# Do Reverse Proxies provide real security?

In the process of building / designing the infrastructure for a new project the following question was asked: “shouldn't we use a reverse proxy to secure or protect the web servers?” Of course the first question I asked myself is “**do reverse proxies provide real security?**”

After consulting with some fellow members of ISECOM (a non-profit organization based in Barcelona which among other projects has the OSSTMM), to see if they had already some experience testing environments with a similar setup. This is the answer I got from Pete Herzog, the Managing Director:

```
“It's a pretty standard set-up. But if you're using the RP (reverse proxy)
for extra *security* then I don't think you'll get it. You (would) need to
crunch the numbers to know quantity but you're increasing your Attack
Surface with it.”
```

So I setup the environment (one with and one without a reverse proxy) and we proceeded to test it. The rest of this article presents what we tested, the results we obtained, how to measure the actual security of the environment with the RAVs (crunch the numbers) and how these can be used to make a fact based comparison of both based on these results.

## The scope or environment

For the scope we decided to exclude the web applications themselves, because they will be different in each scenario. The focus has been set on the architecture and the infrastructure.

For these tests the following environment depicted in Figure 1was setup in an IaaS aka Cloud environment.

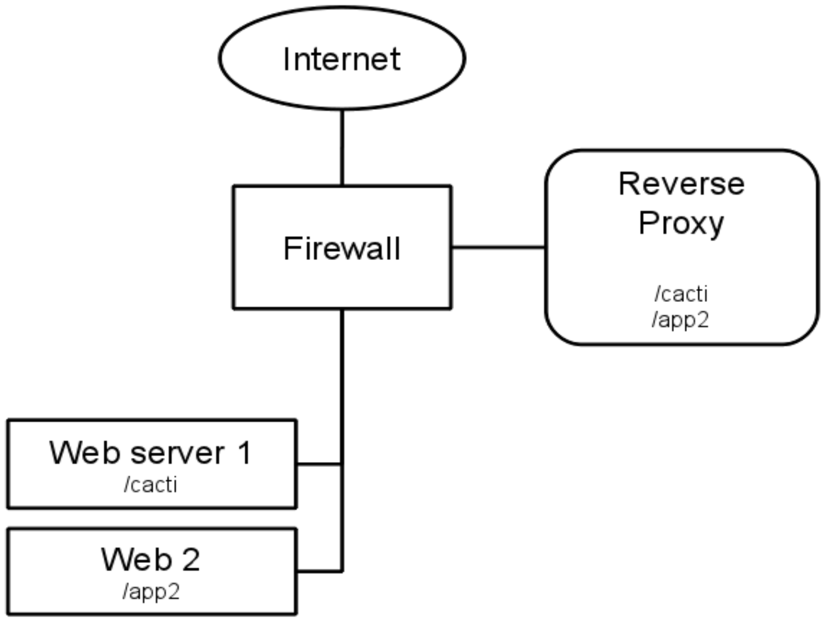


*Figure 1: Environment diagram*

We have all servers reachable individually to make both scenarios available simultaneously for testing purposes.

**Web server 1 aka cacti:** is a standard Linux server running Apache 2.22 with PHP, MySQL 5.x to serve the cacti application.

**Web server 2 aka app2:** is a standard Linux server running Apache 2.22, serving a static html site.

**Reverse Proxy aka RP:** is a standard Linux server running Apache 2.22 with mod_proxy_html enabled and configured as a reverse proxy server.

The firewall in the environment allows pings (ICMP echo-request / reply), SSH (22/TCP), HTTP (80/TCP) and HTTPS (443/TCP), while all other ports and services are denied.

All of the servers are configured to work only with HTTPS and the default configuration has been used for all other configuration items. After the initial rounds of tests we found that the web servers were disclosing too much information in the HTTP-headers, error page footers (see Codebox 1 and the server banner:

```
Apache/2.2.22 (Amazon) Server at 10.251.10.187 Port 443
```

*Codebox 1: Error page footer*

The following minor configuration tweaks were made to make it a more reasonable scenario.

```
In /etc/httpd/conf/httpd.conf:

ServerSignature Off
ServerTokens Prod


In the /etc/php.ini:
expose_php=Off
```

*Codebox 2: simple configuration changes in Apache and PHP configuration files*

## Tests and analysis

From here on, the following names will be used for the different systems:

| | |
|---|---|
| **Cacti** | https://176.x.y.218/cacti/ |
| **App2** | https://54.u.v.162/app2/ |
| **RP** | https://54.u.p.245/cacti/ and https://54.u.p.245/app2/ |

We used the OSSTMM 3 as the basis for the testing so we could measure the Attack Surface. Even though, measuring the Attack Surface with RAVs (the method outlined in the OSSTMM) seems complicated at first, it is actually pretty straight-forward once you understand the concepts.

