Sommaire

1 Qui sommes nous ?

La communauté zenk-security a pour objet principal la sécurité informatique, nous sommes des touches à tout, des fouineurs, nous expérimentons à tout va et nous partageons sans autre restriction que le respect.

Notre site répertorie nos tutos, articles informatifs et autres textes techniques ou non, c'est le côté partage de notre communauté.
Pourtant, et vous vous en rendrez vite compte, nous cultivons la discrétion et la qualité, le principal contenu de notre forum n'est accessible qu'aux membres de notre communauté.

La raison est simple : certains savoirs ne sont pas à placer entre toutes les mains.

Quid des acharnés d'une utopie du partage alors?

Notre position est ambiguë nous devons l'admettre, nous prônons le partage des connaissances sans restriction, c'est la pierre angulaire de la communauté, mais nous ne sommes pas aveugles au point de penser que tous ont l'intelligence ou la maturité nécessaire à l'utilisation judicieuse d'un savoir qui par définition est neutre.
Fort du constat que la connotation du savoir dépend avant tout de la moralité et de la franchise envers lui même de l'utilisateur, nous préférons réserver ce savoir pour ceux qui sont aptes à l'utiliser pour le bien commun.

Arbitraire, certes, mais avez vous mieux à proposer?

Le savoir est une arme autant que les mots ou l'acier et nous ne sommes pas une armurerie.

La communauté n'est pas considérée par ses membres comme un énième lieu de leech à tout va, nous partageons réellement et notre credo, notre dogme, c'est d'apporter ce que nous pouvons, dans la mesure de nos moyens et de nous élever grâce aux contributions des autres membres.

Du partage naît l'apprentissage et de l'apprentissage naît le partage, nous ne cherchons pas à savoir qui de l'un a engendré l'autre en premier, nous nous contentons d'entretenir la boucle ainsi formée et de progresser en nous aidant les uns les autres, simplement.

2 Nos audits de sécurité

Au sein de la communauté nous utilisons le nom de Zenk Roulette, le principe est simple nous choisissons une application open source que nous installons sur nos serveurs, à partir de là nous commençons un audit de sécurité sur l'application choisie.
Cet audit est fait exclusivement pour le fun, c'est un plaisir avant tout et il reste entièrement privé.
Les audits sont fait par des professionnelles et des passionnés du monde de la sécurité informatique.
Suite à cet audit nous fournissons un rapport aux "propriétaires" de l'application lui fournissant quelques conseils, ensuite nous attendons sa réponse par mail et l'application de correctif sous une période correcte avant de rendre publique notre rapport.
Généralement si au bout d'un mois nous n'avons pas de réponse des propriétaires nous rendons publique le rapport, dans le cas contraire nous nous arrangeons avec les propriétaires pour le rendre publique une fois les vulnérabilités corrigées.
Bien sur nous restons disponible pour toute question.

3 Notre choix

Application auditée : Cacti 0.8.7g
Description : Outil de monitoring
URL : http://www.cacti.net/
Date : Mercredi 8 novembre 2010

4 Les participants

kr0ch0u

K1wy

Essandre

Barbch

5 Compte rendu de l'audit

Fichier : auth_changepassword.php

ligne : 57

```
header("Location: " . $_POST["ref"]); break;
```

Faille de redirection

ligne : 40

Multiple Http splitting dans le switch

Toutes les fonction form_save() sont vulnérables à des SQLi. Certaines injections SQL sont protégées par un addslashes, mais avec un charset et quelques tours de magies, cela saute. Ce n'est pas une protection suffisante.

Fichier : auth_changepassword.php

```
<input type="hidden" name="ref" value="<?php print $_REQUEST["ref"];?>">
```

Faille XSS

Fichier : auth_login.php

ligne : 195

```
if (isset($_SERVER["HTTP_REFERER"])) {
    $referer = $_SERVER["HTTP_REFERER"];
    if (basename($referer) == "logout.php") {
        $referer = "index.php";
    }
} else if (isset($_SERVER["REQUEST_URI"])) {
    $referer = $_SERVER["REQUEST_URI"];
    if (basename($referer) == "logout.php") {
         $referer = "index.php";
    }
    } else {
         $referer = "index.php";
    }
    header("Location: " . $referer);
```

Faille HTTP splitting

ligne : 265

```
<form name="login" method="post" action="<?php print
basename($_SERVER["PHP_SELF"]);?>">
```

Redirection possible vers un site malicieux pour récupérer le login et le pass d'un user.

