ProtoWrap: Using wrappers to protect specific network services

Gunnar Eyal Wolf Iszaevich
gwolf@campus.iztacala.unam.mx
UNAM FES Iztacala
DSC - DGSCA
UNAM, Mexico

May 30, 2001

Abstract

In this text I present ProtoWrap, a potential security tool, developed by me during the last year as a final paper project. This document's structure touches only some of ProtoWrap's main points. For further information, you can read my full final paper, available at http://www.gwolf.cx/seguridad/wrap

0.1 Early warning on Gunnar and ProtoWrap

- Gunnar is not a particularly gifted programmer
- Gunnar has some ides he thinks are neat... You may not agree
- Gunnar may need to reimplement this whole mess!
- Gunnar only wants to show a nice idea, not to convince everybody about his implementation.

1 Introduction - Why ProtoWrap was born

1.1 Internet decades ago, Internet today

- Academic/militar network vs. business/entertainment network
- Mainly expert users vs. mainly newbie users
- Trustable network vs. fear-inspiring network
- TCP/IP just designed vs. TCP/IP fully understood

1.2 Enemies? Who? Why?

- Directed attacks — Spying, stealing, personal rivality
- Ego attacks — script-kiddies, cracker wannabes
- Almost random attacks — Search for resources, pirate servers, bridge attacking machine
- Automatic attacks — Large scale scanning, worms

1.3 Attacks: a taxonomy

- Protocol abuses
- Buffer overflows
- Denial of service
- Information gathering (*footprinting*)
- Spam
- Others

2 Previous steps and research

2.1 How to protect the system from attacks

- Fixing bugs in the code
- Proactive auditing (OpenBSD, SELinux)
- Using firewalls
- Mandatory Access Controls (Trusted*whatever*)
- External protection, at the compiler (Stackguard/Immunix)
- Isolating or wrapping the server — ProtoWrap

2.2 Why did I choose Perl — Main points

- Automatic memory management
- Easy use of strings and pattern matching
- Native use of different programming paradigms
- Immune to buffer overflows
- The only language I am confortable with

2.3 Why would I *not* choose Perl —- Main problems

- Slower at startup than compiled languages
- Slower at runtime than some compiled languages
- Vulnerable to denial of service (speed, more resources needed)
- Not-so-confortable object implementation

2.4 How should it work — Basic characteristics

- Line oriented — Not for every protocol! (UDP, DNS, IRC...)
- Main colateral effect: Server and client never have direct contact
- Base functionality: Protection against buffer overflows
- Base characteristic: Extensibility, which avoids protocol abuses or direct attacks to the server

2.5 Dangers that arise from using wrappers

- An extra point of failure to keep an eye on
- Propension to denial of service (and why this does not hurt that much)
- Very noticeable use of extra system resources
- Not every protocol can be wrapped! (HTTP, FTP...)

2.6 Full adherence to standards — RFCs

- Highest-ranking definitory documents on Internet
- The *whole* protocol must be implemented, at least on its base version
- ProtoWrap must be transparent to client
- When ProtoWrap allows itself to be seen

3 Important points learnt while creating ProtoWrap

3.1 The dual I/O problem

- Forking+IPC
- preset whole lines
- lower-level interfaces (IO::Handle+IO::Select+IPC::Open2)

3.2 Allowing for different data sources

- Socket in vs. STDIN in: We need to define standalone and inetd modes
- Socket out vs. STDOUT out: We need to define destination types (pipe or ip)

3.3 Low ports, low privilege

- Sometimes we have to start as root - i.e., listening at a low port, running a server program that requires root privilege
- A wrapper should *never* run as root! (Remember, we want to help security, not intruders!)
- setUidTo, runSrvSuid

3.4 Auto-looping

- What happens if an ordinary user starts a wrapper is configured to connect to its own listening socket? A cascade of connections that can make the machine not to respond.
- Preventing it? No, thanks. The same effect can be achieved in many, many other ways. Consider:

  ```
  fork while(1);
  ```

- The bottom line? If you don't trust local users, revoke their access.

