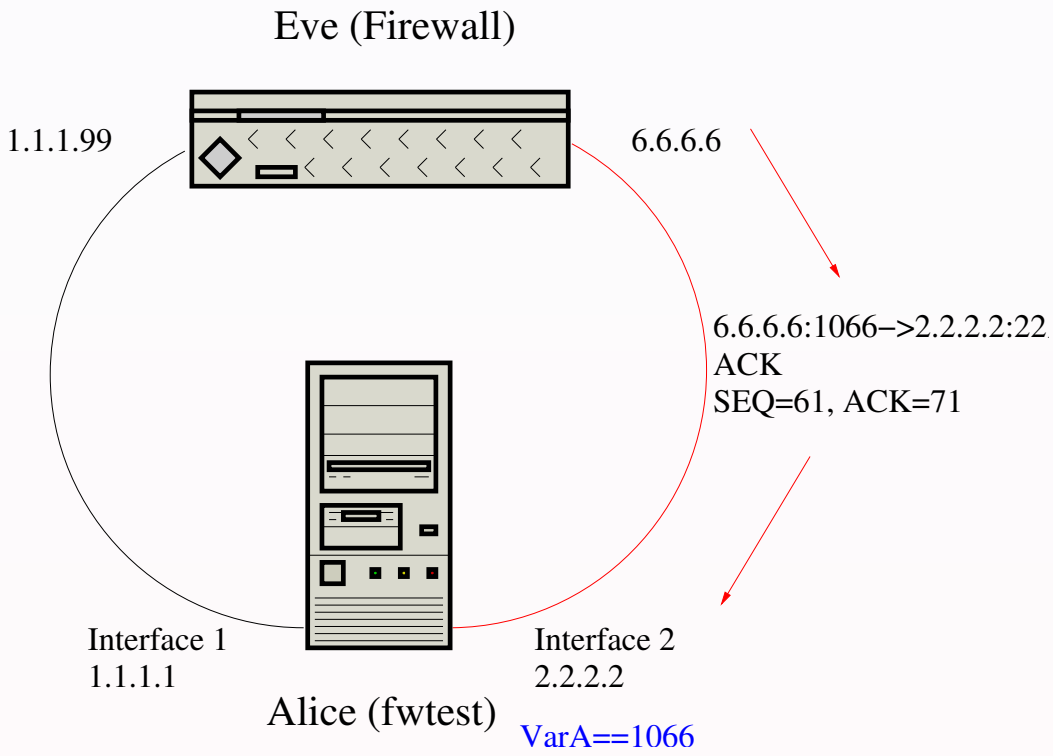


- Firewall sendet Paket mit originalem Ziel-Port weiter



- sende ein ACK-Paket

NAT-Unterstützung
Beispiel – Firewall sendet ACK-Paket weiter



- setze für Source-Port den gleichen Wert wie vorher gewählt ein
- sende Paket weiter
- **fwtest** überprüft, ob Variablenwert immernoch gleich ist

Generierung der Pakete

- Generieren der Pakete erst kurz vor dem Senden
  - weniger Speicher zur Laufzeit benötigt
    - Pakete nach dem Senden wieder löschen
  - Variablenwerte für Generierung benötigt
- Löschen der Pakete
  - kurz nach dem Senden
  - Beschreibung des erwarteten Paketes immernoch vorhanden

Schlussfolgerung

- viele Änderungen an der Programmstruktur von **fwtest** waren nötig um es als eine Instanz benützen zu können
- eine Instanz mit zwei Schnittstellen funktioniert
- einige Änderungen nötig, um zweite Schnittstelle in Betrieb zu nehmen
- Vereinfachung des Timers
- Unterstützung von NAT gelungen
  - durch ID Paket eindeutig identifizierbar
  - durch einführung von Variablen **alle** Felder auf Gleichheit wie erwartet überprüfbar

Ausblick

- Variabler Timeout
  - vom Benützer wählbarer Timeoutwert an der Kommandozeile
  - automatische Wahl des Timeoutwertes durch **fwtest**
- Sendewiederholung von Paketen, die nicht angekommen sind
- Erwartung von ICMP-Paketen (ohne sie zu senden)
  - Firewalls können Fehlermeldungen senden, die man ev. prüfen will
- weitere ICMP-Typen
- IPv6-Unterstützung
- Versenden von Paketen mit Daten