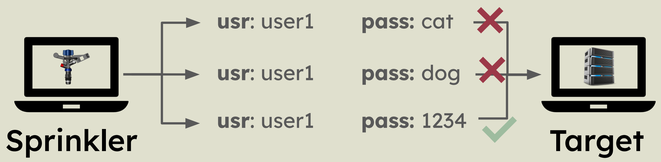
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References

What's password spraying?

- Guessing server login credentials as fast as possible, without getting detected



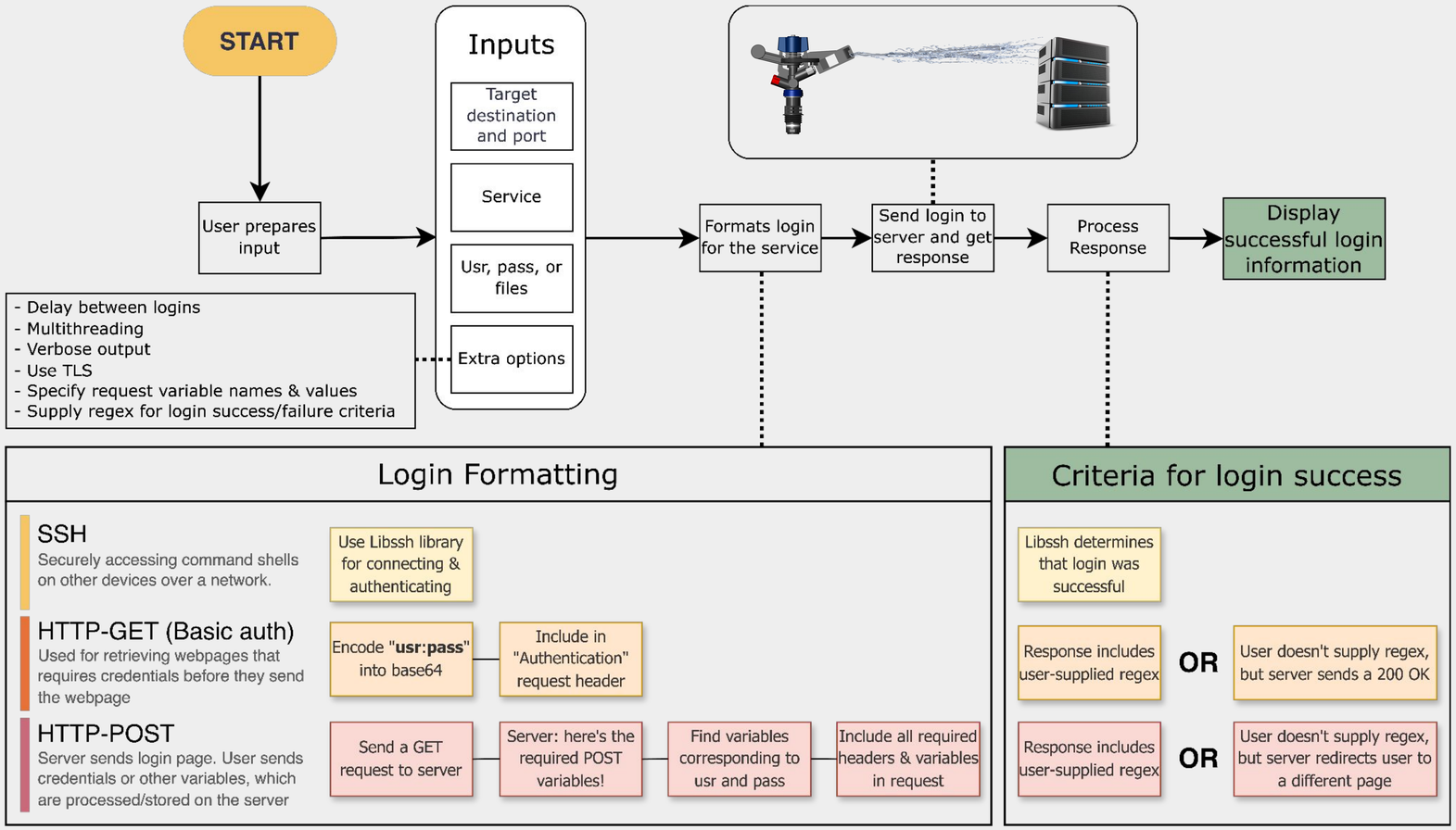
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

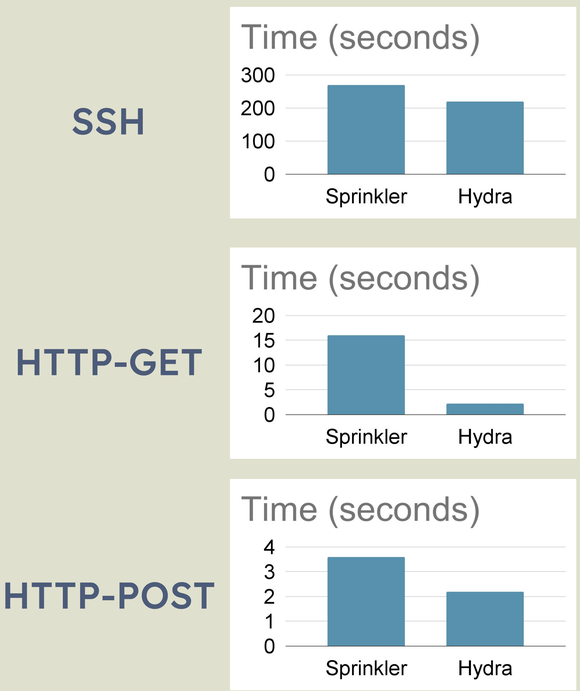
Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Prepare a test server compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

How Sprinkler works



Performance



Acknowledgements: Jeff Ondich, our classmates, and former Carleton CS graduates

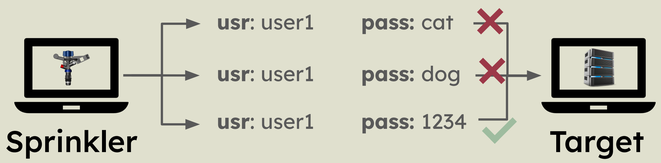
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



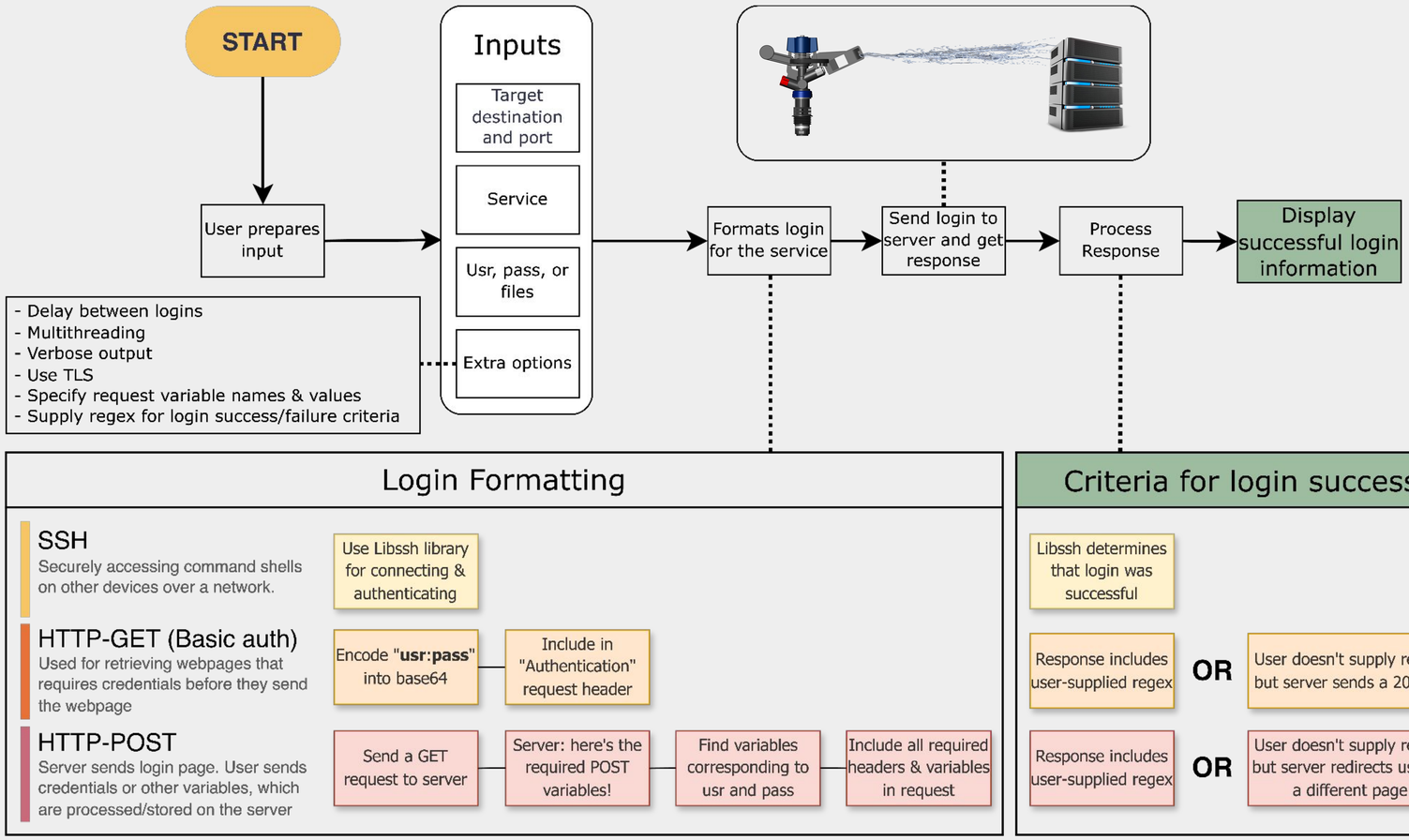
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

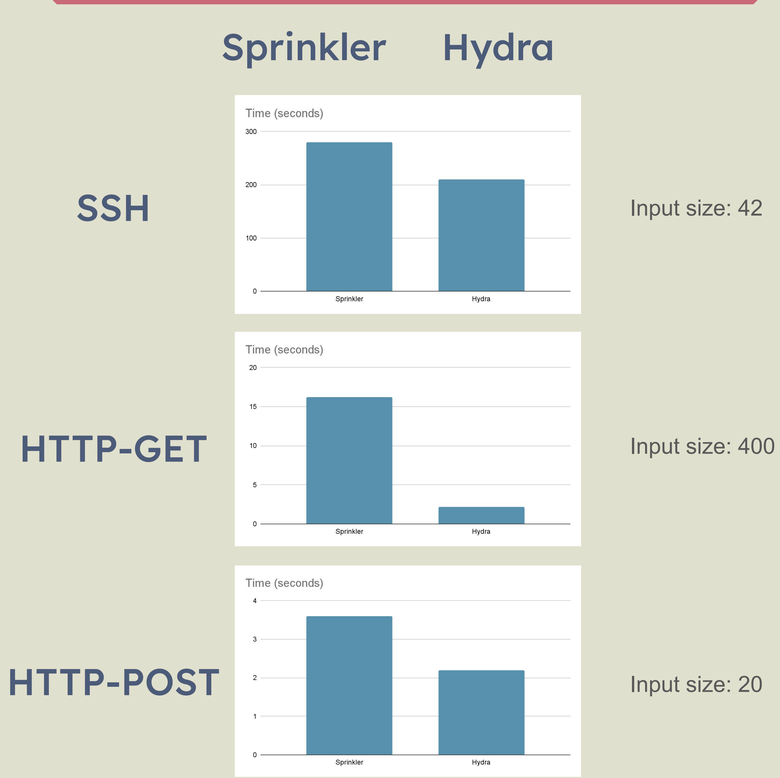
Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Prepare a test server compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

How Sprinkler works



Performance



Acknowledgements: Jeff Ondich, our classmates, and former Carleton CS graduates

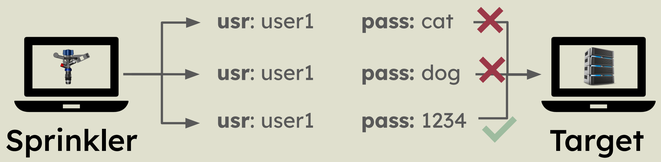
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