4 ProtoWrap's interface to the world

4.1 A basic, generic class — Creation time attributes

Required attributes:

`standalone`

`listenPort`

`destType`

`(pipeCmd | destAddr, destPort)`

Optional attributes:

`maxLineLength`

`logLevel`

`logName`

`testLine`

`testReply`

`runSrvSuid`

`setUidTo`

4.2 Attributes generated at runtime

- `pid`
- `srcPort`
- `srcAddr`
- `srcName`

4.3 Public methods

- `new`
- `getProp`
- `set_maxLineLength`
- `set_logLevel`
- `startServer/stopServer`
- `version`

4.4 Writing protocol-specific ProtoWrap modules

- The `testLine` attribute: Code reference
- Invocation parameters: Reference to the line, socket, procedence

4.5 Existing extensions: SMTP, POP3

- What gets blocked, what passes: Each line is analyzed for correctness in current stage
- Attributes defined for SMTP: `maxMsgSize`, `blockAddrList`, `blockBodyList`, `maxRcpt`, `relayDomainList`, `relayIpList`
- Attributes defined for POP3: `maxLoginAttempts`

4.6 SMTP example: Reimplementing server's functionality

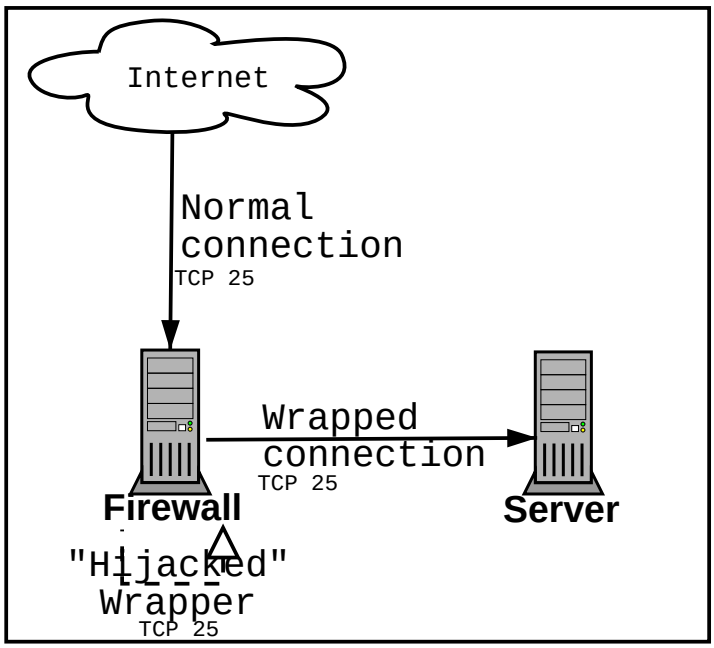
- Why must functionality be reimplemented: Relay controls (`/etc/mail/access`)
- Matching the server's lost functionality: `$wrap->{blockAddrList}`, `$wrap->{blockIpList}`, `$wrap->{relayIpList}`, `$wrap->{relayDomainList}`
- Enhancing functionality: Regex matching, `$wrap->{blockBodyList}`

4.7 Future steps

- Master configuration and startup file
- Signal handling
- Load balancing / high availavility
- POD format documentation

5 Configuration ideas

5.1 ProtoWrap at the firewall

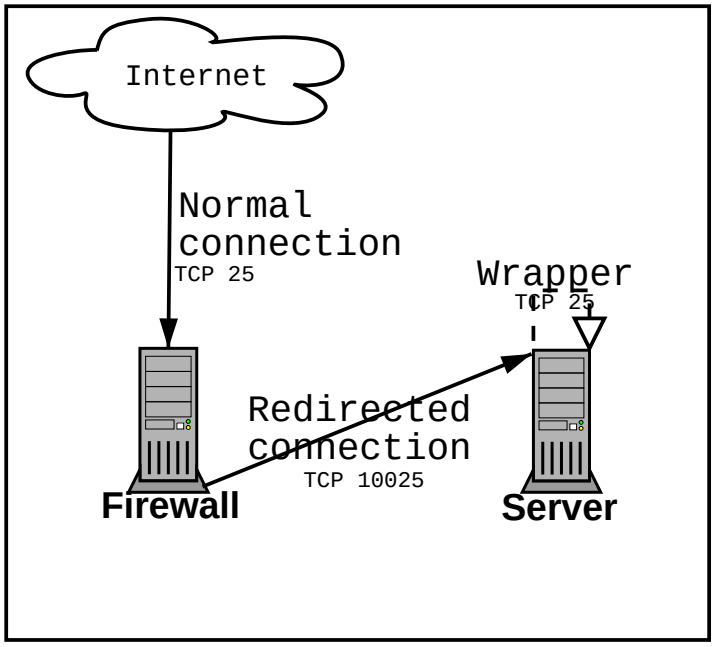


At the firewall, we modify the destination address of every connection, redirecting it to a local port. ProtoWrap is invoked as follows:

```
#!/usr/bin/perl

use ProtoWrap::SMTP;
use strict;

my $wrap;

$wrap = ProtoWrap->new('standalone' => 1,
       'listenPort' => 10025,
       'destType' => 'ip',
       'destAddr' => '192.168.0.1',
       'destPort' => 25,
       'maxMsgSize' => 5000000,
       'blockAddrList' => ['spam.net','spammer@otra.org'],
       'blockBodyList' => ['Content-type.*PIF'],
```

```
       'maxRcpt' => 10,
       'relayDomainList' => ['mivecino.org'],
       'relayIpList' => ['127.0.0'],
       'logLevel' => 2
       );

die "No pude iniciar ProtoWrap" unless (defined($wrap));
$wrap->startServer() or warn 'No puedo iniciar el servidor: '.
                  $wrap->printParam();

sleep;
```

