ALPHA

Johannes Gilger, Florian Weingarten

December 13th, 2008

Adaptive and Lightweight Protocol
for Hop-By-Hop Authentication

What we did this week

Tasks

- **Goal:** Encapsulate (arbitrary layer 3) traffic from host A to host B in UDP packets (while preserving application layer transparency)
- **Task:** Find out how to do it

Solution

- TUN/TAP interface

What we did this week

Tasks

- **Goal:** Encapsulate (arbitrary layer 3) traffic from host A to host B in UDP packets (while preserving application layer transparency)
- **Task:** Find out how to do it

Solution

- TUN/TAP interface

What we did this week

Tasks

- **Goal:** Encapsulate (arbitrary layer 3) traffic from host A to host B in UDP packets (while preserving application layer transparency)
- **Task:** Find out how to do it

Solution

- TUN/TAP interface

Universal **TUN/TAP** device driver

- (virtual) network devices, supported entirely in software
- available for Linux ($\Rightarrow$ OpenWRT, Nokia), *BSD, Windows, OS X
- used in OpenVPN, VTun, OpenSSH, ...
- similar to ipqueue, but higher portability

- **TUN: userland support for IP tunneling (layer 3)**
- TAP: userland support for ethernet bridging (layer 2)

- tun0 can be used like any other device in the routing table
- traffic gets passed to userspace via `/dev/net/tun`
- transparent for application layer

Universal **TUN/TAP** device driver

- (virtual) network devices, supported entirely in software
- available for Linux ($\Rightarrow$ OpenWRT, Nokia), *BSD, Windows, OS X
- used in OpenVPN, VTun, OpenSSH, ...
- similar to ipqueue, but higher portability

- **TUN: userland support for IP tunneling (layer 3)**
- TAP: userland support for ethernet bridging (layer 2)

- tun0 can be used like any other device in the routing table
- traffic gets passed to userspace via `/dev/net/tun`
- transparent for application layer

Universal **TUN/TAP** device driver

- (virtual) network devices, supported entirely in software
- available for Linux ($\Rightarrow$ OpenWRT, Nokia), *BSD, Windows, OS X
- used in OpenVPN, VTun, OpenSSH, ...
- similar to ipqueue, but higher portability

- **TUN: userland support for IP tunneling (layer 3)**
- TAP: userland support for ethernet bridging (layer 2)

- tun0 can be used like any other device in the routing table
- traffic gets passed to userspace via `/dev/net/tun`
- transparent for application layer

Universal **TUN/TAP** device driver

- (virtual) network devices, supported entirely in software
- available for Linux ($\Rightarrow$ OpenWRT, Nokia), *BSD, Windows, OS X
- used in OpenVPN, VTun, OpenSSH, ...
- similar to ipqueue, but higher portability

- **TUN: userland support for IP tunneling (layer 3)**
- TAP: userland support for ethernet bridging (layer 2)

- `tun0` can be used like any other device in the routing table
- traffic gets passed to userspace via `/dev/net/tun`
- transparent for application layer

Universal **TUN/TAP** device driver: How to use it?

- `tun = open("/dev/net/tun", O_RDWR);`
- `ifr.ifr_flags = IFF_TUN | IFF_NO_PI;`
- `ioctl(tun, TUNSETIFF, &ifr);`
- `ifconfig tun0 up`
- `route add ... dev tun0`
- `read(tun, ...);`
- `write(tun, ...);`

Universal **TUN/TAP** device driver: How to use it?

- `tun = open("/dev/net/tun", O_RDWR);`
- `ifr.ifr_flags = IFF_TUN | IFF_NO_PI;`
- `ioctl(tun, TUNSETIFF, &ifr);`
- `ifconfig tun0 up`
- `route add ... dev tun0`
- `read(tun, ...);`
- `write(tun, ...);`

Routing

A few thoughs about routing

- Problem: We can't use the public IP of the remote host in the application. Route is either through tun or through the "regular" network interfaces.
- Solution: Introduce private IP addresses for traffic to be used with the Alpha protocol.
- Alternate solution: Each host has two public IP addresses, one for tunneling and one for addressing.

Routing

A few thoughs about routing

- Problem: We can't use the public IP of the remote host in the application. Route is either through tun or through the "regular" network interfaces.
- Solution: Introduce private IP addresses for traffic to be used with the Alpha protocol.
- Alternate solution: Each host has two public IP addresses, one for tunneling and one for addressing.

Routing

A few thoughs about routing

- Problem: We can't use the public IP of the remote host in the application. Route is either through tun or through the "regular" network interfaces.
- Solution: Introduce private IP addresses for traffic to be used with the Alpha protocol.
- Alternate solution: Each host has two public IP addresses, one for tunneling and one for addressing.

One or many Alpha daemons?

Alpha-Daemons

- Problem: Use one Alpha-daemon (and one tun-device) for every client-server connection or use one global daemon?
  1. Is certainly easier to implement but might exhaust resources.
  2. Second requires a lot of extra work in Alpha. Processing packets from different hosts, mapping private IPs to public IPs, using the right hash-chains etc.
- Solution: None yet ;)

One or many Alpha daemons?

Alpha-Daemons

- Problem: Use one Alpha-daemon (and one tun-device) for every client-server connection or use one global daemon?
  1. Is certainly easier to implement but might exhaust resources.
  2. Second requires a lot of extra work in Alpha. Processing packets from different hosts, mapping private IPs to public IPs, using the right hash-chains etc.
- Solution: None yet ;)

One or many Alpha daemons?

Alpha-Daemons

- Problem: Use one Alpha-daemon (and one tun-device) for every client-server connection or use one global daemon?
  1. Is certainly easier to implement but might exhaust resources.
  2. Second requires a lot of extra work in Alpha. Processing packets from different hosts, mapping private IPs to public IPs, using the right hash-chains etc.
- Solution: None yet ;)

One or many Alpha daemons?

Alpha-Daemons

- Problem: Use one Alpha-daemon (and one tun-device) for every client-server connection or use one global daemon?
  1. Is certainly easier to implement but might exhaust resources.
  2. Second requires a lot of extra work in Alpha. Processing packets from different hosts, mapping private IPs to public IPs, using the right hash-chains etc.
- Solution: None yet ;)

What we still have to do

- Implement handshake and exchange of hash-chains for Alpha communication.
- Implement the core Alpha-protocol.
- Implement a smart way of choosing between the different Alpha-modes.
- Make sure it's portable code.

What we still have to do

- Implement handshake and exchange of hash-chains for Alpha communication.
- Implement the core Alpha-protocol.
- Implement a smart way of choosing between the different Alpha-modes.
- Make sure it's portable code.

What we still have to do

- Implement handshake and exchange of hash-chains for Alpha communication.
- Implement the core Alpha-protocol.
- Implement a smart way of choosing between the different Alpha-modes.
- Make sure it's portable code.

What we still have to do

- Implement handshake and exchange of hash-chains for Alpha communication.
- Implement the core Alpha-protocol.
- Implement a smart way of choosing between the different Alpha-modes.
- Make sure it's portable code.