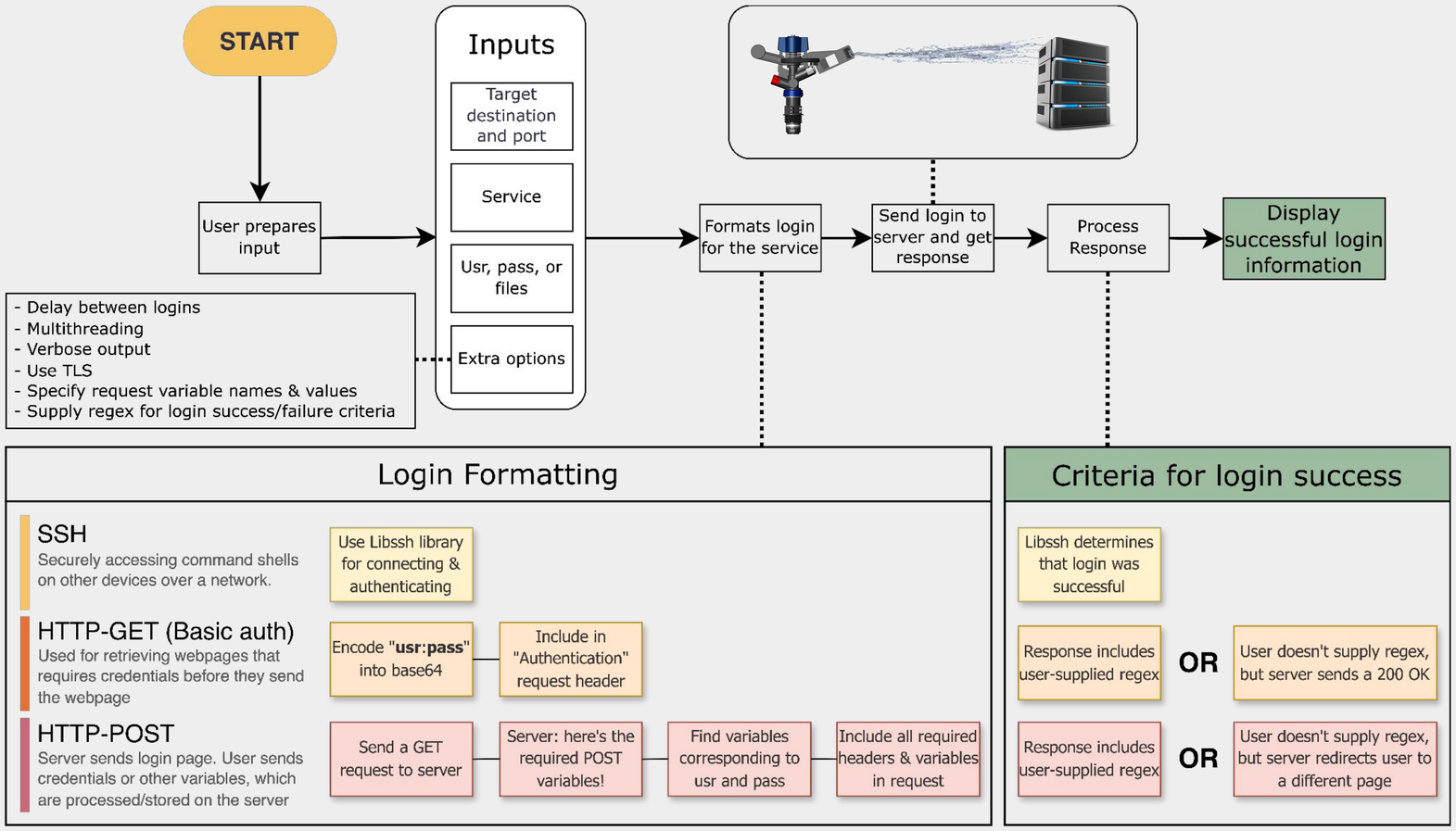
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

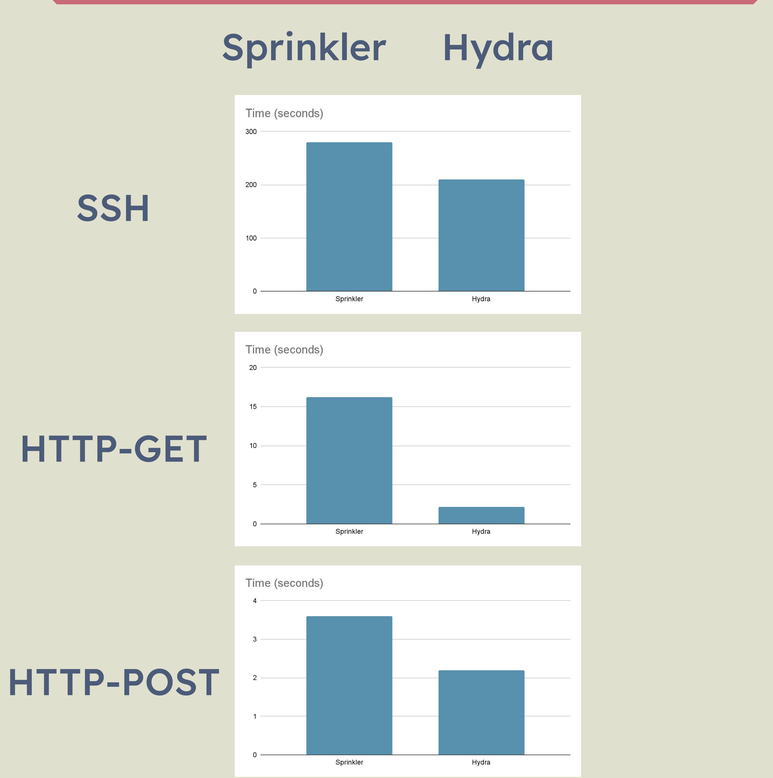
Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Prepare a test server compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

How Sprinkler works



Performance



Acknowledgements: Jeff Ondich, our classmates, and former Carleton CS graduates

Sprinkler: A Multi-Service Password Sprayer

Sam E, Prompt E, advised by Jeff Ondich

Carleton

What is password spraying

- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How spraying works for each service

HTTP-Get (basic auth):

1. Join usr and pass into "usr:pass" and encode into base64
2. Format a GET request with encoded credentials
3. Send request to server
4. Login success = server responds with 200 OK

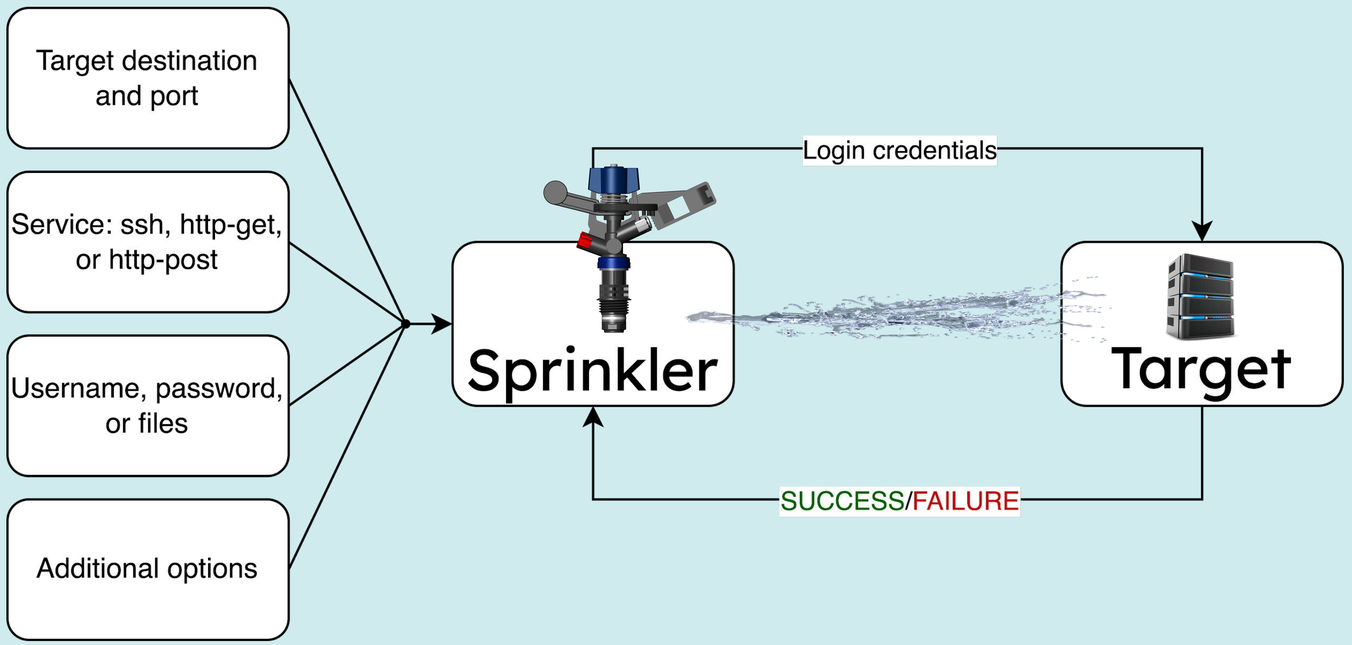
HTTP-Post:

1. Send a GET request
2. Server sends all the variable names required for a POST request
3. Find out which variable names correspond to the username and password
4. Format a POST request with all the required variables, and send to server
5. Login success = user-supplied regex is in server's response OR server redirects to a different page

SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Carleton

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

What's password spraying?

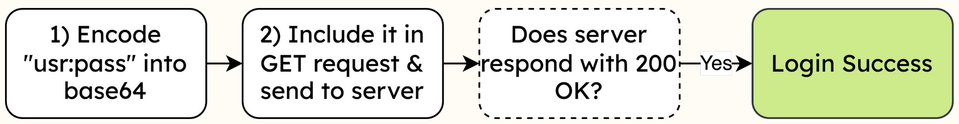
- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

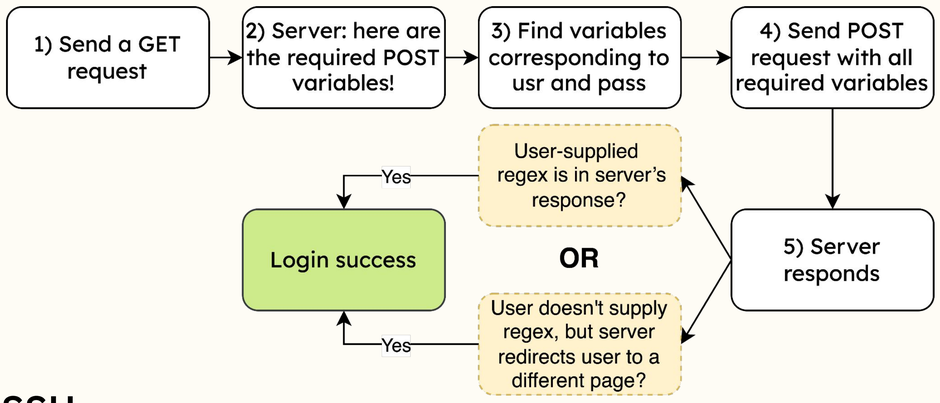
- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How sprinkler works for each service?