Fichier : cdef.php

ligne : 50

```
case 'item_movedown':
    item_movedown();
    header("Location: cdef.php?action=edit&id=" . $_GET["cdef_id"]);
    break;
case 'item_moveup':
    item_moveup();
    header("Location: cdef.php?action=edit&id=" . $_GET["cdef_id"]);
    break;
case 'item_remove':
    item_remove();
    header("Location: cdef.php?action=edit&id=" . $_GET["cdef_id"]);
```

Faille HTTP Splitting

ligne : 125

```
header("Location: cdef.php?action=edit&id=" . (empty($cdef_id) ? $_POST["id"] :
$cdef_id));
```

ligne : 147

```
header("Location: cdef.php?action=item_edit&cdef_id=" . $_POST["cdef_id"] .
"&id=" . (empty($cdef_item_id) ? $_POST["id"] : $cdef_item_id));
header("Location: cdef.php?action=edit&id=" . $_POST["cdef_id"]);
```

ligne : 112 & 130

```
$save["hash"] = get_hash_cdef($_POST["id"]);
```

Faille HTTP Splitting

Fichier : lib/functions.php

```
function get_hash_cdef($cdef_id, $sub_type = "cdef") {
    $hash = db_fetch_cell("select hash from cdef where id=$cdef_id");
}
```

ligne : 127

```
$sequence = get_sequence($_POST["id"], "sequence", "cdef_items", "cdef_id=" .
$_POST["cdef_id"]);
```

ligne : 116 & 137

```
$cdef_id = sql_save($save, "cdef");
```

Fichier : database.php

ligne : 325

```
$replace_result = $db_conn->Replace($table_name, $array_items, $key_cols, FALSE,
$autoinc);
_adodb_replace($this, $table, $fieldArray, $keyCol, $autoQuote, $has_autoinc);
```

Voir la fonction pour plus d'info, mais autoQuote est passé à faux là.

ligne : 175

```
duplicate_cdef($selected_items[$i], $_POST["title_format"]);
```

Car dans le Fichier : lib/utility.php **ligne :** 521

```
$cdef["name"] = str_replace("<cdef_title>", $cdef["name"], $cdef_title);

sql_save($save, "cdef_items");
```

ligne : 536

```
ORDER BY " . get_request_var_request("sort_column") . " " .
get_request_var_request("sort_direction") .
```

sanitize_search_string est inneficace dans ce cas là.

ligne : 239

```
<input type='hidden' name='drp_action' value='" . $_POST["drp_action"] . "'>
```

Faille XSS

ligne : 503

```
<input type="text" name="filter" size="40" value="<?php print
htmlspecialchars(get_request_var_request("filter"));?>">
```

Faille XSS possible avec un bon vector, à vérifier.

Fichier : color.php

ligne : 62

```
$save["id"] = $_POST["id"];
$color_id = sql_save($save, "colors");
```

Une SQL injection

Fichier : data_input.php

ligne : 88

```
$save["name"] = form_input_validate($_POST["name"], "name", "", false, 3);
$save["input_string"] = form_input_validate($_POST["input_string"],
"input_string", "", true, 3);
```

```
$save["type_id"] = form_input_validate($_POST["type_id"], "type_id", "", true,
3);
$data_input_id = sql_save($save, "data_input");
```

La même avec la ligne : 119

ligne : 555

```
ORDER BY " . get_request_var_request('sort_column') . " " .
get_request_var_request('sort_direction') . "
```

ligne : 220

```
<input type='hidden' name='drp_action' value='" . $_POST["drp_action"] . "'>
```

Faille XSS

Fichier : data_queries.php

```
header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
    break;
    case 'item_movedown_dssv':
        data_query_item_movedown_dssv();
        header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
        break;
    case 'item_remove_dssv':
        data_query_item_remove_dssv();
        header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
        break;
    case 'item_moveup_gsv':
        data_query_item_moveup_gsv();
header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
        break;
    case 'item_movedown_gsv':
        data_query_item_movedown_gsv();
        header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
        break;
    case 'item_remove_gsv':
        data_query_item_remove_gsv();
        header("Location: data_queries.php?action=item_edit&id=" .
$_GET["snmp_query_graph_id"] . "&snmp_query_id=" . $_GET["snmp_query_id"]);
        break;
    case 'item_remove':
        data_query_item_remove();
        header("Location: data_queries.php?action=edit&id=" .
$_GET["snmp_query_id"]);
```

Faille HTTP Splitting

Mauvais code :

```
function sql_sanitize($value) {
    //$value = str_replace("'", "''", $value);
    $value = str_replace(";", "\;", $value);
    return $value;
}

if (empty($regexp_match)) { $regexp_match = ".*"; }
    if ((!ereg($regexp_match, $field_value) || (($allow_nulls == false) &&
($field_value == "")))) {
        raise_message($custom_message);
}
```

Voila !

Au bout de quelques heures on a commencé a abandonner car il serait plus rapide de refaire toutes l'application plutot que d'essayer de patcher.

Donc, il doit rester plein de failles etc.. mais on n'a pas eu le courage de continuer !