ALPHA

Johannes Gilger, Florian Weingarten

December 13th, 2008

Adaptive and Lightweight Protocol
for Hop-By-Hop Authentication

Tasks

Our goals for this week

1. Figure out a solution for the routing-question from last week.
2. Get familiarized with *TRAC*.
3. Implement the UDP encapsulation and a dummy handshake.

Goals for the holidays

1. Finish implementing the encapsulation/handshake :-)
2. Have a look at Merkle-tree source code from Tobi.
3. Document our efforts.

Tasks

Our goals for this week

1. Figure out a solution for the routing-question from last week.
2. Get familiarized with *TRAC*.
3. Implement the UDP encapsulation and a dummy handshake.

Goals for the holidays

1. Finish implementing the encapsulation/handshake :-)
2. Have a look at Merkle-tree source code from Tobi.
3. Document our efforts.

Tasks

Our goals for this week

1. Figure out a solution for the routing-question from last week.
2. Get familiarized with *TRAC*.
3. Implement the UDP encapsulation and a dummy handshake.

Goals for the holidays

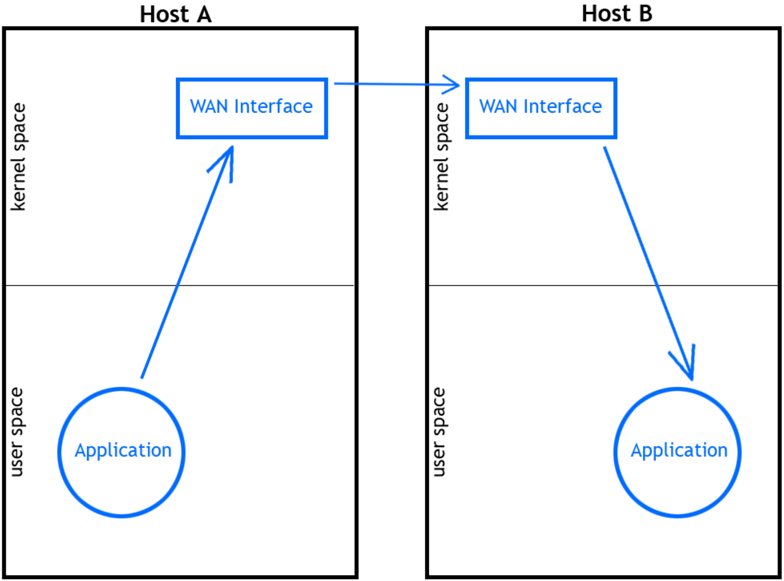
1. Finish implementing the encapsulation/handshake :-)
2. Have a look at Merkle-tree source code from Tobi.
3. Document our efforts.

Encapsulation

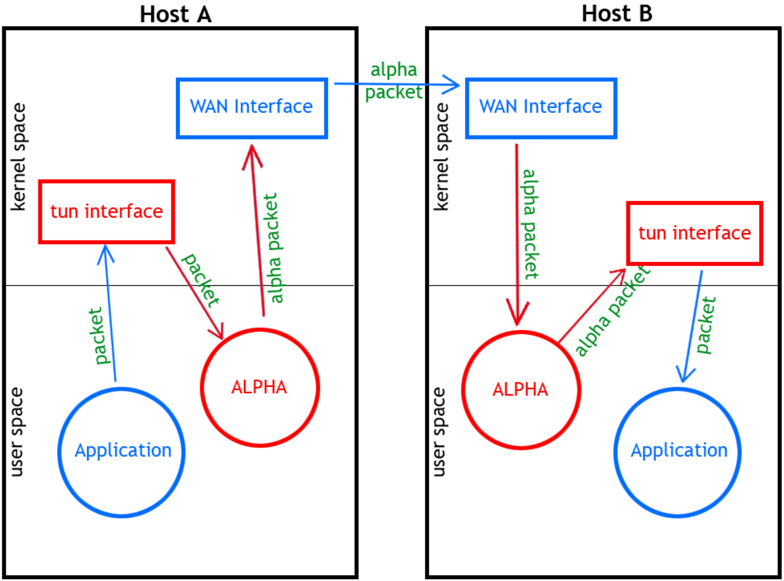
Encapsulation of arbitrary IP packets into UDP packets.

Was presented in a bad way last week, that's why we want to reiterate and really point out what Alpha is supposed to do and the problems we've been having.