HTTP-Get (basic auth)



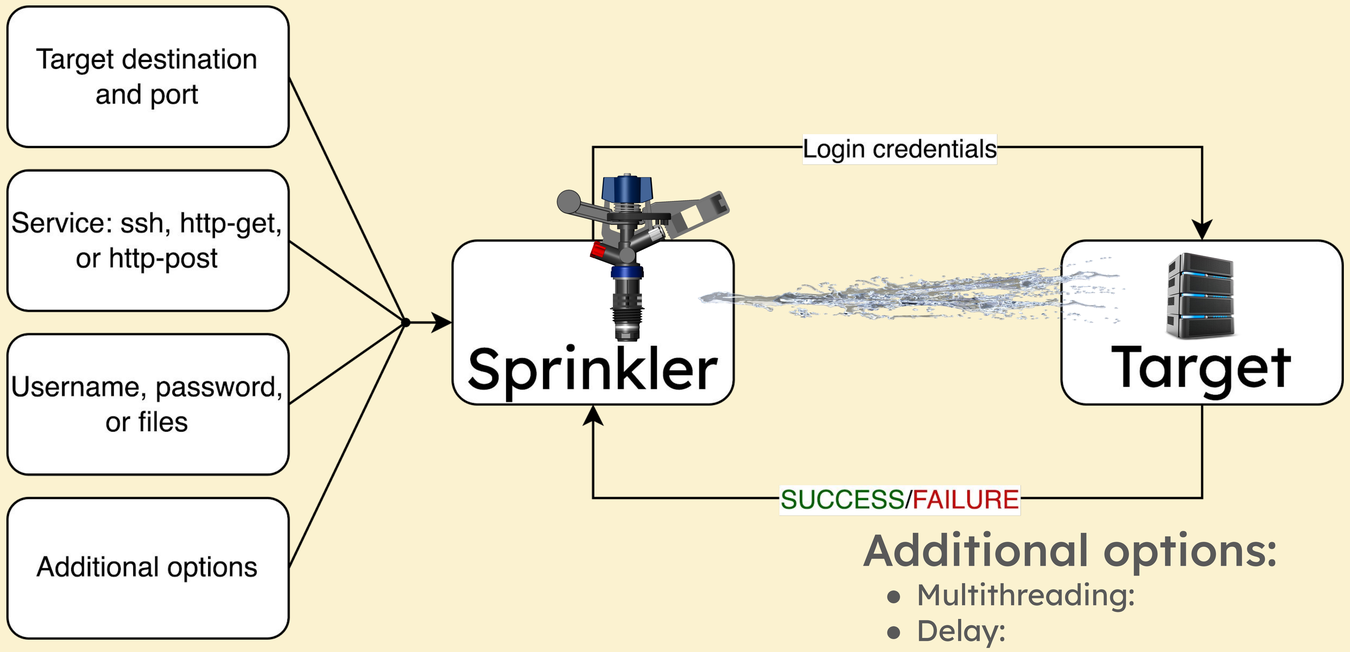
HTTP-Post



SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

What's password spraying?

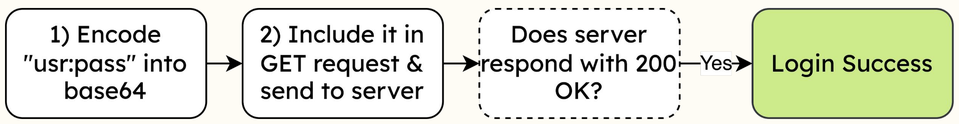
- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

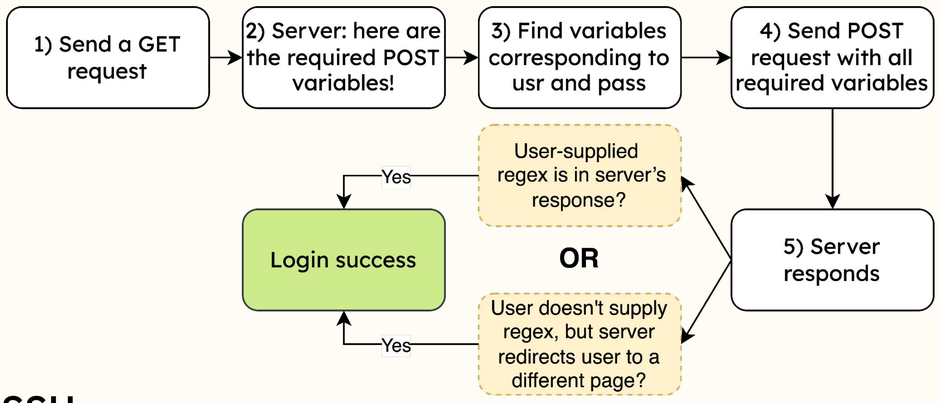
- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How sprinkler works for each service?

HTTP-Get (basic auth)



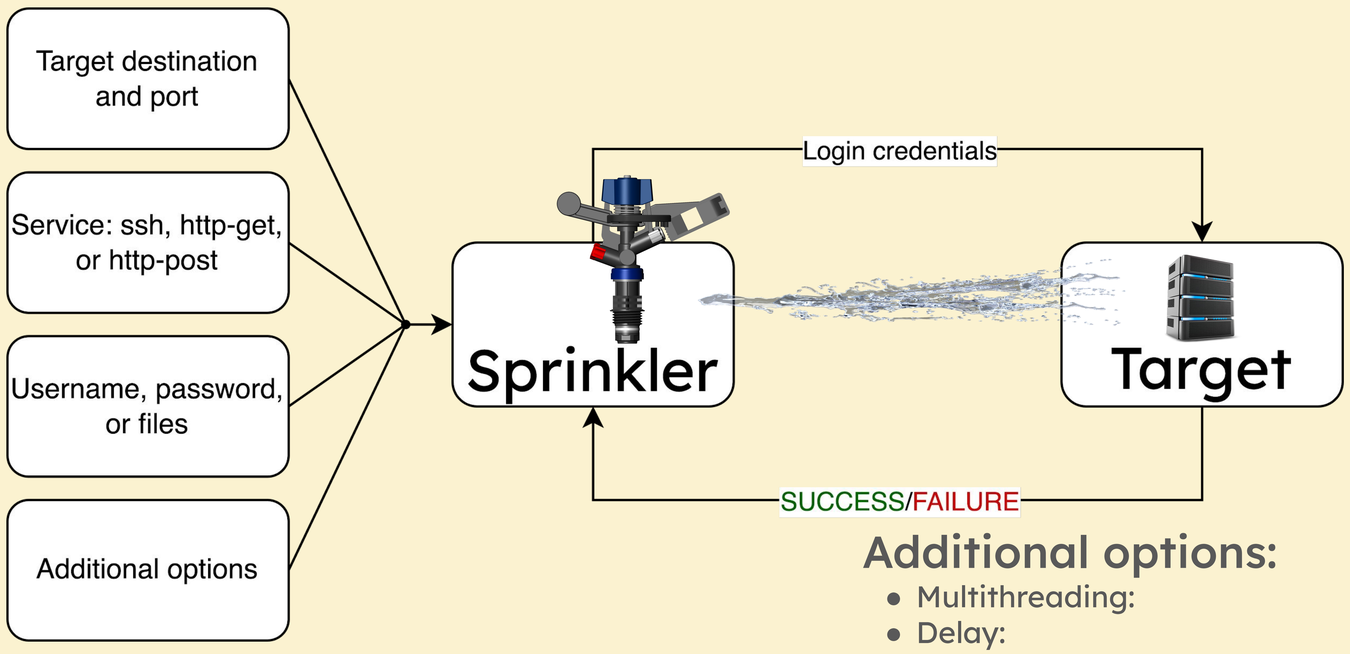
HTTP-Post



SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

What's password spraying?

- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

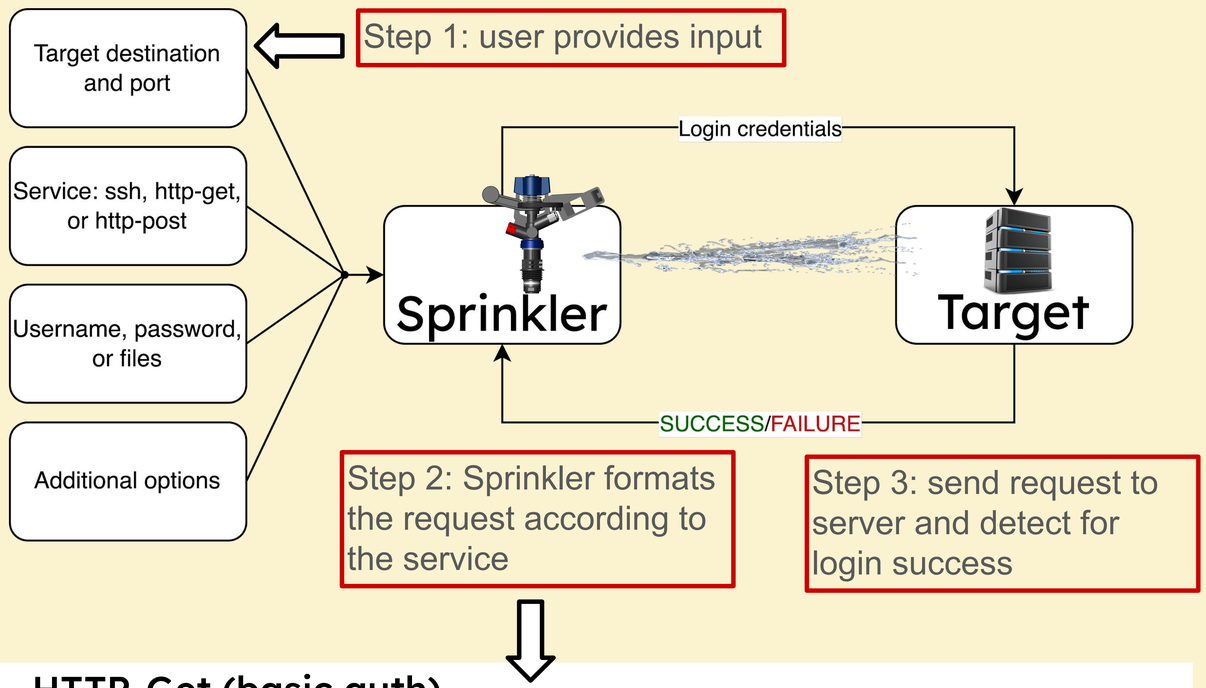
Our goal

1. Make a fast password sprayer that works for ssh, http-get, http-post
2. Make a test server and machine compatible with the services we need
3. See how we can configure our test server to protect from password spraying attacks
4. Compare our password sprayer's performance with Hydra

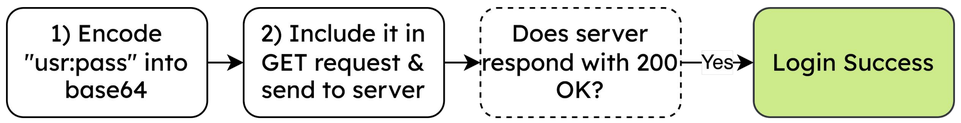
The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

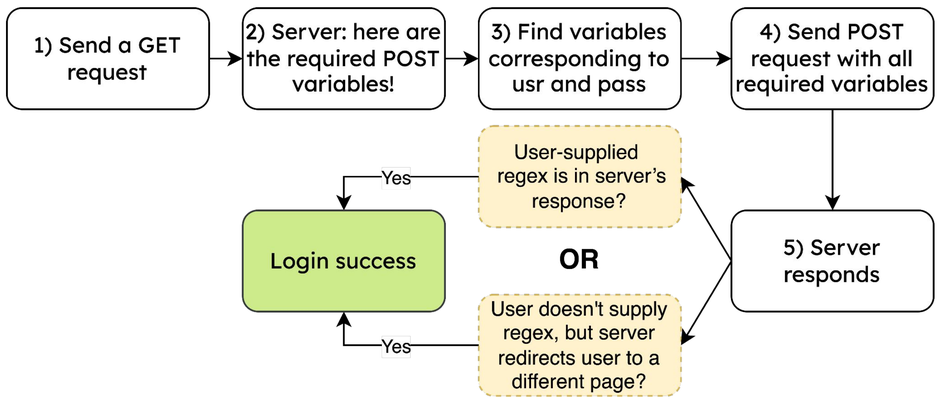
Visualization of Password Spraying



HTTP-Get (basic auth)



HTTP-Post



SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Performance testing

- Blah

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

Why is it important?

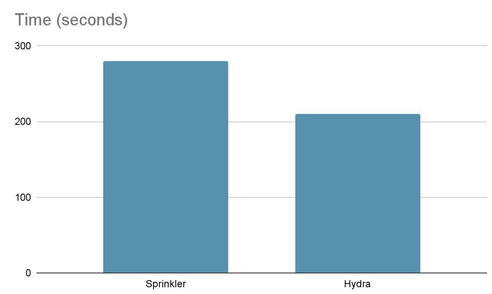
What's password spraying?