Full disclosure, myself and all the testers I asked to test this scenario are OPST (www.opst.org) and OPSA (www.opsa.org) certified to assure we all knew how to do it right.

The OSSTMM 3 uses interactions and not risk as its base for defining security. Some of these interactions are between objects within the security scope so they are defined as Trusts. When the interactions happen between the inside and the outside of the security scope, they are defined as Accesses. Interactions which are purely informational and serve only to expose what is in the security scope are designated as Visibilities. These three things make up a scope's Porosity which is its base Attack Surface.

**Porosity = Visibility + Access + Trust**

While measuring the Porosity, we also look for Controls, the types of protection mechanisms used within the scope, and Limitations which are types of vulnerabilities based on cause/effect. These things influence the size of the Attack Surface as well and help us calculate the final RAV a unit which shows the percentage of protection where 100% is a perfect balance between security and operations.

The OSSTMM also defines 10 types of operational controls and 5 types of Limitations. With a little practice, they become easy to identify while doing a test. Here we include the Controls and Limitations we found while measuring the Porosity. Then in the final RAV if it's above 100% we know we have more security than we need which costs us more to build and maintain and if it's under then we know we have too little. One of the other key points of the RAVs is seeing if we have the right controls in place to handle a variety of threats or if we have too many of just one type of control which might make the infrastructure appear stronger but only against specific threats while simultaneously being weaker against many other types.

### *Operational Controls:*

| Class A: Interactive controls | Class B: Process controls |
|---|---|
| **Authentication** | Non-Repudiation |
| **Indemnification** | Confidentiality |
| **Resilience** | Privacy |
| **Subjugation** | Integrity |
| **Continuity** | Alarm |

### *Limitations:*

| Category | | OpSec | Limitations |
|---|---|---|---|
| **Operations** | | Visibility | Exposure |
| | | Access | Vulnerability |
| | | Trust | |
| **Controls** | Class A Interactive | Authentication | Weakness |
| | | Indemnification | |
| | | Resilience | |
| | | Subjugation | |
| | | Continuity | |
| | Class B Process | Non-Repudiation | Concern |
| | | Confidentiality | |
| | | Privacy | |
| | | Integrity | |
| | | Alarm | |
| | | | Anomalies |

## *Access*

For this test, we don't count interactions in the web application logic because our scope and focus is the infrastructure and not the application.

In this case to measure Access, we used various port scanners to do the job: nmap and Unicornscan.

```
bash# nmap -T4 -p 0-65535 -sV cacti
Nmap scan report for cacti (176.x.y.218)
Host is up (0.030s latency).
PORT    STATE    SERVICE  VERSION
22/tcp  open     ssh      OpenSSH 5.3 (protocol 2.0)
80/tcp  filtered http
443/tcp open     ssl/http Apache httpd
```

*Codebox 3: output of nmap TCP scan*

For the UDP scans, two approaches were taken: one using nmap and testing all ports which is really time intensive and one using Unicornscan. This last tool is much more efficient for this type of test.

```
bash# nmap -sU -T4 -p 0-65353 cacti

bash# unicornscan -mU -vv -Ir 100 cacti:a
```

*Codebox 4: tools that can be used for UDP scans*

The following results were obtained:

**Cacti**

- open ports: 22 /TCP, 443 /TCP
- ICMP echo replies
- no closed ports
- no UDP

**App2**

- open ports: 22 /TCP, 443 /TCP
- ICMP echo replies
- no closed ports
- no UDP

**RP**

- open ports: 22 /TCP, 443 /TCP
- ICMP echo replies
- no closed ports
- no UDP

The systems respond with TCP SYN ACK on open ports, but no TCP RST packets are received outside the scope. But because SYN ACK's are received, the filtered status for other ports can't be explained by server down, unreachable etc. This means that the packets are dropped by a firewall, an ACL in a router or any other packet filter.

This firewall/packet filter is counted as 1x the Authentication for all hosts.