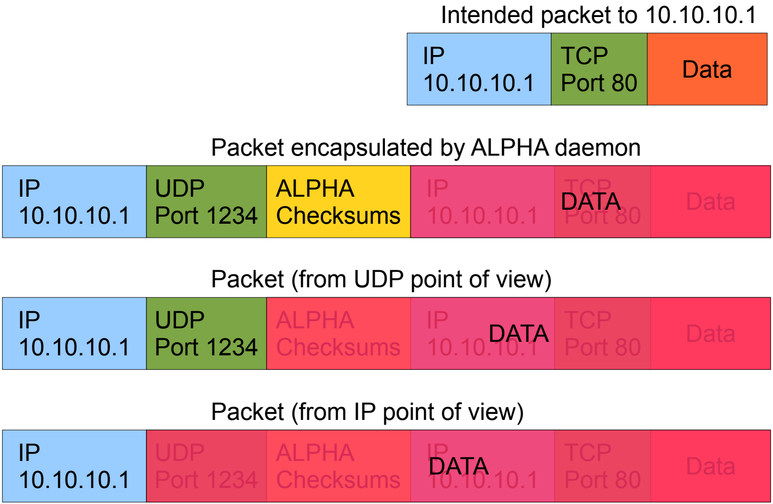
The „usual“ situation



The ALPHA situation



What does an ALPHA packet look like?



Intended packet to 10.10.10.1

IP 10.10.10.1 | TCP Port 80 | Data

Packet encapsulated by ALPHA daemon

IP 10.10.10.1 | UDP Port 1234 | ALPHA Checksums | IP 10.10.10.1 | TCP Port 80 | Data

DATA

Packet (from UDP point of view)

IP 10.10.10.1 | UDP Port 1234 | ALPHA Checksums | IP 10.10.10.1 | TCP Port 80 | Data

DATA

Packet (from IP point of view)

IP 10.10.10.1 | UDP Port 1234 | ALPHA Checksums | IP 10.10.10.1 | TCP Port 80 | Data

DATA

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Idea 1

- „Thats basically what `bind()` does, isn't it?"
- **No!**
- `bind()` binds to *addresses*, not to *interfaces*

Idea 2

- socket option `SO_DONTROUTE`
- **No!**
- Routing is completely ignored, only connections to hosts which are directly reachable are possible

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass tun0)

Idea 1

- „Thats basically what bind() does, isn't it?“
- **No!**
- bind() binds to *addresses*, not to *interfaces*

Idea 2

- socket option SO_DONTROUTE
- **No!**
- Routing is completely ignored, only connections to hosts which are directly reachable are possible

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Idea 1

- „Thats basically what `bind()` does, isn't it?“
- **No!**
- `bind()` binds to *addresses*, not to *interfaces*

Idea 2

- socket option `SO_DONTROUTE`
- **No!**
- Routing is completely ignored, only connections to hosts which are directly reachable are possible

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass tun0)

Idea 3

- iptables/netfilter
- something like `iptables -A POSTROUTING -j ROUTE --oif tun0 ! -p udp --dport 1234`
- **No!**
- Not portable

Idea 4

- socket option `SO_BINDTODEVICE`
- **No!** (unfortunately)
- Was our best bet, found (almost completely undocumented) in old Linux Kernel documentation files.
- Seems to be available at least for BSD based sockets (Linux, OSX).
- Also didn't work the way we expected it :(

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Idea 3

- `iptables`/netfilter
- something like `iptables -A POSTROUTING -j ROUTE --oif tun0 ! -p udp --dport 1234`
- **No!**
- Not portable

Idea 4

- socket option `SO_BINDTODEVICE`
- **No!** (unfortunately)
- Was our best bet, found (almost completely undocumented) in old Linux Kernel documentation files.
- Seems to be available at least for BSD based sockets (Linux, OSX).
- Also didn't work the way we expected it :(

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Idea 3

- `iptables`/netfilter
- something like `iptables -A POSTROUTING -j ROUTE --oif tun0 ! -p udp --dport 1234`
- **No!**
- Not portable

Idea 4

- socket option `SO_BINDTODEVICE`
- **No!** (unfortunately)
- Was our best bet, found (almost completely undocumented) in old Linux Kernel documentation files.
- Seems to be available at least for BSD based sockets (Linux, OSX).
- Also didn't work the way we expected it :(

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Remaining ideas

- Look into RAW sockets and if they can be used with SO_BINDTODEVICE.
- Last resort: Private IP address space (mentioned before)

Private IP address space - why we didn't want to go there

- **Not** "transparent" for user-space, at least from the addressing point of view
- Alpha-Daemon will need a way to associate public IP with private IP, either by a translation scheme or by a table supplied by the user

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Remaining ideas

- Look into RAW sockets and if they can be used with SO_BINDTODEVICE.
- Last resort: Private IP address space (mentioned before)