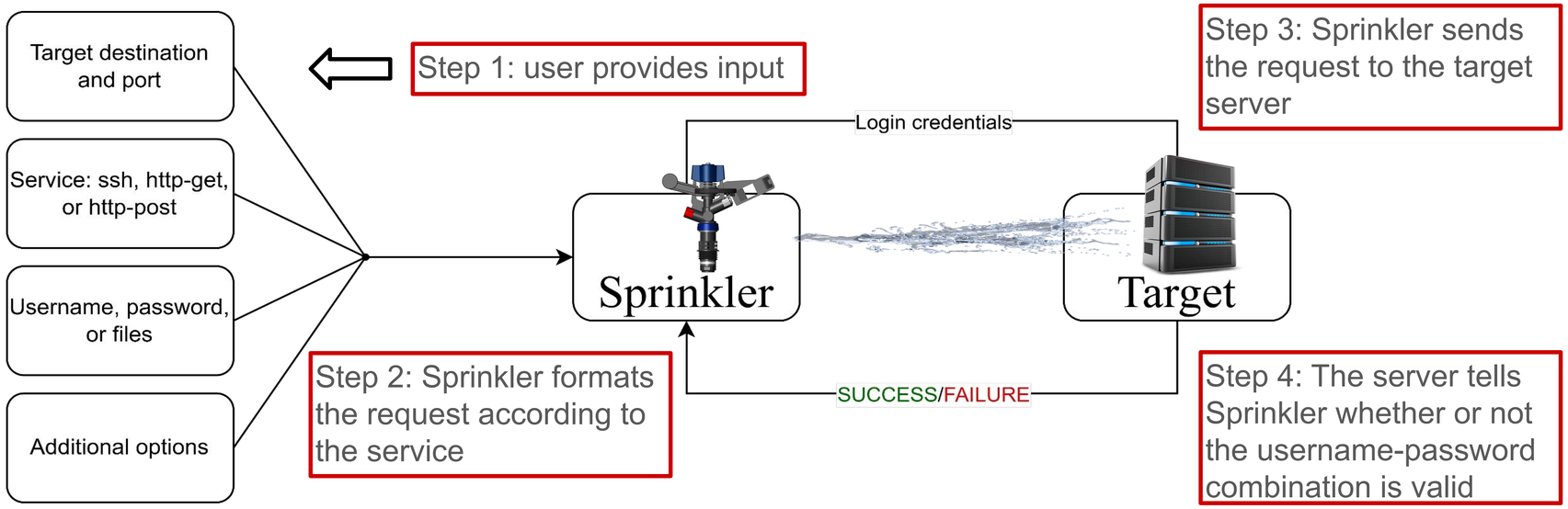
- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

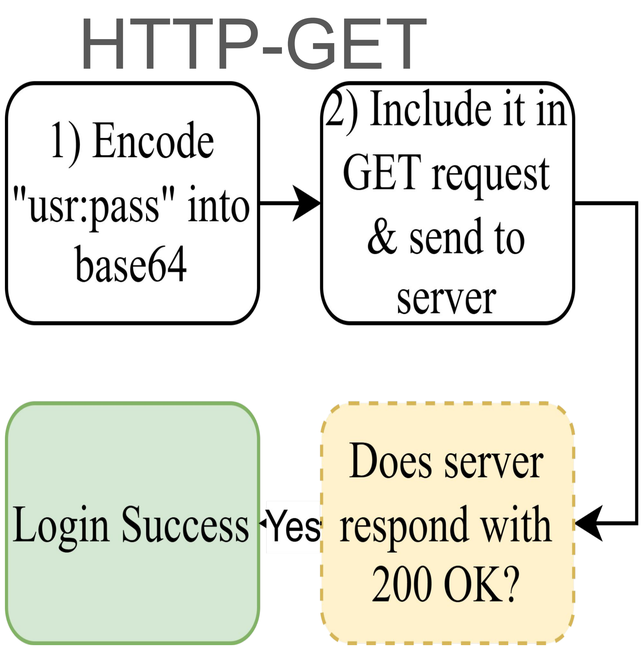


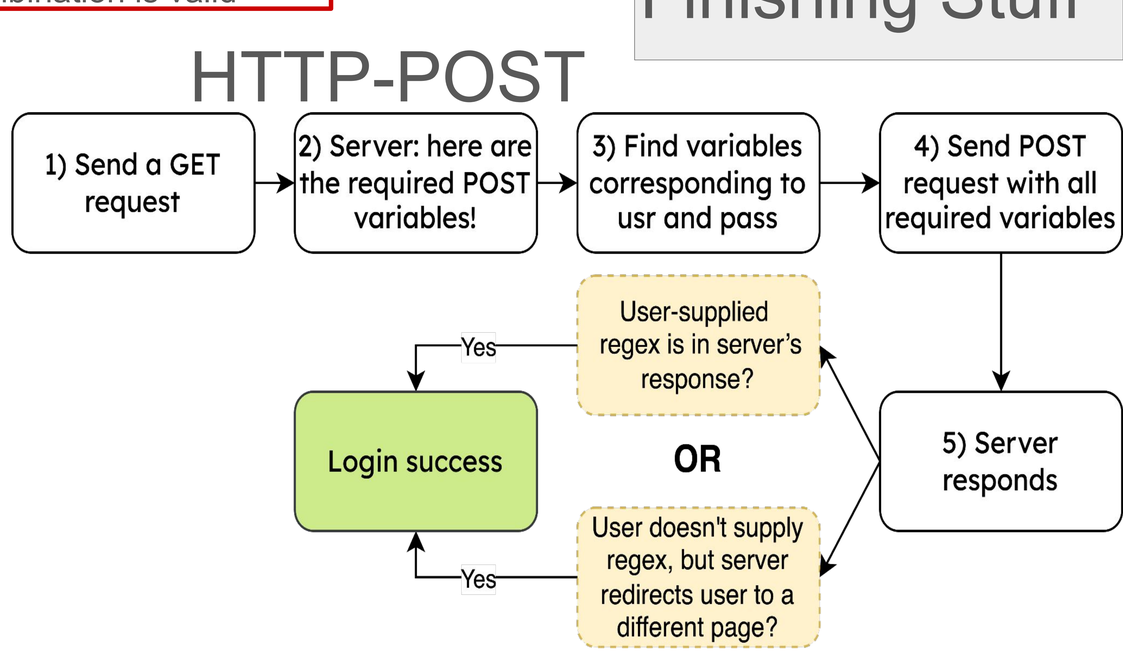


Finishing Stuff

Finishing Stuff

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal





Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

What's password spraying?

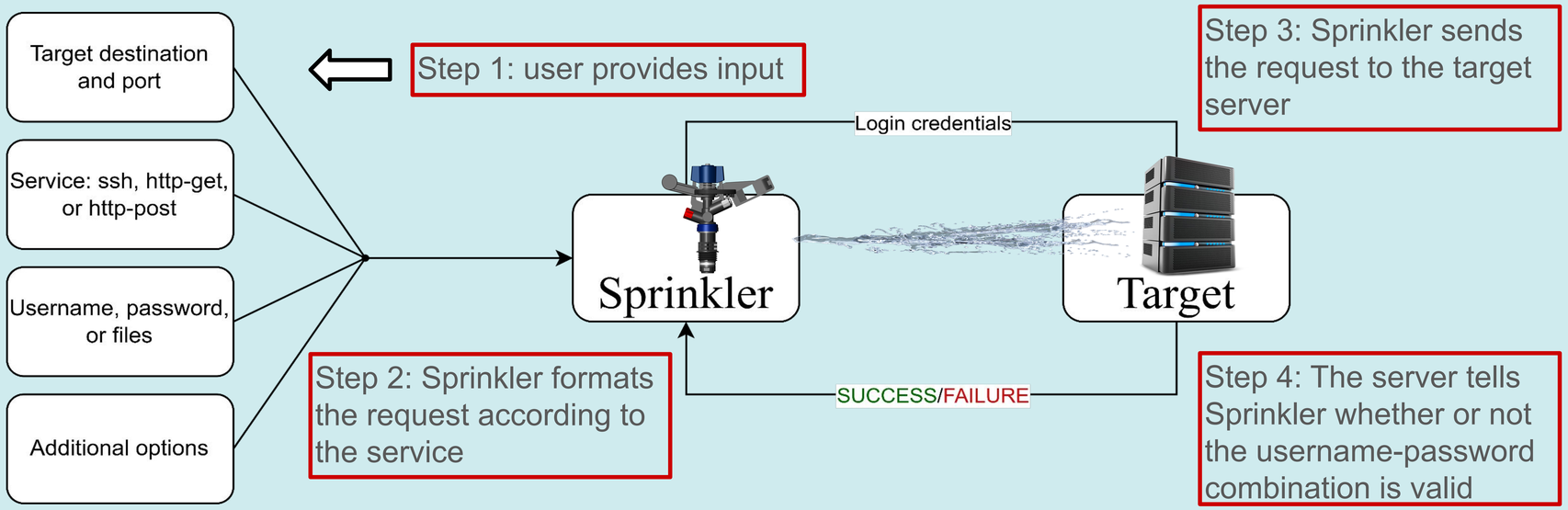
- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our Goal

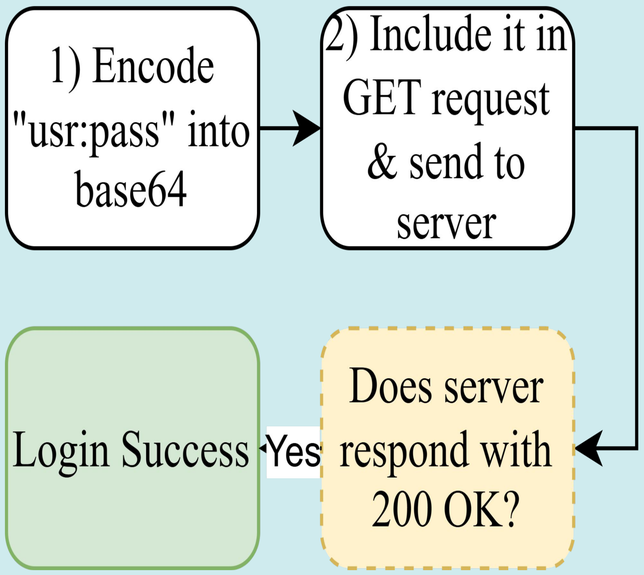
- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need



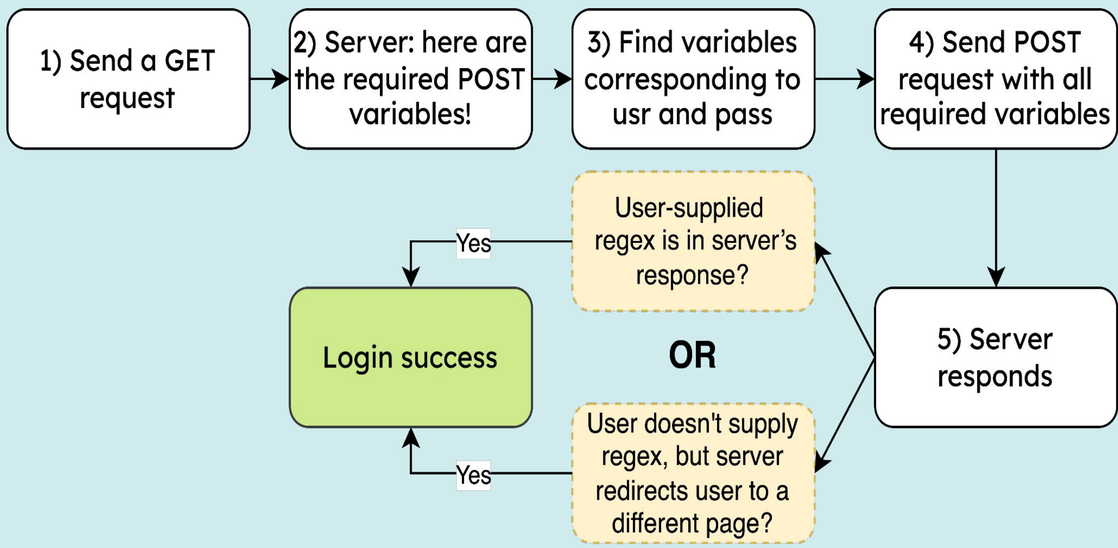
SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

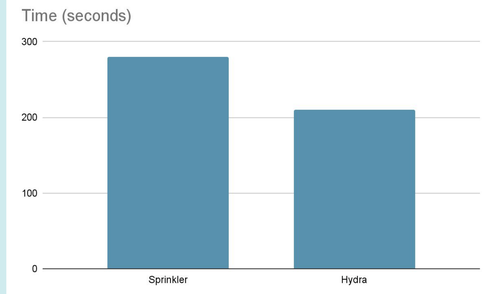
HTTP-GET

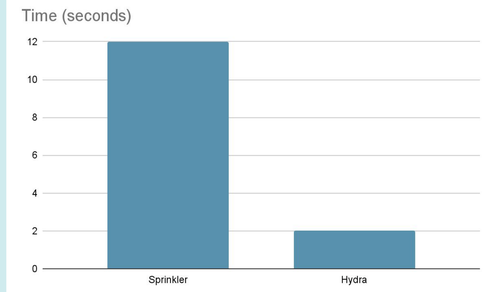


HTTP-POST



Hydra Comparisons





And lots of missing formats, including but not limited to: RSA, CISCO, more HTTP stuff, sshkey, and several others

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

QR

What's password spraying?

