

CSCI 331:  
Introduction to Computer Security

Lecture 10: Stack Smashing

Instructor: Dan Barowy  
**Williams**

## Announcements

- TA applications open tomorrow; due by Oct 27.
- TA feedback survey Oct 27.



## Announcements

- **Midterm exam**, in class, Thursday, Oct 19.
- Colloquium: **What I Did Last Summer (Research Edition)**, 2:35pm in Wege Auditorium.



## Topics

Rainbow Table Generation

Rainbow Table Lookup

A Sample Exploit

## Your to-dos

1. Start studying for midterm.
2. Lab 3 part 2, **due Sunday 10/15**.
3. Project part 2, out soon.

How I think about rainbow lookups

Class activity:  
generating rainbow chains  
of length 3

Hash function lookup table:

plaintext	Hash of plaintext
vvvv	4A7D1ED14474E4033AC29CCB8653D9B
vvv*	25BBD06C32D4777FA1C3E4A91B032
vv**	FC1198178C3594BFDDA3CA2996EB65CB
vv***	AE2BAC2E4B4DA805D01B2952D7E35BA4
v***	DB2F40F24260BC41DB48D82D5E7ABF1D
v***	814F06AB7F40B2CFF77F2C7BDFFD3415
v***	2A66ACBC1C39026B5D70457BB71B142B
v***	7D7C45B9A935CF9D845FC75679A41559
*vvv	A9B7BA70783B617E9998DC4DD82EB3C5
**vv	B8C37E33DEFDE51CF91E1E03E51657DA
***v	1E48C4420B7073BC11916C6C1DE226BB
****	7F975A56C761DB6506ECA0B37CE6EC87
****	1E8E0A04D20F50967C64DAC2D639A577
****	C6BFF625DB0393992C9D4DB0C6BBE45
****	2CBGA44843A864533EC05B321AE1F9D1
****	B59C67BF196A4758191E42F76670CEBA

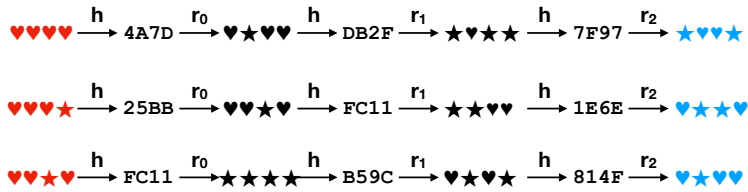
func reducer(c,i):

Convert the ith hexadecimal digit of c into a plaintext using the following table:

hex	plaintext
0	vvvv
1	vvv*
2	vv**
3	vv***
4	v***
5	v***
6	v***
7	v***
8	*vvv
9	**vv
A	***v
B	****
C	****
D	****
E	****
F	****

Find the first three rainbow chains of length 3.

### First three rainbow chains



end	start
★♥♥★	♥♥♥♥
♥★♥♥	♥♥♥★
♥♥♥♥	♥♥★♥

### Rainbow table (for first 3 chains)

end	start
★♥♥★	♥♥♥♥
♥★♥♥	♥♥♥★
♥♥♥♥	♥♥★♥

Decrypt FC11.

Hypothesis: FC11 is the third link in the chain.

$FC11 \xrightarrow{r_2} \heartsuit\heartsuit\heartsuit\star$  Is  $\heartsuit\heartsuit\heartsuit\star$  an end? **No.**

Hypothesis: FC11 is the second link in the chain.

$FC11 \xrightarrow{r_1} \star\star\star\heartsuit \xrightarrow{h} 1E6E \xrightarrow{r_2} \heartsuit\star\star\heartsuit$  Is  $\heartsuit\star\star\heartsuit$  an end? **Yes.**

Decrypt from **start**  $\heartsuit\heartsuit\heartsuit\star$ :



### Countermeasures Against Cracking Attacks

- Password salts.
- Uniformly-distributed passwords.
- Two-factor authentication.
- Last-known IP address.
- Make hashing expensive.

### Key Stretching

**Key stretching** is a technique used to make password decryption attacks **computationally expensive**. Unlike an ordinary user, an attacker must invoke a hash function many times. Key stretching **amplifies the cost of a hash function** using a **stretch factor s**.

$f^s(p) = c^s$  is an iterated hash function, where

$$f^1(p) = f(p) = c^1$$

$$f^2(p) = f(f(p)) = c^2$$

$$f^3(p) = f(f(f(p))) = c^3$$

...

$$f^n(p) = c^n$$

## Key Stretching

There are many publicly-available key stretching implementations. Two commonly-used implementations:

- bcrypt (default hash function in OpenBSD)
- PBKDF2 (part of the RSA encryption standard)

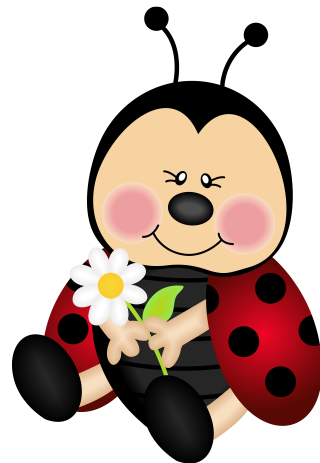
Both are interesting in that the stretch factor can be tied to available computational power.

### LastPass...|

We're also taking this as an opportunity to roll out something we've been planning for a while: PBKDF2 using SHA-256 on the server with a 256-bit salt utilizing 100,000 rounds. We'll be rolling out a second implementation of it with the client too. In more basic terms, this further mitigates the risk if we ever see something suspicious like this in the future. As we continue to grow we'll continue to find ways to reduce how large a target we are.

## Paper Discussion

## Bugs





## Anatomy of a bug

We can use the GCC assembler to figure out what's going on.

Note that I use a large number of GCC flags here. Don't let them scare you. These flags essentially reduce the program to its simplest form in assembly.

- -S : output assembly
- -c : compile as library (no C startup routines)
- -fno-dwarf2-cfi-asm : no control flow integrity
- -fno-asynchronous-unwind-tables : no exception support
- -fno-exceptions : exception support
- -z execstack : allow executable stacks
- -fno-stack-protector : disable stack canaries

## Recap & Next Class

### Today we learned:

Rainbow table generation

Rainbow table lookup

Sample buffer overflow exploit

### Next class:

Midterm review / low-level programming