Since in the RP scenario, it is the only server directly reachable, the Access must be counted separately for each case

**Access count for RP scenario: 3**
RP has 3 accesses: 2x open TCP ports (22 and 443) and 1x ICMP.

**Access count for direct scenario: 6**
Each web server has 3 accesses: 2x open TCP ports (22 and 443) and 1x ICMP.

### *Visibility*

All systems are exposed, so the Visibility is at least one for each of the systems.

To detect a reverse proxy server, we connected to the target and retrieved the page /cacti/ or /app2/. When the Max-Forwards request header was set to 1, the responses for Cacti and RP were similar and the responses for App2 and RP were similar. When the Max-Forwards request header was set to 0, the responses were different. Cacti and App2 replied similar as when the Max-Forwards request header was set to 1, but RP replied with a **502 Proxy Error**.

This test proved the presence of a reverse proxy server. The example here is shown for /cacti/ on RP. (Funny enough, the TTL on arrival is different for the different values of Max-Forwards).

This can be done by multiple ways. One approach would be to do it by hand using OPENSSL as shown in Codebox 5 and another would be by using Firefox and TamperData. A screenshot of the error message can be seen in Figure 2

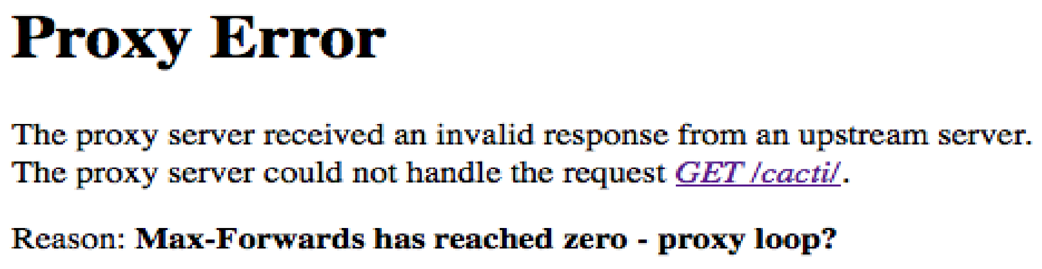
```
$ openssl s_client -connect 54.u.p.245:443
...snip...
GET /cacti/ HTTP/1.0
Max-Forwards: 0


HTTP/1.1 502 Proxy Error
...snip...


$ openssl s_client -connect 54.u.p.245:443
...snip...
GET /cacti/ HTTP/1.0
Max-Forwards: 1


HTTP/1.1 200 OK
...snip...
<html><head><title>Login to Cacti</title><style type="text/css">
...snip...
```

*Codebox 5: extracts of manually testing HTTPS requests*



Proxy Error

The proxy server received an invalid response from an upstream server.
The proxy server could not handle the request GET /cacti/.

Reason: **Max-Forwards has reached zero - proxy loop?**

*Figure 2: Proxy error*

Because you can interact with the RP, the RP is visible.
Because you can interact with the web server via RP, the web server is visible.

By running a sniffer (tcpdump) alongside the ports cans it was possible to identify different TTL values in the IP headers from the responses from /app2/ and /cacti/. Based on our analysis, one explanation for this would be that there are different distances from server to client, which means there have to be different kernels. Another explanation is there are different paths from server to client, which is unlikely if both sites are serviced by the same host. This is why I chose the former to be most likely.

As a conclusion the Visibility of RP is now a total of 3 (2 web servers and a reverse proxy server).

**Visibility count for RP scenario: 3**
RP and two web servers are visible.

**Visibility count for direct scenario: 2**
It is possible to interact with two different web servers or at least two different IPs.

A reverse proxy provides the operational controls of Privacy by hiding the actual server you are connecting to and Subjugation, which is what, makes a control mandatory, meaning the user has no choices regarding the use of that control. Sometimes it provides Authentication as well although Authentication has not proven here, so it won't be counted.

## *HTTPS*

All servers speak HTTPS. Each server provides following operational controls: 1x Confidentiality, 1x Integrity and 1x Subjugation.