Private IP address space - why we didn't want to go there

- **Not** "transparent" for user-space, at least from the addressing point of view
- Alpha-Daemon will need a way to associate public IP with private IP, either by a translation scheme or by a table supplied by the user

Problem: Bind traffic of ALPHA daemon to WAN interface (bypass `tun0`)

Remaining ideas

- Look into RAW sockets and if they can be used with SO_BINDTODEVICE.
- Last resort: Private IP address space (mentioned before)

Private IP address space - why we didn't want to go there

- **Not** "transparent" for user-space, at least from the addressing point of view
- Alpha-Daemon will need a way to associate public IP with private IP, either by a translation scheme or by a table supplied by the user

TRAC - Web-based bug-tracker/wiki/source-code-browser

What is TRAC used for?

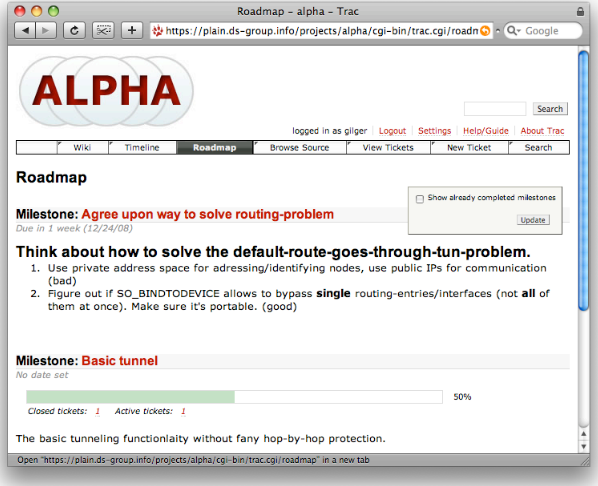
- Public Wiki (used for documentation and ideas). Free-text format
- Bugtracker: Tickets can be created, assigned to a user, given a priority and their solution can be linked to milestones to be reached
- Source-Code-Browser: Display a SVN-repository with the usual actions of diffing etc. (We don't really use it, but good for public projects)

TRAC - Web-based bug-tracker/wiki/source-code-browser

What is TRAC used for?

- Public Wiki (used for documentation and ideas). Free-text format
- Bugtracker: Tickets can be created, assigned to a user, given a priority and their solution can be linked to milestones to be reached
- Source-Code-Browser: Display a SVN-repository with the usual actions of diffing etc. (We don't really use it, but good for public projects)

TRAC - Web-based bug-tracker/wiki/source-code-browser



Roadmap – alpha – Trac
https://plain.ds-group.info/projects/alpha/cgi-bin/trac.cgi/roadm
Google
ALPHA
Search
logged in as gilger | Logout | Settings | Help/Guide | About Trac
Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search

Roadmap

□ Show already completed milestones

Update

Milestone: Agree upon way to solve routing-problem

Due in 1 week (12/24/08)

Think about how to solve the default-route-goes-through-tun-problem.

1. Use private address space for adressing/identifying nodes, use public IPs for communication (bad)
2. Figure out if SO_BINDTODEVICE allows to bypass **single** routing-entries/interfaces (not **all** of them at once). Make sure it's portable. (good)

Milestone: Basic tunnel

No date set

50%

Closed tickets: 1 *Active tickets:* 1

The basic tunneling functionlaity without fany hop-by-hop protection.

Open "https://plain.ds-group.info/projects/alpha/cgi-bin/trac.cgi/roadmap" in a new tab

TRAC - Web-based bug-tracker/wiki/source-code-browser

When to use TRAC

- For public projects with lots of prospective dev comfortable with using web-based tools.
- When wanting to make sure everyone can write documentation (and when you don't want to).
- For having a convenient and standardized way to receive bug-reports and track their progress.

Alternatives

- Manage documentation and TODOs directly with the source-code.
- Manage bug-reports and developer/user-communication using a mailing list.
- Don't require anyone to use a webbrowser to participate, especially when then project is a program written for the shell.
- Think about how much time you spend putting stuff into web-forms that is already available in plain-text (time better spent coding).

TRAC - Web-based bug-tracker/wiki/source-code-browser

When to use TRAC

- For public projects with lots of prospective dev comfortable with using web-based tools.
- When wanting to make sure everyone can write documentation (and when you don't want to).
- For having a convenient and standardized way to receive bug-reports and track their progress.

Alternatives

- Manage documentation and TODOs directly with the source-code.
- Manage bug-reports and developer/user-communication using a mailing list.
- Don't require anyone to use a webbrowser to participate, especially when then project is a program written for the shell.
- Think about how much time you spend putting stuff into web-forms that is already available in plain-text (time better spent coding).