Sprinkler: A Multi-Service Password Sprayer

Prompt Eua-anant, Sam Ederington

Introduction

Would you like to find the password for your mom's X account? A password sprayer is your friend.

In a nutshell, a password sprayer...

- Accepts wordlists of common usernames and passwords
- Sends a bunch of usernames and passwords to a remote server **rapidly**
- Then tells you which ones grant you access

Why should you care?

- Hackers use password sprayers to **gain unauthorized access** to accounts with weak passwords
- If you own a server that allows users to authenticate themselves, you want to know how a password sprayer works, and how to protect against password-spraying attacks.
- A password sprayer can help you learn about password-spraying tactics and see how easy it is to gain unauthorized access to someone's account on your server.

Our goal

- We made Sprinkler, a password sprayer written in C, that supports the following services: ssh, http-get (basic auth), and http-post
- We set up a target server compatible with all those services above.

What is Sprinkler?

- A password sprayer written in C that supports spraying across the following services: ssh, http-get, and http-post.
- Allows

Service	Description
ssh (**s**ecure **sh**ell)	A protocol that allows you to gain access to the OS of a remote server (i.e. another device far away). Once you've successfully entered the login credentials, you get a shell where you can type in OS commands, and they'll be executed on the remote server! Moreover, the communication between you and the remote server is secure, meaning that it's very unlikely to be intercepted and read by third parties
http-get	This concerns HTTP/ HTTPS servers that use the basic auth scheme to authenticate users before sending the web page.
http-post	This concerns HTTP/HTTPS servers that send the user a login form, and expect the user to send login credentials as form variables, which will be stored on the server.

How Sprinkler works?

TL;DR

1. User input options/arguments through command line
2. Sprinkler processes the command line and checks for errors
3. Sprinkler prepares login request for the service specified
4. Sprinkler sends login request to server
5. If a login success is detected, stop spraying that username and move on to the next.

Longer version

1. **User input options/arguments through command line**

   The user input the following mandatory options/argument through the command line
   a. Username or username file (-u username or -U FILE)

b. Password or password file (-p password or -P FILE)
c. Target server's port (-s PORT)
d. Target server's IP address or domain name
e. Service

The user can also input extra options such as

- Set a delay time between logins (-d DELAY)
- Prints out status report and/or all login attempts (-v or -V)
- Connect via TLS (-S)
- Input a regex that determines a login success/failure based on whether the regex is found in target's response (-r REGEX)
- Specify form parameter names and values (if applicable) for servers that use form-based authentication (-i PARAMETERS)

2. **Sprinkler processes the command line**

In sprinkler.c file, Sprinkler parses the command line using C's <unistd.h> library function **getopt**. Essentially, a single call to **getopt** returns the nearest option character in the command line, starting with the first one. **getopt** also sets the variable *optarg* if there is an argument next to the option. Accordingly, Sprinkler has a while loop where each iteration gets the option, checks what option it is, whether an argument is supplied, and initialises appropriate variables.

While processing each option/argument, Sprinkler also checks its validity by using **getopt**'s feature: if the user types in an argument to an option that does not expect one, or there's no argument for an option that requires an argument, then **getopt** returns a recognisable marker. If multiple invalidities exist in the command line, Sprinkler detects as many as it can, prints the error messages for each invalidity, and exits.

The following invalidities are detected during this process:

- The five mandatory inputs (see above) are not provided
- Mismatch between actual vs. expected number of arguments
- Port is not a number or <= 0
- Username or password files do not exist
- Delay time is not a number or <= 0
- Input regex does not start with S= or F=
- Unknown option provided
- Service unsupported

3. **Sprinkler prepares login request for the service specified**

Each service expects wildly different format for the login request.

SSH

Sprinkler uses Libssh library functions to perform the following tasks:

- Set up a ssh_session object, which contains information about

HTTP-GET

HTTP-POST

NAME

sprinkler - a very fast password sprayer that works for ssh, http-get (basic auth), and http-post

SYNOPSIS

sprinkler [-u USERNAMES | -U FILE] [-p PASSWORD | -P FILE] [-s PORT] SERVICE TARGET

sprinkler [-h SERVICE | -h]

DESCRIPTION

sprinkler is a fast password sprayer that supports the following services:

ssh, http-get, http-post

This tool allows researchers and pen-testers to see how easy it would be to gain unauthorized access to a remote server.

MANDATORY OPTIONS

TARGET The server (and directory, if applicable) to attack, can be an IPv4 address or domain name

SERVICE The target's login service. Type 'sprinkler -h services' to list the protocols available

-s PORT
the target server's port
-u USERNAME / -U FILE
supply a login username, or -U FILE to supply a username list, one line per username
-p PASSWORD / -P FILE
supply a password, or -P FILE to supply a password list, one line per password

EXTRA GLOBAL OPTIONS
-d DELAY
set a delay time in seconds between each login attempt
-v / -V
Verbose mode. -V prints out all login attempts.