5.2 Redirecting firewall

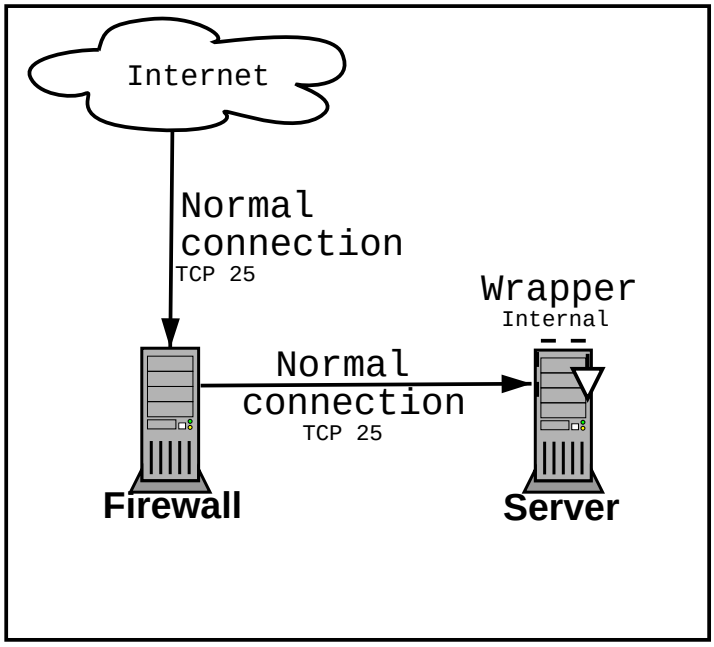


At the firewall, we redirect every incoming packet to get to the port where ProtoWrap is running. We run ProtoWrap with:

```
#!/usr/bin/perl

use ProtoWrap::SMTP;
use strict;

my $wrap;

$wrap = ProtoWrap->new('standalone' => 1,
       'listenPort' => 10025,
       'destType' => 'ip',
       'destAddr' => '127.0.0.1',
       'destPort' => 25,
       'maxMsgSize' => 5000000,
       'blockAddrList' => ['spam.net','spammer@otra.org'],
       'blockBodyList' => ['Content-type.*PIF'],
       'maxRcpt' => 10,
       'relayDomainList' => ['mivecino.org'],
       'relayIpList' => ['127.0.0'],
       'logLevel' => 2
```

```
      );

die "No pude iniciar ProtoWrap" unless (defined($wrap));
$wrap->startServer() or warn 'No puedo iniciar el servidor: '.
                  $wrap->printParam();

sleep;
```

