1 Protecting Your Site
1.1 Description

Securing your site should be your first priority, because of the consequences a break-in might have. We discuss the various authentication and authorization techniques available, a very interesting use of mod_perl.

1.2 The Importance of Your site’s Security

Let’s face it, your site or service can easily become a target for Internet "terrorists". It can be because of something you said, the success of your site, or for no obvious reason whatever. If your site security is compromised, all your data can be deleted or important information can be stolen. You may risk legal action or the sack if this happens.

Your site can be paralyzed through a _simple_ denial of service (DoS) attack.

Whatever you do, as long as you are connected to the network your site will be vulnerable. Cut the connections, turn off your machine and put it into a safe. Now it is protected--but useless.

So what can you do?

Let’s first get acquainted with some security related terminology:

- **Authentication**
  
  When you want to make sure that a user is who he claims to be, you generally ask her for a username and a password. Once you have both, you can check them against your database of username/password pairs. If they match, the user has passed the **Authentication** stage. From now on if you keep the session open all you need to do is to remember the username.

- **Authorization**
  
  You might want to allow user **foo** to have access to some resource, but restrict her from accessing another resource, which in turn is accessible only for user **bar**. The process of checking access rights is called **Authorization**. For **Authorization** all you need is an authenticated username or some other attribute which you can authorize against. For example, you can authorize against IP number, allowing only your local users to use some service. But be warned that IP numbers or session_ids can be spoofed (forged), and that is why you should not do **Authorization** without **Authentication**.

Actually you’ve been familiar with both these concepts for a while.

When you telnet to your account on some machine you go through a login process (**Authentication**).

When you try to read some file from your file systems, the kernel checks the permissions on this file (**Authorization**). You may hear about Access control which is another name for the same thing.
1.3 Illustrated Security Scenarios

I am going to present some real world security requirements and their implementations.

1.3.1 Non authenticated access for internal IPs, Authenticated for external IPs

An Extranet is very similar to an Intranet, but at least partly accessible from outside your organization. If you run an Extranet you might want to let your internal users have unrestricted access to your web server. If these same users call from outside your organization you might want to make sure that they are in fact your employees.

These requirements are achieved very simply by putting the IP patterns of the organization in a Perl Access Handler in an .htaccess file. This sets the REMOTE_USER environment variable to the organization’s generic username. Scripts can test the REMOTE_USER environment variable to determine whether to allow unrestricted access or else to require authentication.

Once a user passes the authentication stage, either bypassing it because of his IP address or after entering a correct login/password pair, the REMOTE_USER variable is set. Then we can talk about authorization.

Let’s see the implementation of the authentication stage. First we modify httpd.conf:

```httpd.conf
PerlModule My::Auth
<Location /private>
    PerlAccessHandler My::Auth::access_handler
    PerlSetVar Intranet "10.10.10.1 => userA, 10.10.10.2 => userB"
    PerlAuthenHandler My::Auth::authen_handler
    AuthName realm
    AuthType Basic
    Require valid-user
    Order deny, allow
    Deny from all
</Location>
```

Now the code of My/Auth.pm:

```perl
sub access_handler { 
    my $r = shift;

    unless ($r->some_auth_required) {
        $r->log_reason("No authentication has been configured");
        return FORBIDDEN;
    }

    # get list of IP addresses
    my $ips = split /\s*(?:=>|,\s*)\s*/, $r->dir_config("Intranet");

    if (my $user = $ips($r->connection->remote_ip)) {
        # update connection record
```

15 Feb 2014
You can implement your own `authen_dbi()` routine, or you can replace `authen_handler()` with an existing authentication handler such as Apache::AuthenDBI.

If one of the IP addresses is matched, `access_handler()` sets `REMOTE_USER` to be either `userA` or `userB`.

If neither IP address is matched, `PerlAuthenHandler` will not be set to OK, and the Authentication stage will ask the user for a login and password.
1.4 Authentication code snippets

1.4.1 Forcing re-authentication

To force an authenticated user to reauthenticate just send the following header to the browser:

```plaintext
WWW-Authenticate: Basic realm="My Realm"
HTTP/1.0 401 Unauthorized
```

This will pop-up (in Netscape at least) a window saying Authorization Failed. Retry? with OK and a Cancel buttons. When that window pops up you know that the password has been discarded. If the user hits the Cancel button the username will also be discarded. If she hits the OK button, the authentication window will be brought up again with the previous username already in place.

In the Perl API you would use the note_basic_auth_failure() method to force reauthentication.

This may not work! The browser’s behaviour is in no way guaranteed.

