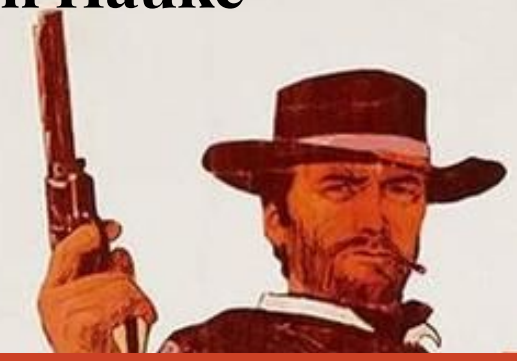
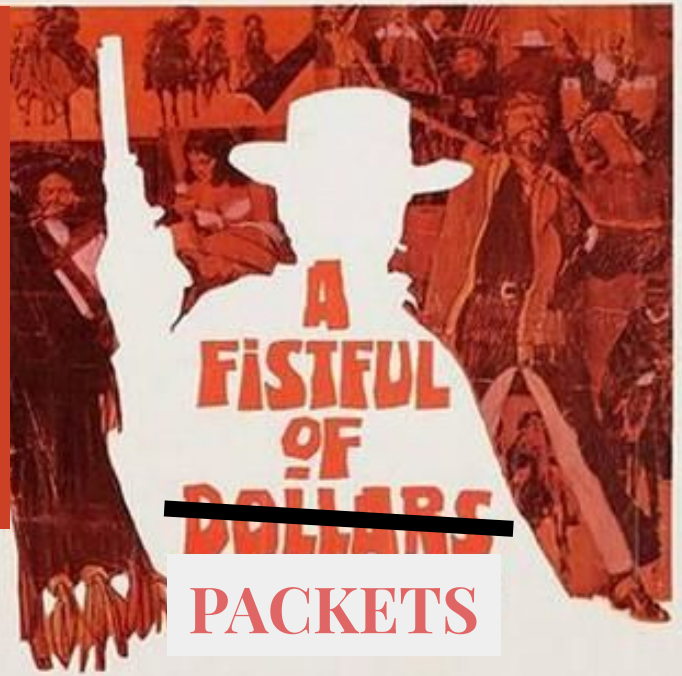
Nathan Hauke David Renardy



Denial of Service with a Fistful of Packets: Exploiting Algorithmic Complexity Vulnerabilities



Who are we?

- Security researchers at Two Six Labs
- One of us is a broomball national champion

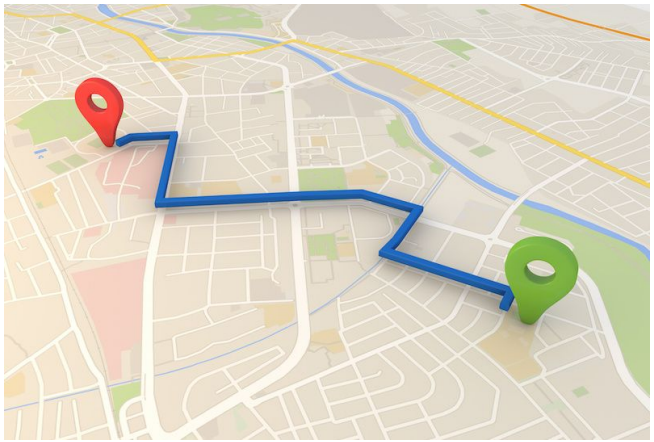
twosix LABS

Nathan Hauke

David Renardy

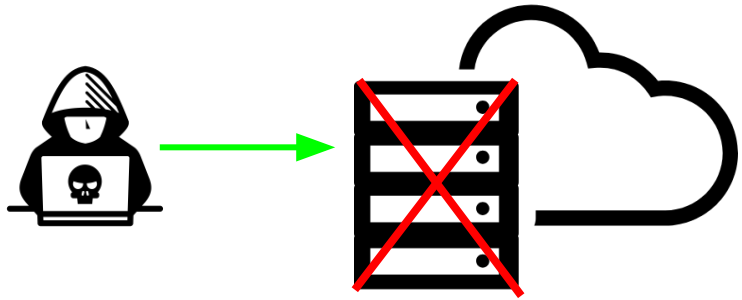
Talk Roadmap

- Algorithmic Complexity (AC) vulnerability recap
- 3 new AC vulnerabilities we discovered:
  - PDF specification
  - Linux VNC servers
  - Dropbox's zxcvbn algorithm
- Defense and Mitigations
  - ACsploit - Arsenal at 11:30



What is an AC Vulnerability?

- **Impact:** Resource consumption attack (DoS).
- **Cause:** Back-end algorithm has unacceptable worst-case performance.
- **Types:**
  - AC Time (CPU)
  - AC Space(memory).