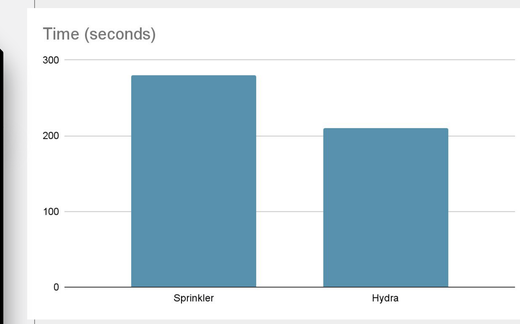
- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

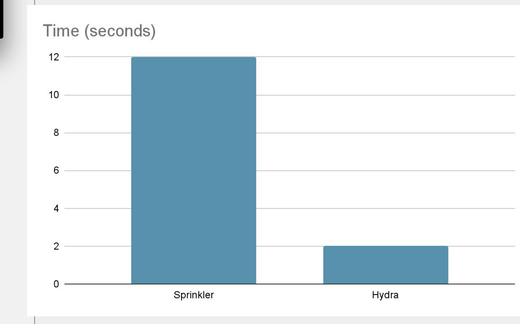
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

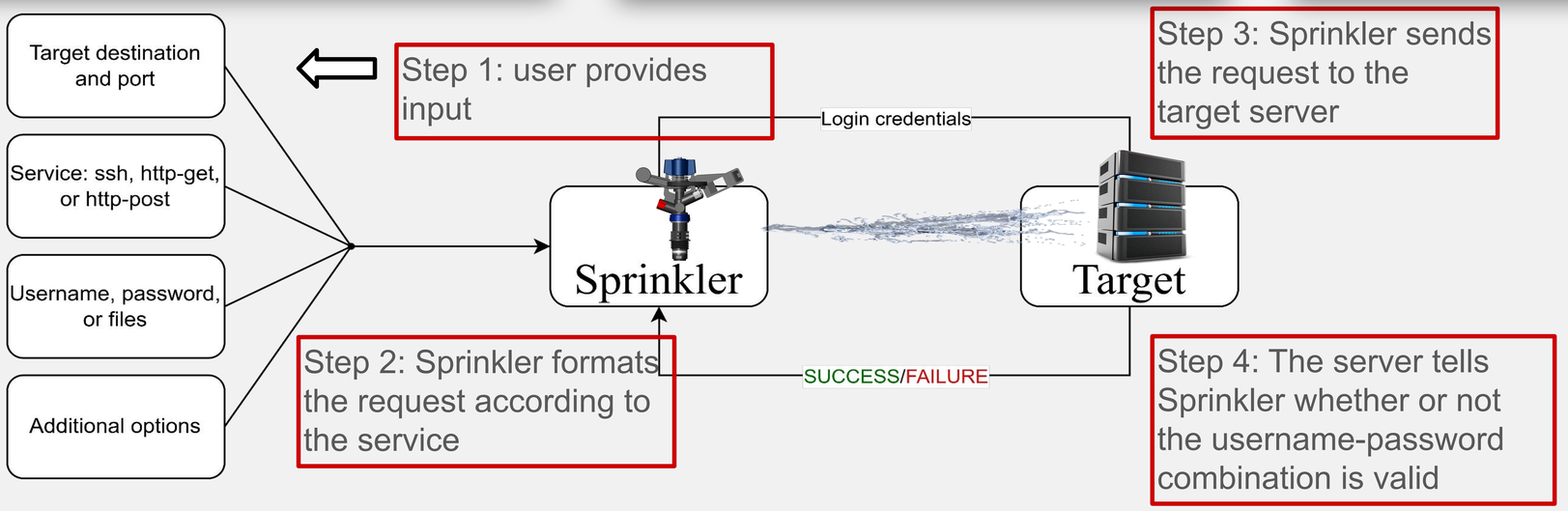
Our goal

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.





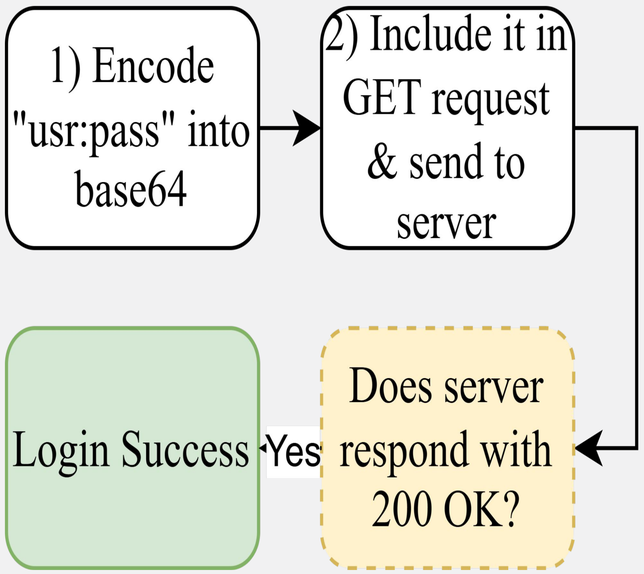
Finishing Stuff



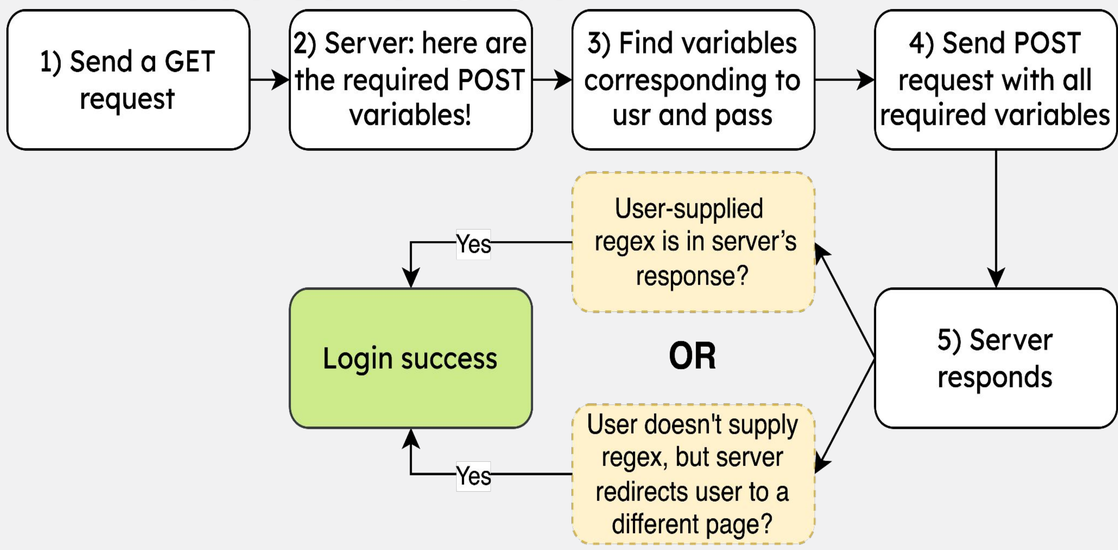
SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

HTTP-GET



HTTP-POST



Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

ACK

What's password spraying?

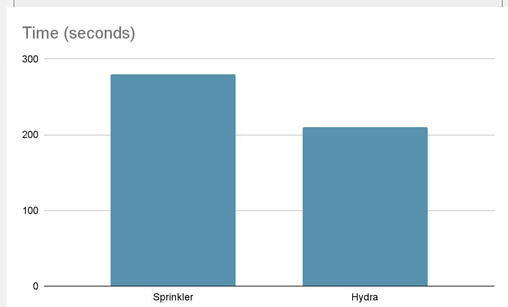
- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

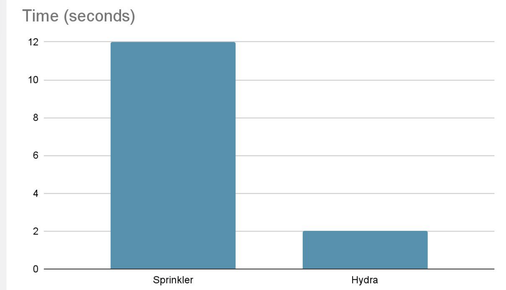
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need





Step 1: user provides input through command-line

Target destination and port

Service

Usr, pass, or files

General or service options

Prepare login for the service

Send login to server and get response

Process Response

SSH

Use Libssh library to connect to server

HTTP-GET: Basic auth

Encode "**usr**:**pass**" into base64

Include in "Authentication" header

HTTP-POST

Set a GET request to server

Server: here's the required POST variables!

Find variables corresponding to usr and pass

Include all required headers & variables

Libssh determines that login was successful?

Server sends a 200 OK?

Response includes user-supplied regex?

OR

User doesn't supply regex, but server redirects to a different page?

LOGIN SUCCESS

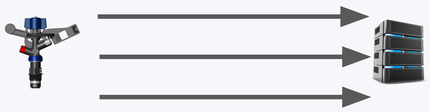
Finishing Stuff

Sprinkler: a Multi-Service Password Sprayer ACK

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

What's password spraying?

- Trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.

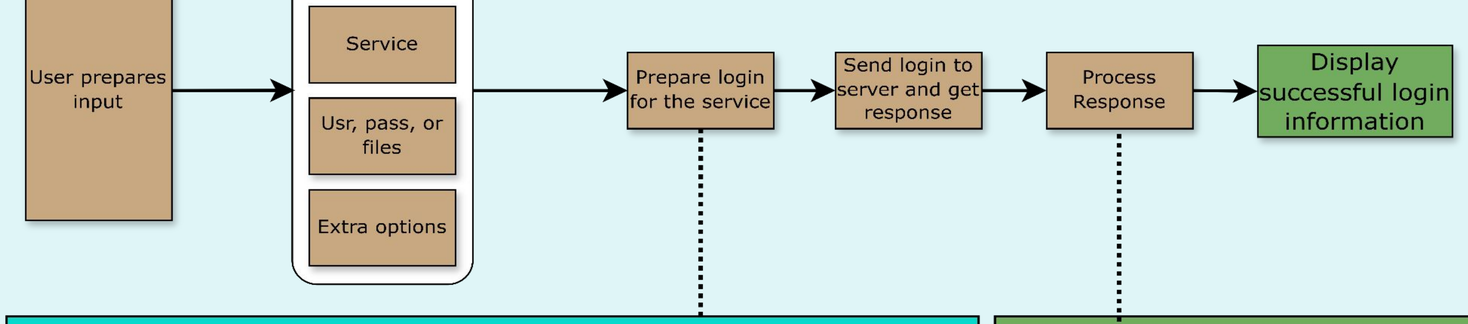


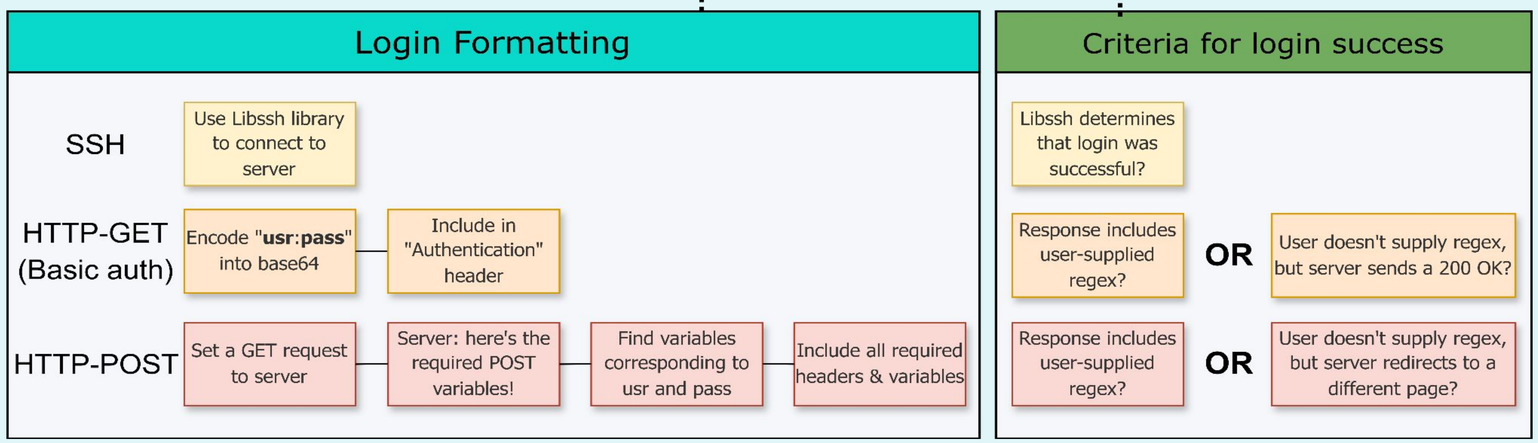
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need





Extra Options:

- Verbose mode
- Delay between attempts
- Multithreading mode

SSH

- SSH is a service typically used to remotely login to command shells on other devices over a network.