1.4.2 OK, AUTH_REQUIRED and FORBIDDEN in Authentication handlers

When your authentication handler returns OK, it means that user has correctly authenticated and now $r->connection->user will have the username set for subsequent requests. For Apache::Registry and friends, where the environment variable settings weren’t erased, an equivalent $ENV{REMOTE_USER} variable will be available.

The password is available only through the Perl API with the help of the get_basic_auth_pw() method.

If there is a failure, unless it’s the first time, the AUTH_REQUIRED flag will tell the browser to pop up an authentication window, to try again. For example:

```perl
my ($status, $sent_pw) = $r->get_basic_auth_pw;
unless($r->connection->user and $sent_pw) {
    $r->note_basic_auth_failure;
    $r->log_reason("Both a username and password must be provided");
    return AUTH_REQUIRED;
}
```

Let’s say that you have a mod_perl authentication handler, where the user’s credentials are checked against a database. It returns either OK or AUTH_REQUIRED. One of the possible authentication failure case might happen when the username/password are correct, but the user’s account has been suspended temporarily.

If this is the case you would like to make the user aware of this, by displaying a page, instead of having the browser pop up the authentication dialog again. You will also refuse authentication, of course.
The solution is to return `FORBIDDEN`, but before that you should set a custom error page for this specific handler, with help of `$r->custom_response`. It looks something like this:

```perl
use Apache::Constants qw(:common);
$r->custom_response(SERVER_ERROR, "/errors/suspended_account.html");
return FORBIDDEN if $suspended;
```

### 1.5 Apache::Auth* modules

#### PerlAuthenHandler’s

- **Apache::AuthAny** Authenticate with any username/password
- **Apache::AuthenCache** Cache authentication credentials
- **Apache::AuthenCookie** Authen + Authz via cookies
- **Apache::AuthenDBI** Authenticate via Perl’s DBI
- **Apache::AuthExpire** Expire Basic auth credentials
- **Apache::AuthenGSS** Generic Security Service (RFC 2078)
- **Apache::AuthenIMAP** Authentication via an IMAP server
- **Apache::AuthenPasswdSrv** External authentication server
- **Apache::AuthenPasswd** Authenticate against `/etc/passwd`
- **Apache::AuthLDAP** LDAP authentication module
- **Apache::AuthPerLDAP** LDAP authentication module (PerLDAP)
- **Apache::AuthenNIS** NIS authentication
- **Apache::AuthNISPlus** NIS Plus authentication/authorization
- **Apache::AuthenRaduis** Authentication via a Radius server
- **Apache::AuthenSmb** Authenticate against NT server
- **Apache::AuthenURL** Authenticate via another URL
- **Apache::DBILogin** Authenticate to backend database
- **Apache::DCELogin** Obtain a DCE login context
- **Apache::PHLogin** Authenticate via a PH database
- **Apache::TicketAccess** Ticket based access/authentication

#### PerlAuthzHandler’s

- **Apache::AuthCookie** Authen + Authz via cookies
- **Apache::AuthzAge** Authorize based on age
- **Apache::AuthzDCE** DFS/DCE ACL based access control
- **Apache::AuthzDBI** Group authorization via Perl’s DBI
- **Apache::AuthzGender** Authorize based on gender
- **Apache::AuthzNIS** NIS authorization
- **Apache::AuthzPasswd** Authorize against `/etc/passwd`
- **Apache::AuthzSSL** Authorize based on client cert
- **Apache::RoleAuthz** Role-based authorization

#### PerlAccessHandler’s

- **Apache::AccessLimitNum** Limit user access by number of requests
- **Apache::BlockAgent** Block access from certain agents
- **Apache::DayLimit** Limit access based on day of week
- **Apache::IPThrottle** Limit bandwidth consumption by IP
- **Apache::RobotLimit** Limit access of robots
- **Apache::SpeedLimit** Control client request rate
1.6 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- Stas Bekman [http://stason.org/]

1.7 Authors

- Stas Bekman [http://stason.org/]

Only the major authors are listed above. For contributors see the Changes file.
# Table of Contents:

1. **Protecting Your Site** .......................................................... 1  
   1.1 **Description** ................................................................. 2  
   1.2 **The Importance of Your site’s Security** ............................. 2  
   1.3 **Illustrated Security Scenarios** ....................................... 3  
   1.3.1 **Non authenticated access for internal IPs, Authenticated for external IPs** .............................. 3  
   1.4 **Authentication code snippets** ......................................... 5  
   1.4.1 **Forcing re-authentication** ......................................... 5  
   1.4.2 **OK, AUTH_REQUIRED and FORBIDDEN in Authentication handlers** .............................. 5  
   1.5 **Apache::Auth* modules** ................................................ 6  
   1.6 **Maintainers** ................................................................. 7  
   1.7 **Authors** ................................................................. 7