

# Zero-Day Exploit Protection

## "dmz blueprint"

Ivan Bütler – 12. November 2009



### How to defend Zero Day Exploitation

At my talk at the Compass Security Conference 2009 I brought up some defense in-depth strategies to armor an internet facing server to make it more robust against zero-day exploits. I performed some practical live-demo examples and conceptual recommendations. This paper introduces a practical implementation of the recommendations and how I do it with the technology I use.

## Pre-Condition

I consider any internet facing server as vulnerable – or at least from time to time. Therefore, we need to set up these servers accordingly and should abide the following conceptual recommendations. We always expect the hacker to place malicious payload into the server's memory. The hacker's payload placed in the service's memory is expected to gain local admin privileges (which is not that easy by the way) as a working scenario.

## Conceptual defense in-depth recommendation

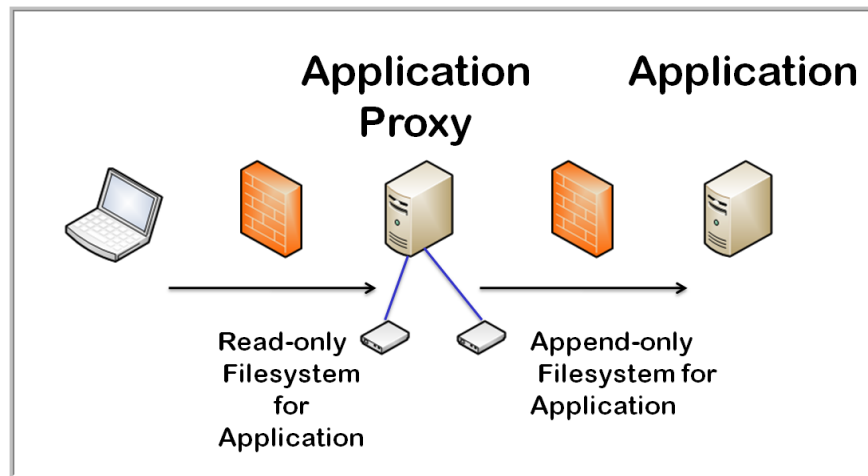
This is my recommended setup of an internet-facing server within your demilitarized zone.

1. Your Internet-facing servers should be a proxy to the backend
2. Your Internet-facing server has zero customer data (it acts as a proxy)
3. Your Internet-facing server runs the services with minimum privileges
4. Your Internet-facing server is firewalled (outside-in)
5. Your Internet-facing server is not allowed to access the Internet (inside-out)
6. Your Internet-facing server has append permissions to the log files

7. Your Internet-facing server can be rebuilt within minutes/seconds in case of an emergency (e.g. Solaris Zones, VM Snapshots)

## Recommendation 1+2

1. Your Internet-facing servers should be a proxy to the backend
2. Your Internet-facing server has zero customer data (it acts as a proxy)



In this setup a hacker's zero day shall run on the proxy server. Customer data is not available here and therefore we can reduce the impact of the exploit.

## Recommendation 3

3. Your Internet-facing server runs the services with minimum privileges

Make sure your service runs with least privileges. On our server, the apache webserver runs with wwwrun privileges, the files belong to wwwadm and root controls the keying material.

```
apache:~ # ps -ef -o uid,user,fname,comm |grep http
117 wwwrun httpd /zpool/applic/httpd-2.2.6A/bin/httpd
117 wwwrun httpd /zpool/applic/httpd-2.2.6A/bin/httpd
117 wwwrun httpd /zpool/applic/httpd-2.2.6A/bin/httpd
117 wwwrun httpd /zpool/applic/httpd-2.2.6A/bin/httpd
117 wwwrun httpd /zpool/applic/httpd-2.2.6A/bin/httpd
117 wwwrun httpd /zpool/applic/httpd-2.2.6/bin/httpd
0 root httpd /zpool/applic/httpd-2.2.6/bin/httpd
```