HTTP-GET

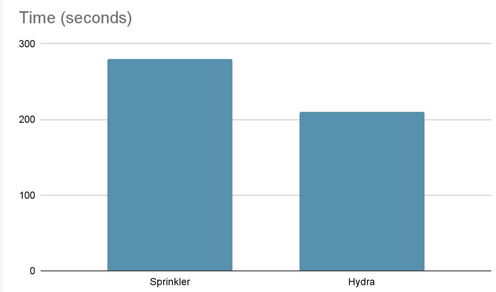
- HTTP-Get logins are used when websites require login credentials before they send the webpage.

HTTP-POST

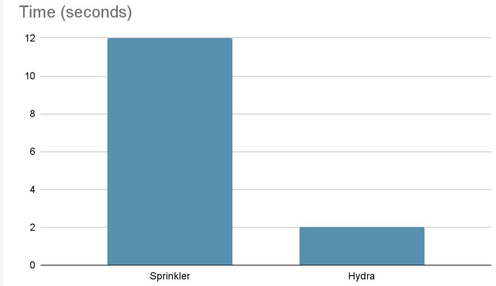
- HTTP-POST logins are often used when websites require login credentials when sending other information to the webpage.

Hydra Comparisons

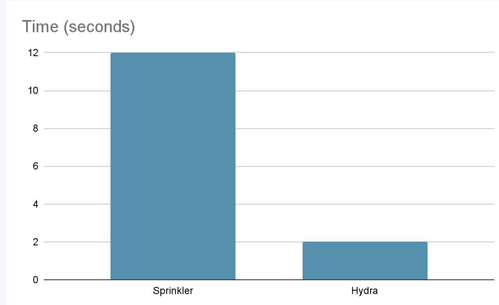
SSH



HTTP-GET



HTTP-POST



And lots of missing formats, including but not limited to: RSA, CISCO, more HTTP stuff, sshkey, and several others

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

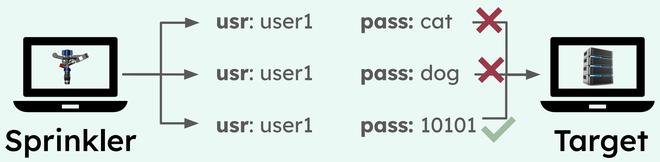
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References & methodology

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



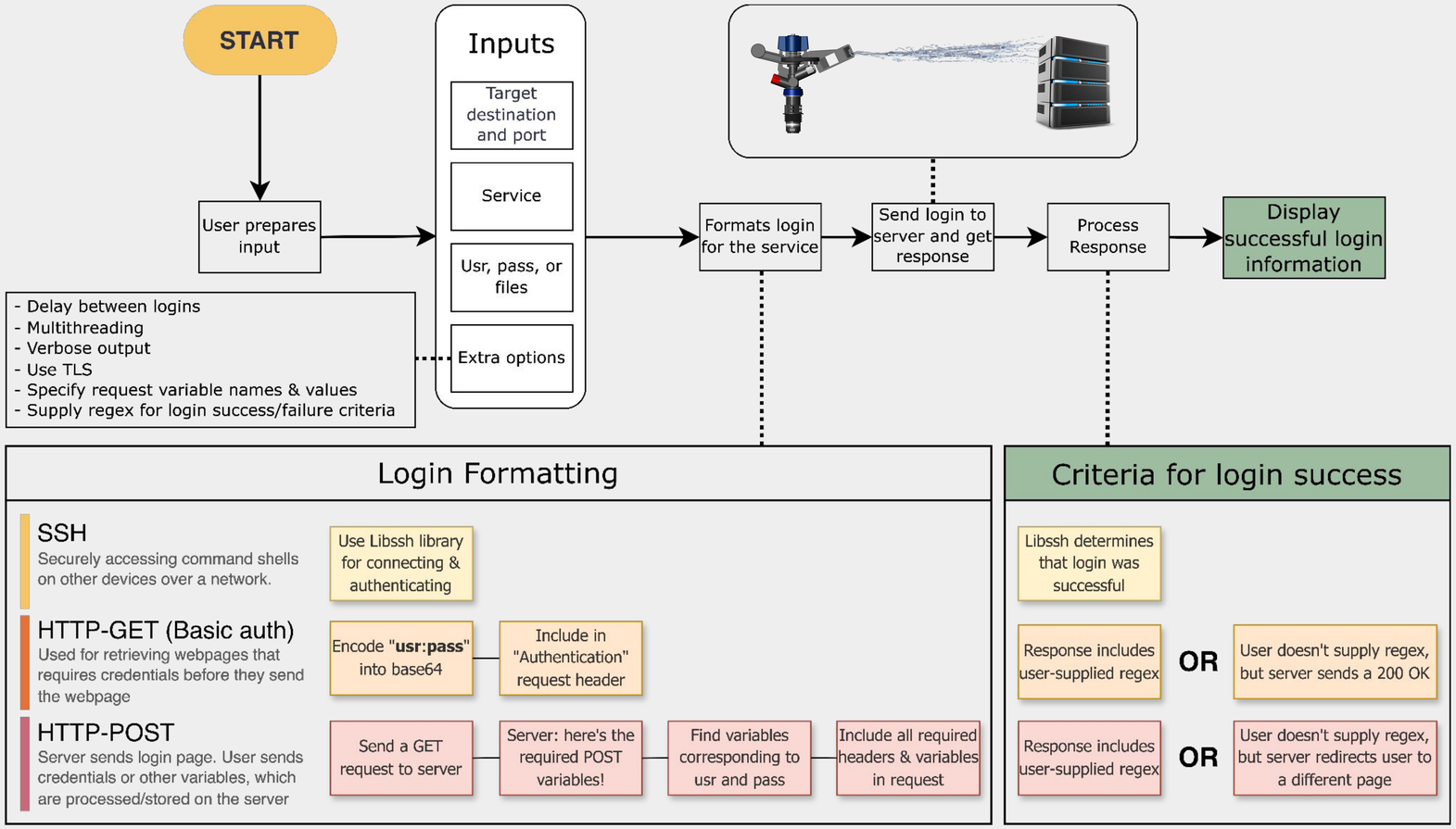
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

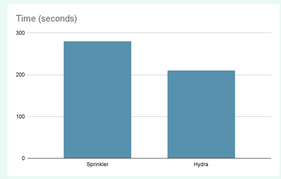
- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

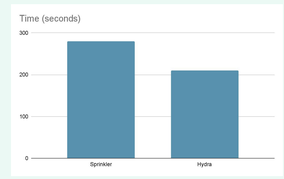
How Sprinkler works



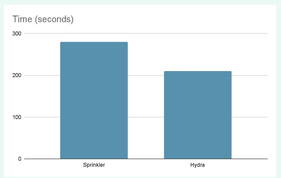
Performance

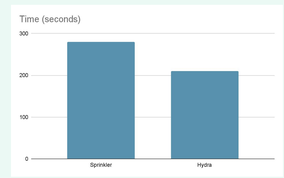
SSH



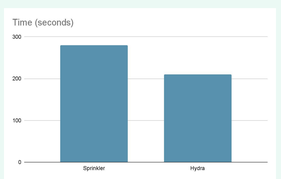


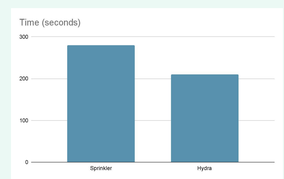
HTTP-GET





HTTP-POST





Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, libssh, openSSL, and flaticon

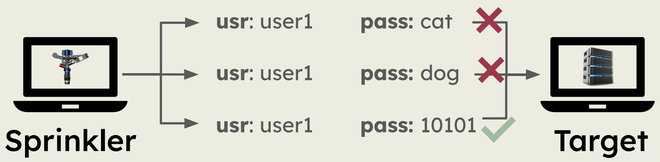
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References & methodology

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



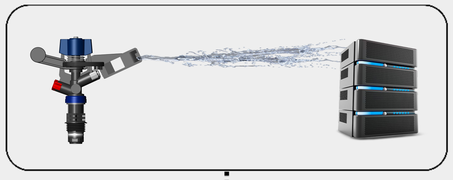
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

How Sprinkler works



START

User prepares input

Inputs
- Target destination and port
- Service
- Usr, pass, or files
- Extra options

- Delay between logins
- Multithreading
- Verbose output
- Use TLS
- Specify request variable names & values
- Supply regex for login success/failure criteria

Formats login for the service

Send login to server and get response

Process Response

Display successful login information

Login Formatting

SSH
Securely accessing command shells on other devices over a network.
- Use Libssh library for connecting & authenticating

HTTP-GET (Basic auth)
Used for retrieving webpages that requires credentials before they send the webpage
- Encode "**usr:pass**" into base64
- Include in "Authentication" request header

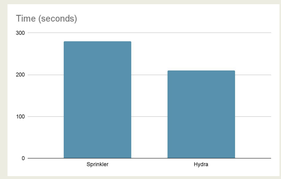
HTTP-POST
Server sends login page. User sends credentials or other variables, which are processed/stored on the server
- Send a GET request to server
- Server: here's the required POST variables!
- Find variables corresponding to usr and pass
- Include all required headers & variables in request

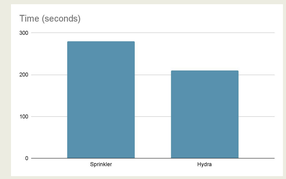
Criteria for login success

- Libssh determines that login was successful
- Response includes user-supplied regex **OR** User doesn't supply regex, but server sends a 200 OK
- Response includes user-supplied regex **OR** User doesn't supply regex, but server redirects user to a different page

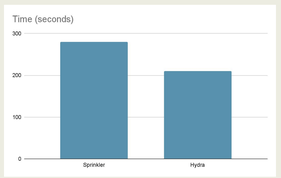
Performance

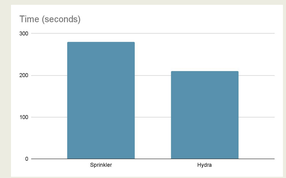
SSH



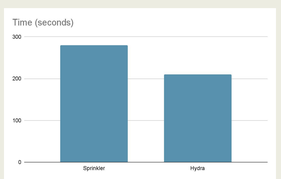


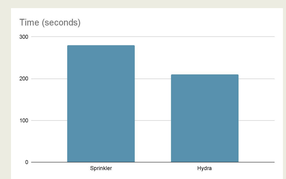
HTTP-GET





HTTP-POST





Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, libssh, openSSL, and flaticon

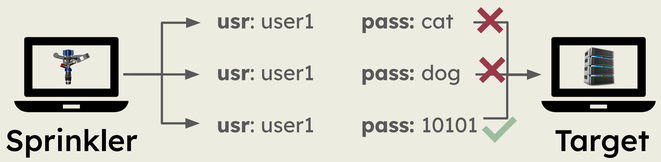
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References & methodology

