Zorp and KZorp: Integrating Packet Filtering and Userspace proxying

Balázs Scheidler
<bazsi@balabit.com>

Zorp

Has been established in 2000, as the first BalaBit product

Code was GPLd right from the start

- Initial set of proxies and the Zorp Core
- Transparent Proxying (merged finally in 2008 after a few incarnations)
- OS integration tools

Used as a part of our solutions, and is embedded in our Firewall product

We reinforced our open source efforts in 2013

- published more of the code and
- further integration into upstream projects

Trends on Our Internet

Internet access is ubiquitous (mobile phones, tablets, WiFi everywhere)

Mobile Apps rely on the availability of the Internet (the „cloud”)

Traditional network boundaries became fuzzy, as we carry our devices around

Consequences

IP addresses are **not static** anymore (CDNs, distributed services, ...)

Everything **talks HTTP,** not just **browsers**, so L7 protocols are not distinctive anymore

More & more things talk **HTTPS** (fortunately!), which is completely opaque to most firewalls

Extensions to HTTP like **WebSockets & WebRTC**

The **complexity** of browsers and apps (that use HTTP) represents an ever growing attack surface

Traditional Firewall Approach

Internal & external networks (+ DMZ)

Shielding clients/servers at the perimeter

Filter based on direction of connection, IP addresses, ports

Protect network stacks against implementation errors (synflood, ISN problems, source routing, Xmas, DoS, ...)

Challenges

To be controlled, user visible services all use HTTP:

- Chatting on Facebook vs. Google Search vs. Online Banking vs. Windows Update

And their IP addresses are changing all the time

Direction of the connection is not really relevant anymore

Network stacks became much better wrt implementation errors, and their effect is more limited

Attacks move higher up in the stack

To provide any **meaningful** control, a **peek** into the L7 traffic is needed, sometimes even **complete proxying**

Use best of Both Worlds!

Use hybrids

Multiple levels of processing:

- Packet filtering
- Circuit level gateways (e.g. synproxy)
- Protocol aware L7 proxying
- Application specific (e.g. Facebook) where you need to exert more control

In order to make it usable, it has to be **easy** to **switch** between **Processing Levels**

With iptables/nft...

It's a bit difficult

- Multiple rules/tables to achieve the same effect
- Iterative evaluation
- Switching a PF rule to proxying can mean changing a number of rules
- A bit of a low-level

Making it easy to switch

Make it trivial to direct the firewall to do one or the other, without inflicting changes on the ruleset

- e.g. create a rule for online banking, or Windows Update

Make it possible to switch between processing levels based on a runtime decision

- e.g. SSL certificates, SNI, etc

We wanted to make the **„service”** be the focus

Service

Conditions, selectors are like in a firewall:

- IP addresses (ipv4 or ipv6)
- Ports
- Interface
- Zones (set of IP addresses)
- VPN connection
- Authentication information
- ...

Key difference: **best match**, instead of iterative evaluation

Zorp Configuration Model

Zones (set of IP addresses, but also DNS names)

Service selection based on **Best Match**

- Integrating many sources of information into the condition part (interface, IP, ports, VPN, authentication, ...)

Actions include:

- Forward via packet filter
- Forward via Proxy
- Drop

Lookup based instead of iterative

Architecture

Policy & customization (Python)

Zorp (proxies)

kzorpd (userspace integration)

kzorp (best match)

Netfilter (conntrack, nat)

Use cases

Zones based on DNS name (e.g. a rule for gmail.com, whatever IP that means)

Easily add custom rule for online banking, w/o affecting the rest of the ruleset

MITM for SSL encrypted HTTPS sessions, fall back to PF (or kernel level proxying) based on SNI

API call inspection/alteration: transparently encrypt Google Calendar data

Zorp Features

Proxies for a number widespread protocols (HTTP, FTP, ...)

Transparent at L3, L2 is possible

Customizable, programmable from Python

Handing over payload to further processing (virus scanning, DLP, etc)

Ongoing Work

Separating & generalizing the KZorp component

Merging as much of this to the kernel as possible

Decouple Zorp from Python, and make that optional

Maintaining Zorp in distributions (Debian, OpenWRT)

Questions?

Zorp Tutorial

http://zorp-gpl-tutorial.readthedocs.org/en/latest/

Zorp Homepage

http://bit.ly/1mx4gnW

Reach the authors:

zorp@lists.balabit.hu

Github:

http://github.com/balabit