Additionally, the wwwrun service runs in its virtual environment. The application binaries are mounted in read-only mode.

```
add fs
set dir=/zpool/data/www
set special=/zpool/data/www
set type=lofs
add options ro
add options nodevices
end

add fs
set dir=/zpool/logs/apache
set special=/zpool/logs/apache
set type=lofs
add options rw
add options nodevices
end

add fs
set dir=/zpool/applic/openssl/openssl-0.9.8g
set special=/zpool/applic/openssl/openssl-0.9.8g
set type=lofs
add options ro
add options nodevices
end

add fs
set dir=/zpool/applic/httpd-2.2.6A
set special=/zpool/applic/httpd-2.2.6A
set type=lofs
add options ro
add options nodevices
end
```

As you can see (in red) above, the application, the static web page and the openssl library are mounted in read-only mode (ro). In such an environment, the attacker could even gain unauthorized root privileges without being able to alter the application configuration. The attacker would then need to go a step further using a local kernel exploit to bypass the defense in-depth restrictions. This is still possible, but is less likely.

Proof? In the example below I assume an exploit gained root privileges within the virtual environment. The server is hardened, so even root should not be able to alter the system. Let's try to delete the application within the apache virtual environment.

```
root@treo[global] ~ # zlogin apache
[Connected to zone 'apache' pts/2]
Last login: Thu Oct 22 01:18:43 on pts/2
Sun Microsystems Inc. SunOS 5.11 snv_54 October 2007

apache:/zpool/applic/httpd-2.2.6A # id
uid=0(root) gid=0(root)

apache:~ # rm -rf /zpool/applic/httpd-2.2.6A

rm: Unable to remove directory bin: Read-only file system
rm: Unable to remove directory build: Read-only file system
rm: Unable to remove directory cgi-bin: Read-only file system
```

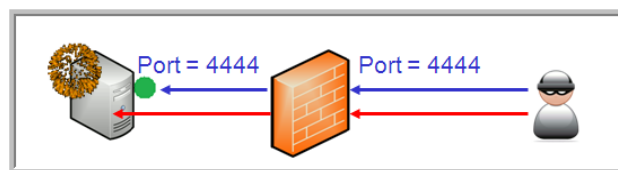
The user root (within the virtual environment "apache") is not allowed to delete the application directory (or place a Trojan or similar). This is because the filesystem is mounted in read-only mode to the virtual environment.

## Recommendation 4 + 5

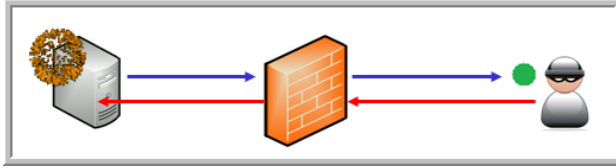
From the Metasploit Framework, we learned about tcp\_bind and reverse\_tcp shell codes and their functionality. How can we stop these payloads?

4. Your Internet-facing server is firewalled (outside-in)
5. Your Internet-facing server is not allowed to access the Internet (inside-out)

The 4<sup>th</sup> recommendation is obvious. Do not pass any others than the required ports through the firewall.



The 5<sup>th</sup> recommendation protects you against reverse\_tcp. Do not allow your demilitarized systems to reach the internet. This could be a trade-off (e.g. nameserver lookup for log files).



If you check out the zf05 advisory, where Kevin Mitnicks and Dan Kaminskys server were hacked, you will find out that especially recommendation 5 is not followed.

zf05 link: <http://antilimit.net/zf05.txt>

## Recommendation 6

Log files store sensitive information. The process owner of the running service shall append log entries, but there read permissions are not mandatory. You should use the append-only file system mode to allow your process to save log entries.

7. Your Internet-facing server has append permissions to the log files (ZFS)

```
/usr/sun/bin/chmod A1=owner@:append_data:allow

root@ctao:/zpool/logs/apache# /usr/sun/bin/ls -v
total 44
-r-----+ 1 daemon  other      10403 Oct 22 17:43 access_log

0:owner@:execute:deny
1:owner@:read_data/append_data:allow
```

## Recommendation 7

1. Your Internet-facing server can be rebuilt within minutes/seconds in case of an emergency (e.g. Solaris Zones, VM Snapshots)

In the case of Solaris10, the use of zonemgr is recommended. With this script, a non-global zone can be deleted, destroyed or re-generated in minutes! In case you do not trust your non-global zone any more, start a zonemgr shell script to rebuild. This is similar to the VMSnapshot technique.