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



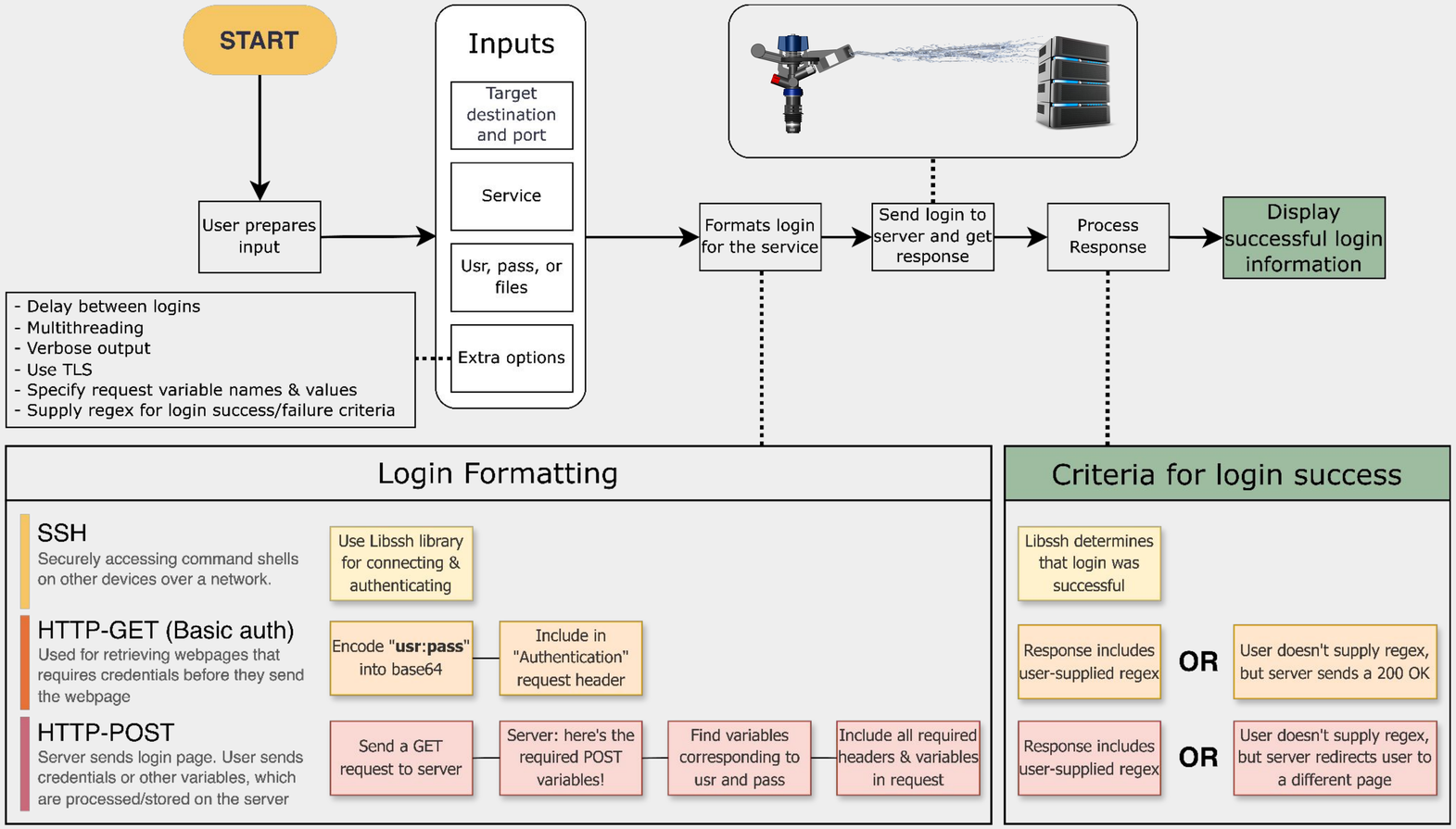
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

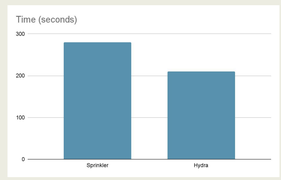
- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

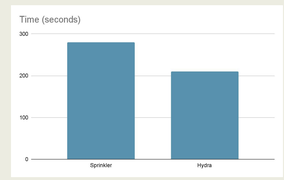
How Sprinkler works



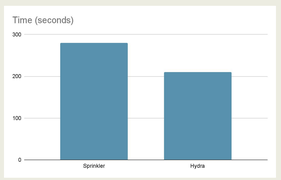
Performance

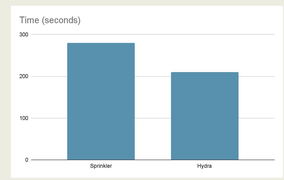
SSH



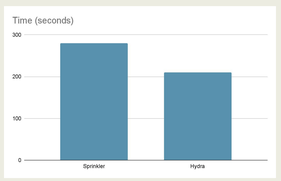


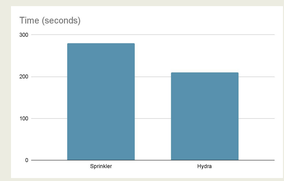
HTTP-GET





HTTP-POST





Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, libssh, openSSL, and flaticon

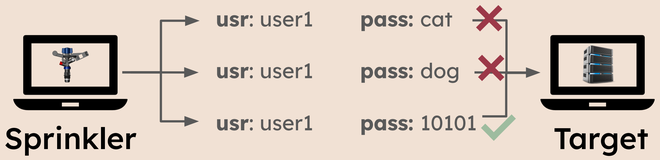
Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

References & methodology

What's password spraying?

- Guessing login credentials as fast as possible, without getting detected



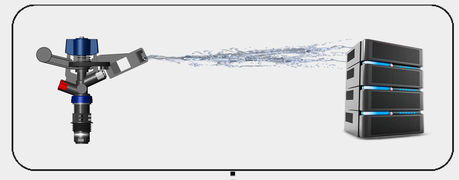
Why it's important?

- It's used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.

Our goal

- Make a fast password sprayer that works for ssh, http-get (basic auth), http-post. Language = C
- Make a test server and machine compatible with the services we need
- Compare Sprinkler's performance with Hydra, a popular password sprayer

How Sprinkler works



START

User prepares input

Inputs
- Target destination and port
- Service
- Usr, pass, or files
- Extra options

- Delay between logins
- Multithreading
- Verbose output
- Use TLS
- Specify request variable names & values
- Supply regex for login success/failure criteria

Formats login for the service

Send login to server and get response

Process Response

Display successful login information

Login Formatting

SSH
Securely accessing command shells on other devices over a network.
- Use Libssh library for connecting & authenticating

HTTP-GET (Basic auth)
Used for retrieving webpages that requires credentials before they send the webpage
- Encode "**usr:pass**" into base64
- Include in "Authentication" request header

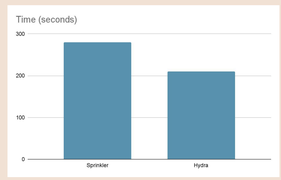
HTTP-POST
Server sends login page. User sends credentials or other variables, which are processed/stored on the server
- Send a GET request to server
- Server: here's the required POST variables!
- Find variables corresponding to usr and pass
- Include all required headers & variables in request

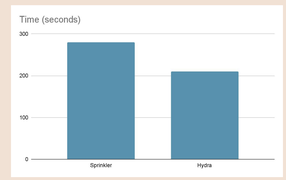
Criteria for login success

- Libssh determines that login was successful
- Response includes user-supplied regex **OR** User doesn't supply regex, but server sends a 200 OK
- Response includes user-supplied regex **OR** User doesn't supply regex, but server redirects user to a different page

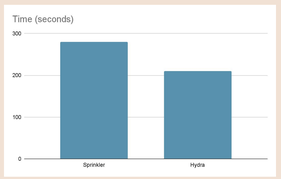
Performance

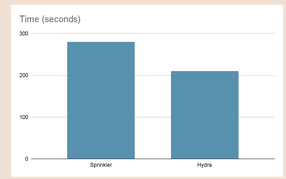
SSH



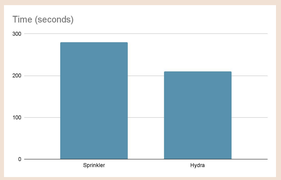


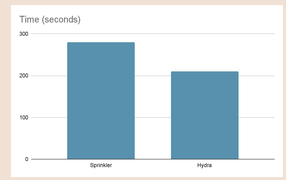
HTTP-GET





HTTP-POST





Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, libssh, openSSL, and flaticon

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

Carleton

QR

What's password spraying?

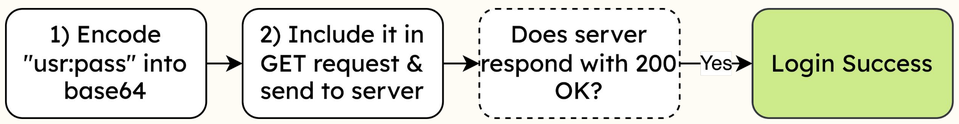
- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

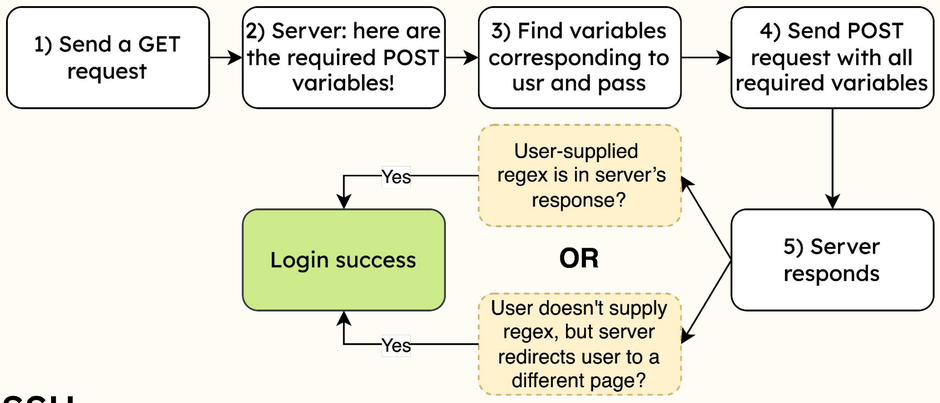
- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How sprinkler works for each service?

HTTP-Get (basic auth)



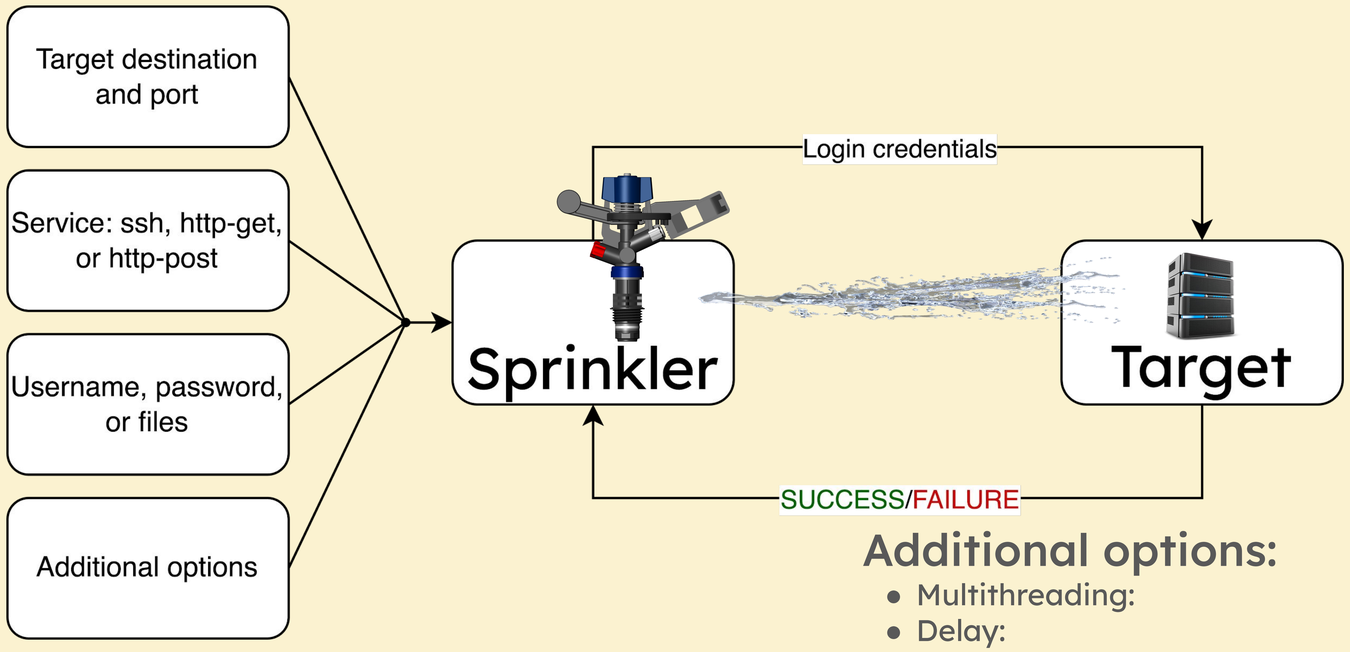
HTTP-Post



SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer
Sam E, Prompt E, advised by Jeff Ondich

Carleton

What's password spraying?

- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How spraying works for each service

HTTP-Get (basic auth):

1. Join usr and pass into "usr:pass" and encode into base64
2. Format a GET request with encoded credentials
3. Send request to server
4. Login success = server responds with 200 OK

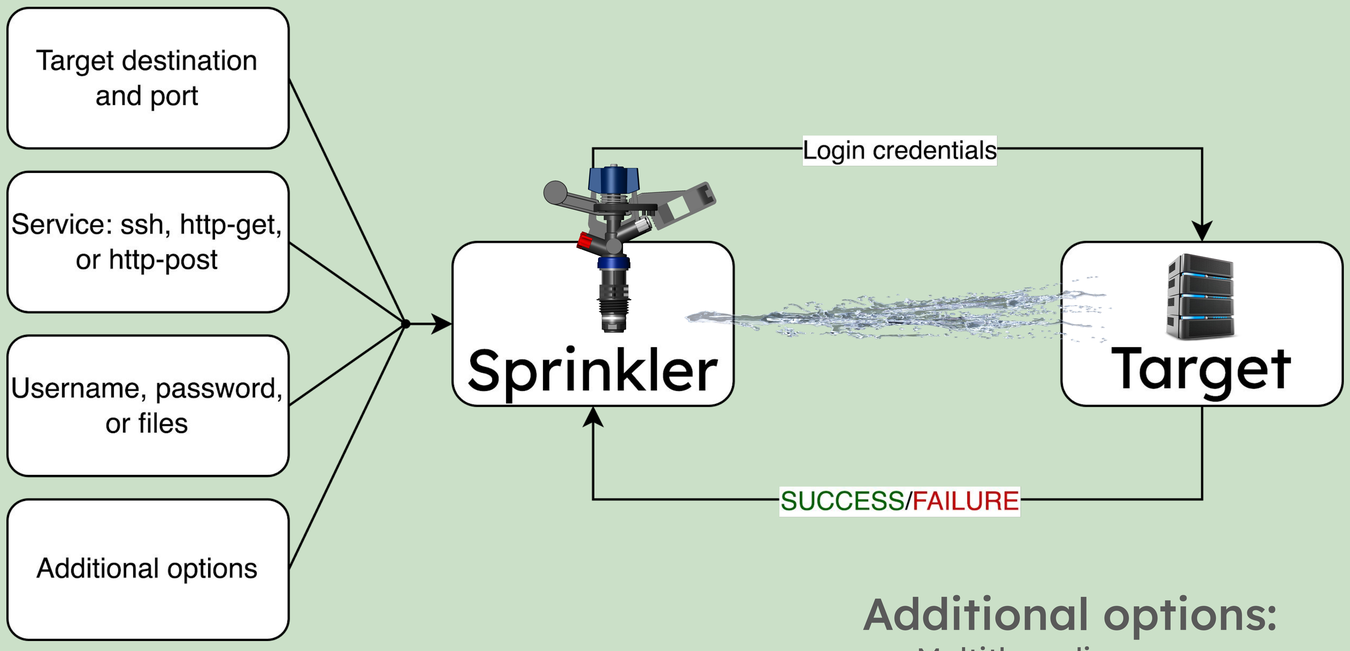
HTTP-Post:

1. Send a GET request
2. Server sends all the variable names required for a POST request
3. Find out which variable names correspond to the username and password
4. Format a POST request with all the required variables, and send to server
5. Login success = user-supplied regex is in server's response OR server redirects to a different page

SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously

- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam Ederington, Prompt Eua-anant, advised by Jeff Ondich

What's password spraying?

- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How spraying works for each service

HTTP-Get (basic auth):

1. Join usr and pass into "usr:pass" and encode into base64
2. Format a GET request with encoded credentials
3. Send request to server
4. Login success = server responds with 200 OK

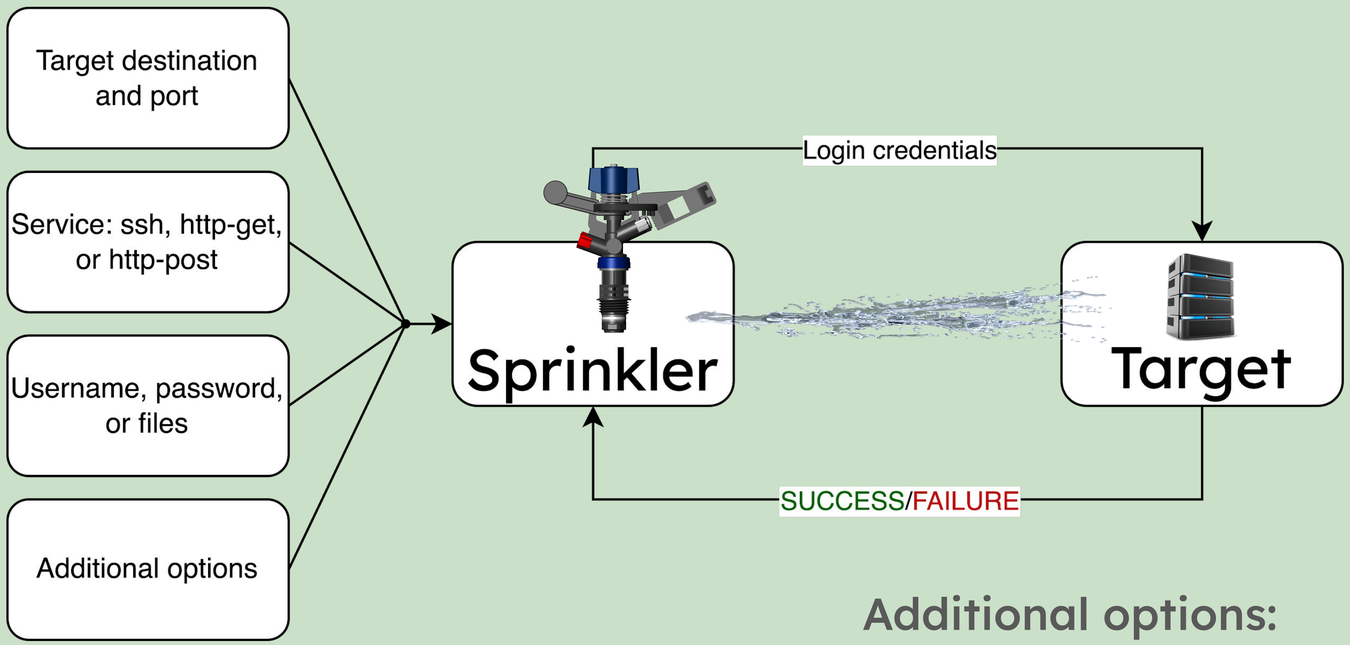
HTTP-Post:

1. Send a GET request
2. Server sends all the variable names required for a POST request
3. Find out which variable names correspond to the username and password
4. Format a POST request with all the required variables, and send to server
5. Login success = user-supplied regex is in server's response OR server redirects to a different page

SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh

Sprinkler: A Multi-Service Password Sprayer

Sam E, Prompt E, advised by Jeff Ondich

Carleton

What is password spraying

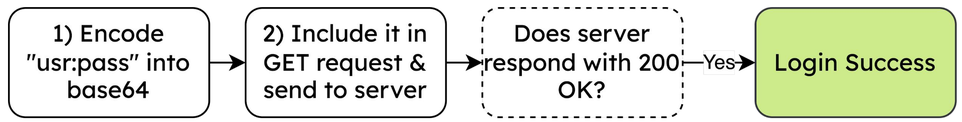
- Password spraying is trying to login to a target server via multiple user accounts with a list of passwords as quickly as possible.
- Password spraying is used for hacking and security purposes, finding accounts with vulnerable login permissions to attack or require changing.
- Password spraying simulates a user's login request to whatever type of service they would use (website, remote desktop, …), sending a username and password and checking for a successful login before terminating the connection.

Our goal

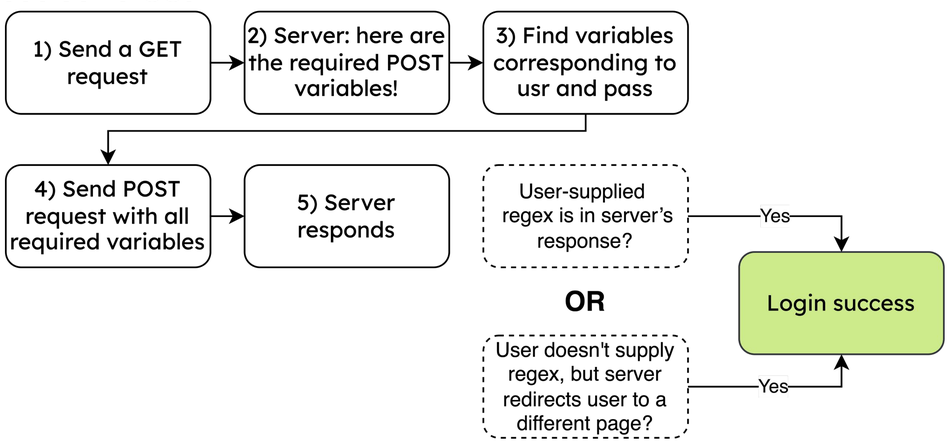
- Make a fast password sprayer that works for ssh, http-get, http-post
- Make a test server and machine compatible with the services we need

How sprinkler works for each service

HTTP-Get (basic auth)



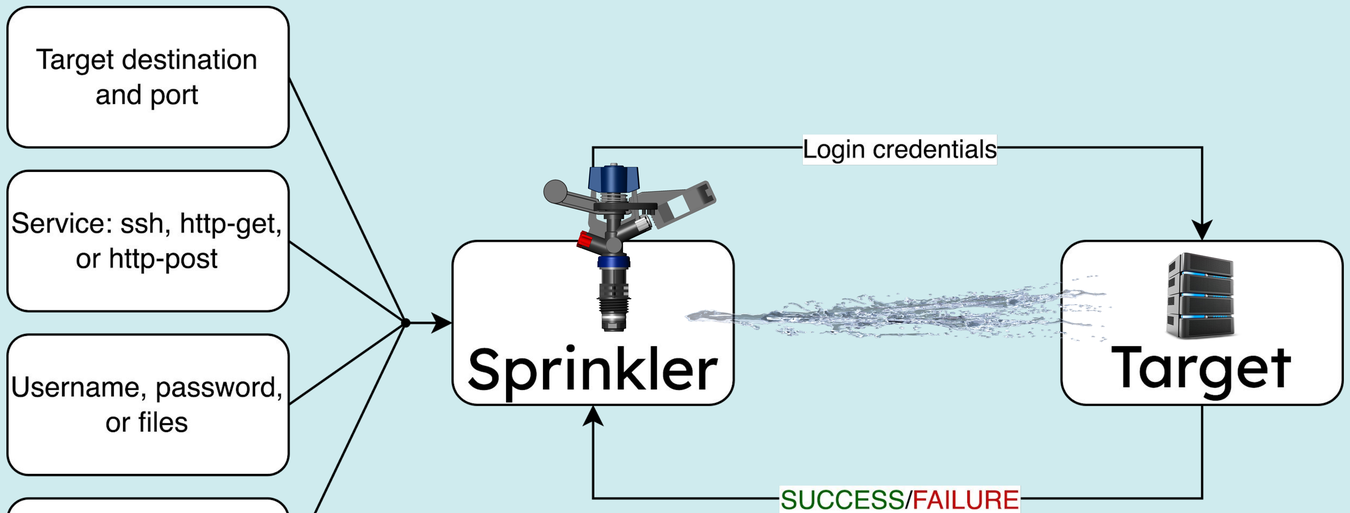
HTTP-Post



SSH

1. We used libssh library functions to connect to the server
2. Format and send the login request to the server
3. Login success = server responds with authentication successful and gives access to its terminal

Visualization of Password Spraying



Additional options:

- Multithreading:
- Delay:
- Regex for success condition:

The Process:

- Sprinkler is given a list of inputs, including a username or username file, passwords, and a target destination as well as any options the user wants to enable
- Sprinkler reformats the inputs into login attempts and sends them off
- As soon as the target sends back a successful login message, Sprinkler marks that attempt as a match for the user and stops spraying to that user
- Continues spraying until every user has had a login or every supplied password fails
- Sprinkler prints out the valid logins

Optimizations for speed

- Multithreading, or using different parts of the processor to perform multiple attempts simultaneously
- Working directly with computer memory, which lets us choose what information we need on hand to perform actions faster

Acknowledgements: Jeff Ondich, knowledgable users on StackOverflow, and libssh