5.3 Server not running, ProtoWrap runs it

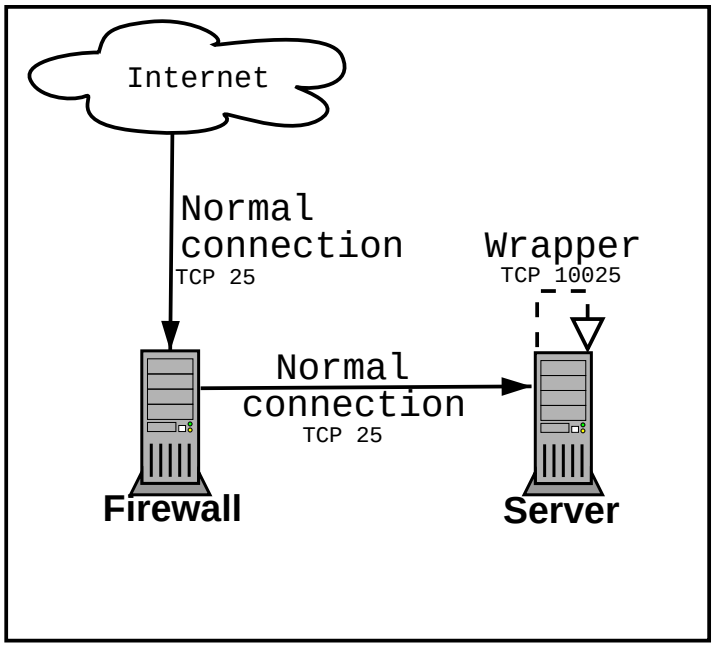


En el servidor no corremos el demonio destino, y ejecutamos ProtoWrap con los siguientes parametros:

```
#!/usr/bin/perl

use ProtoWrap::SMTP;
use strict;

my $wrap;

$wrap = ProtoWrap->new('standalone' => 1,
       'listenPort' => 10025,
       'destType' => 'pipe',
       'pipeCmd' => '/usr/lib/sendmail -bs',
       'maxMsgSize' => 5000000,
       'blockAddrList' => ['spam.net','spammer@otra.org'],
       'blockBodyList' => ['Content-type.*PIF'],
       'maxRcpt' => 10,
       'relayDomainList' => ['mivecino.org'],
       'relayIpList' => ['127.0.0'],
       'logLevel' => 2
       );

die "No pude iniciar ProtoWrap" unless (defined($wrap));
$wrap->startServer() or warn 'No puedo iniciar el servidor: '.
                  $wrap->printParam();
```

```
sleep;
```

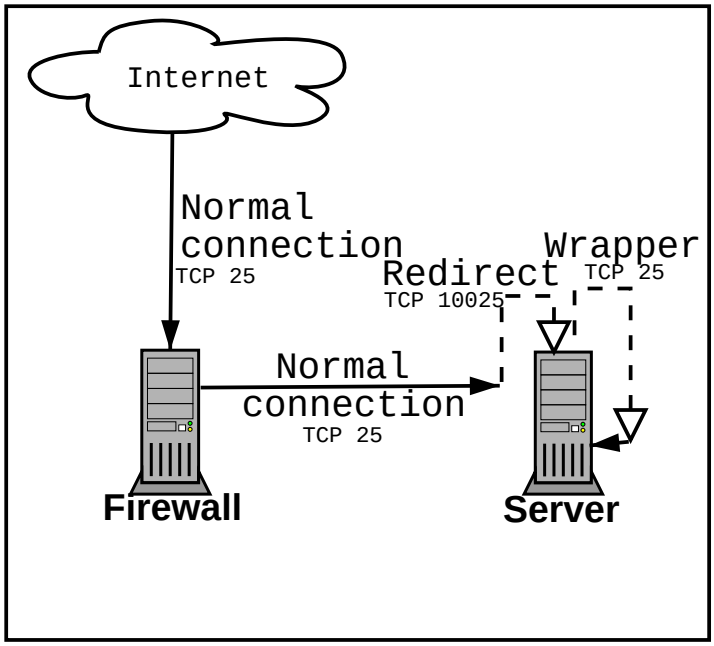
5.4 Server running at a non-standard port



At the server, the daemon's configuration is modified to listen to the port we choose. Of course, no external site on the network should be able to reach it. Then, ProtoWrap is invoked using:

```
#!/usr/bin/perl

use ProtoWrap::SMTP;
use strict;

my $wrap;

$wrap = ProtoWrap->new('standalone' => 1,
       'listenPort' => 25,
       'destType' => 'ip',
       'destAddr' => '127.0.0.1',
       'destPort' => 10025,
       'maxMsgSize' => 5000000,
       'blockAddrList' => ['spam.net','spammer@otra.org'],
       'blockBodyList' => ['Content-type.*PIF'],
       'maxRcpt' => 10,
       'relayDomainList' => ['mivecino.org'],
       'relayIpList' => ['127.0.0'],
       'logLevel' => 2
       );

die "No pude iniciar ProtoWrap" unless (defined($wrap));
$wrap->startServer() or warn 'No puedo iniciar el servidor: '.
                   $wrap->printParam();

sleep;
```

5.5 Local redirecting firewall rules at the server



At the server, we apply local redirection rules so that incoming connections for the daemon go to the port where we are running ProtoWrap, acting as a local firewall. ProtoWrap uses:

```
#!/usr/bin/perl

use ProtoWrap::SMTP;
use strict;

my $wrap;

$wrap = ProtoWrap->new('standalone' => 1,
       'listenPort' => 10025,
       'destType' => 'ip',
       'destAddr' => '192.168.0.1',
       'destPort' => 25,
       'maxMsgSize' => 5000000,
       'blockAddrList' => ['spam.net','spammer@otra.org'],
       'blockBodyList' => ['Content-type.*PIF'],
       'maxRcpt' => 10,
       'relayDomainList' => ['mivecino.org'],
       'relayIpList' => ['127.0.0'],
       'logLevel' => 2
       );

die "No pude iniciar ProtoWrap" unless (defined($wrap));
$wrap->startServer() or warn 'No puedo iniciar el servidor: '.
                  $wrap->printParam();

sleep;
```

6 End

Thank you very much

Gunnar Wolf
UNAM FES Iztacala
DSC-DGSCA
gwolf@campus.iztacala.unam.mx
http://www.gwolf.cx/wrap