The ciphers accepted by the HTTPS server can be checked by testing each of them manually with OPENSSL. A shortcut would be to use SSLYZE, a python script that does precisely that, as can be seen in Codebox 6

The server supports 56 bit ciphers. Such ciphers are considered weak and it is counted as a Concern on Confidentiality, a Limitation which increases the Attack Surface slightly. No other limitations were found in this implementation of the HTTPS protocol.

```
bash$ python sslyze.py --regular rp:443

...snip...
* SSLV2 Cipher Suites :
     Cipher Suite:                 SSL Handshake:      HTTP GET:
     DES-CBC3-MD5   168bits        Preferred           200 OK
     ...snip...
     DES-CBC-MD5    56bits         Accepted            200 OK
```

*Codebox 6: extracts of the output of SSLYZE*

**Note: HTTPS provides** the control of **Confidentiality** (where the **content is protected**) **and not Privacy** (where the **process is protected**). This means a 3rd party can see when you are using HTTPS, where you are attaching to it and where you are coming from (well, the endpoints at least) but they can't see what you are sending or receiving exactly.

## *SSH*

All servers speak SSH. Each server provides: 1x Confidentiality, 1x Integrity and 1x Subjugation. SSHv1.x is not supported, so this is not counted as a limitation.

The SSH servers are disclosing version information via its banner, as seen in the results of the nmap scans.

```
22/tcp  open     ssh      OpenSSH 5.3 (protocol 2.0)
```

*Codebox 7: extracts of the nmap report*

This is counted as a Limitation on Visibility and therefore as an Exposure.

**Note: SSH** also **provides** the control of **Confidentiality** (where the **content is protected**) **and not Privacy** (where the **process is protected**).

## *Trusts*

Trusts are as important to an Attack Surface as Access because it allows for indirect interactions especially where deception is used to get the “trusting” target to accept the unauthorized connection. In this case, the web server connects to the database server (a Trust interaction) by using a login and password, which is counted as 1 x Authentication and 1 x Subjugation.

The web server can either trust the proxy and process all requests forwarded by it without further inspection (which is the case and also the standard) or not. An Authentication process (i.e. based on certificates) could also be used.

**Trust count for RP scenario:** **2**
1 count for each web server that trusts requests from the proxy.

**Trust count for direct scenario:** **0**

### *Non-Repudiation*

No access was granted to the log files (web server, MySQL server or SSH) and because of that, initially we didn't count them as Non-Repudiation.

In case of the RP, it is the RP that connects to the web servers, so there is no control of Non-Repudiation. If there is a proxy-log, you can use that one as Non Repudiation for the web services. But not for SSH. Only the RP accepts SSH connections, so the SSH log only provides the control of Non-Repudiation to RP and not to Cacti or App2.

In case of direct access to the server you can count these as Non-Repudiation controls (assuming the log level is extensive enough and you don't need a "court-legal" type of Non-Repudiation).

MySQL (the database used by cacti) writes decent log files for this per default, so this can be counted as a Non-Repudiation control.

**Note:** To have true "court-legal" Non-Repudiation you need to ensure the **Integrity** of the log files can be verified later. More specifically, if someone (an admin or an attacker) alters the log files, it has to be detectable. But in general, if there is any kind of identifying of connection source and file interaction in logging, it should be counted as Non-Repudiation anyway, even though the legal evidence is weak.

**Non-repudiation count for RP scenario: 3**
1 count for the proxy log + 1 count for the SSH log on the RP + 1 count for the MySQL log.

**Non-repudiation count for direct scenario: 5**
For each web server: 1 count for the Apache log + 1 count for the SSH log. + 1 count for the MySQL log.

At this point we found no further controls. It is however left open for us to know which controls have already been applied to specific interactions. Should we later determine that our Attack Surface is too high, we have the option of increasing the number of controls implemented as well as removing vulnerabilities.

## Limitations

Limitations should be counted only if they have been proven. So all the "vulnerabilities" highlighted by the tests should only be counted if the exploit can been used and the problem is verified.

## RAV counts per scenario

This section presents the summary of all counts for each scenario. After accounting for the analysis, we enter the numbers into in the RAV spreadsheet provided by ISECOM (http://www.isecom.org/research/ravs.html) to help make the final calculations. Here's how the numbers crunched for the 2 scenarios look like:

### *Scenario 1:  Reverse Proxy (RP)*

| | |
|---|---|
| **Visibility:** | 3 – 1 reverse proxy and 2 web servers |
| **Access:** | 3 - ICMP, 22 /TCP, 443 /TCP |
| **Trust**:: | 2 – 2 x web servers that trust the proxy |
| **Authentication:** | 3 - firewall, 2 x SSH login, MySQL login. |
| | (Cacti web login is not counted because the app is not in scope) |
| **Subjugation:** | 4 - firewall, reverse proxy: HTTPS, SSH, MySQL login |
| **Non-Repudiation:** | 3 - proxy log, SSH log, MySQL log |
| **Confidentiality:** | 2 - HTTPS, SSH |
| **Privacy:** | 1 - reverse proxy |
| **Integrity:** | 2 - HTTPS + SSH |
| **Concern:** | 1 - 56 bit ciphers in HTTPS |
| **Exposures:** | 2 - Internal IP can be found in SSL certificate, SSH version in banner |
| **Total RAV:** | **88.3%** |

### *Scenario 2: Direct access to web servers*

| | |
|---|---|
| **Visibility:** | 2 – 2 x web servers |
| **Access:** | 6 – 2 x (ICMP, 22 /TCP, 443 /TCP) |
| **Trust:** | 0 |
| **Authentication:** | 4 - firewall, 2 x SSH login, 1x MySQL auth. |
| | (Cacti web login is not counted because the app is not in scope) |
| **Subjugation:** | 6 - firewall, 2 x (HTTPS, SSH), 1 x MySQL auth |
| **Non-Repudiation:** | 4 – 2 x (apache log, SSH log), 1 MySQL log |
| **Confidentiality:** | 4 – 2 x (HTTPS + SSH) |
| **Integrity:** | 2 – 2 x (HTTPS + SSH) |
| **Concern:** | 2 – 2x 56 bit ciphers in HTTPS |
| **Exposures:** | 4 – 2 x (Internal IP can be found in SSL certificate, SSH version in banner) |
| **Total RAV:** | **87.07%** |

## Conclusion

This is the result of the security measurements for both scenarios: using a RP (reverse proxy) and then without it while the web servers have limitations:

**RP:**

| | |
|---|---|
| **Total RAV** | **88.3%** |

**Direct:**

| | |
|---|---|
| **Total RAV** | **87.07%** |

This shows that the RP has a smaller Attack Surface in this scenario. Don't get fooled by the small difference of 1.23%. Think of a building with 1% of its doors or windows open. Or how even in a risk assessment of potential vulnerable exposure, this 1% on the grand scale of thousands or millions of connections day after day will add up quickly.

However, I had no intention of building an infrastructure with limitations already. So we fixed the problems found and recalculated with the limitations removed. This is how it now compares:

**RP:**

| | |
|---|---|
| **Total RAV without limitations:** | **95.92%** |

**Direct:**

| | |
|---|---|
| **Total RAV without limitations:** | **96.52%** |

What is interesting here is that the balance now tilts the other way, favouring the scenario using direct access to web servers. This means that when considering a scope with a non-optimal configuration of web servers, it can be better protected by using a reverse proxy server. But if they are correctly configured, then the reverse proxy adds to the visibility and expands the Attack Surface.

### *But how does is Scale?*

As an exercise we calculated the RAVs for bigger scenarios to see if the conclusion also scales, with 100 web servers behind a reverse proxy or directly exposed. Again by taking advantage of the RAVs we can measure the security in both scenarios without actually having to build them.

| | |
|---|---|
| RAV of a RP without limitations protecting 100 servers: | 85.29% |
| RAV of 100 web server with limitations: | 66.33% |
| RAV of 100 web servers without limitations: | 91.44% |

Adding servers to the scope highlights the conclusion of our research: **having a reverse proxy does increase the Attack Surface**.

At the end, it comes down to a risk decision: If you think your organization can run securely configured, hardened web servers over time (and keep them that way) you are better off without the proxy. If not, you should have a reverse proxy in place as an umbrella to protect all those web servers or devices that are not maintained securely.

## Contributors

Pablo Endres
Cor Rosielle, Lab 106
Pete Herzog, ISECOM
Lars Heidelberg, adMERITia GmbH

## About the author

Pablo Endres is an Experienced Security Consultant, Linux / UNIX system administrator and a technological solution architect. His main interests are information security, networking, telephony, *NIX and technological research. He holds a degree in computer engineering, ISC2 CISSP, Comptia Security+, ISECOM's OPSA + OPST Certifications, and a black belt in Shuri-Ryu karate-do; and has worked in a variety of industries: wireless phone providers, VoIP solution providers, contact centers, university labs, and founded a IT consulting company. Pablo enjoys reverse engineering, research, self learning, an intellectual challenge, and collaboration.

Published: 12.09.2012