Toy Example: Insertion Sort

- Best Case: Sorted
  - Linear time

6 5 3 1 8 7 2 4

- Worst Case: Reverse Sorted
  - Quadratic time

Our goal: find corner-case inputs to get worst-case performance

Our Story: Motivations and History

- There is a gap in awareness:

Our Story: Motivations and History

- There is a gap in awareness:
  - Application designers
  - Developers
  - Pen-testers
  - Vulnerability researchers

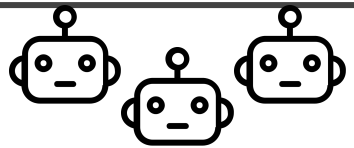
Our Story: Motivations and History

- There is a gap in awareness:
  - Application designers
  - Developers
  - Pen-testers
  - Vulnerability researchers

- We spent 3 years studying AC vulnerabilities while working on DARPA STAC

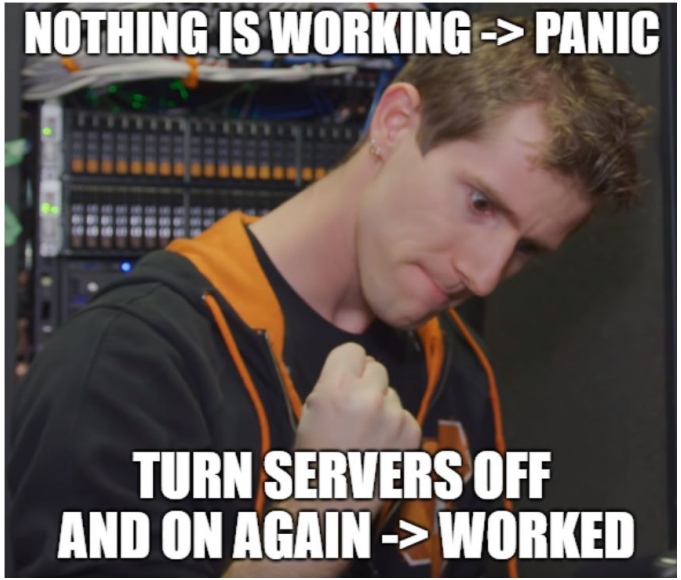
How do AC vulns differ from other vulnerabilities?

- Small inputs give significant effect. **No botnet needed.**

	Effort	Effect
		
AC 		

How do AC vulns differ from other vulnerabilities?

- AC vulnerabilities arise from **intended functionality.** AC vulns are not bugs!
- AC vulns arise from design decisions. Input is valid.
- Temporary DoS can result.

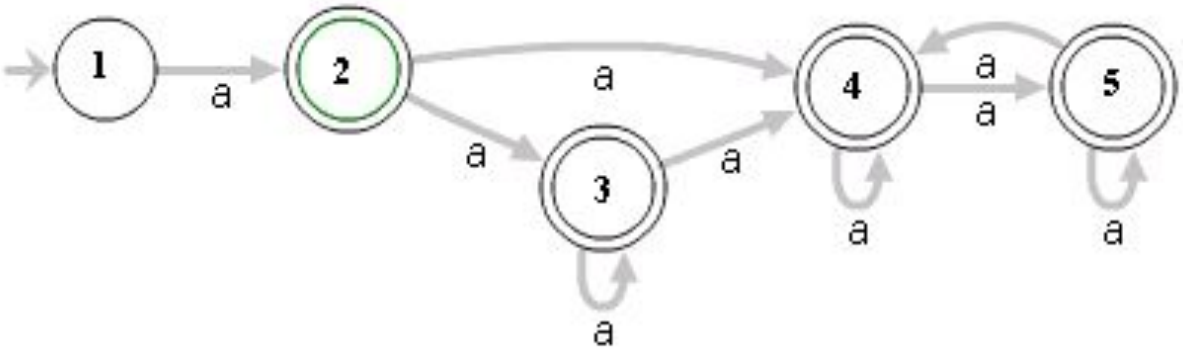


You've seen AC vulns before

- **29C3**: Dan Bernstein, Jean-Philippe Aumasson, Martin Boßlet - Hash-flooding DoS reloaded: attacks and defenses
- **BH-USA-2016**: Cara Marie - I Came to Drop Bombs
- **DEFCON-23**: Eric Davisson - REvisiting RE DoS

AC Vulns in the News: REDoS

- REDoS - leverage worst-case complexity of regular expression parsers to cause denial of service
- Ex: ^(a+)+$ “aaaab” traverses all 16 possible paths

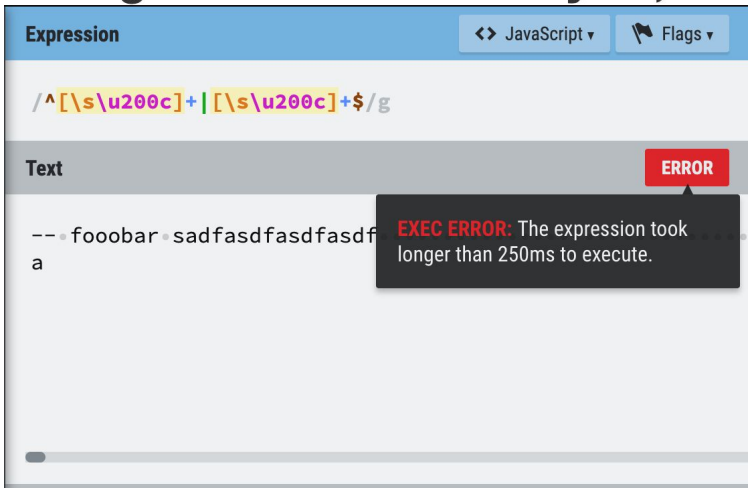


AC Vulns in the News: REDoS

Details of the Cloudflare outage on July 2, 2019

CPU utilization in one of our PoPs during the incident

StackExchange Outage Postmortem - July 20, 2016



Vulnerability 1:
An AC Time Vulnerability in the PDF Specification

PDF Decompression Bomb?

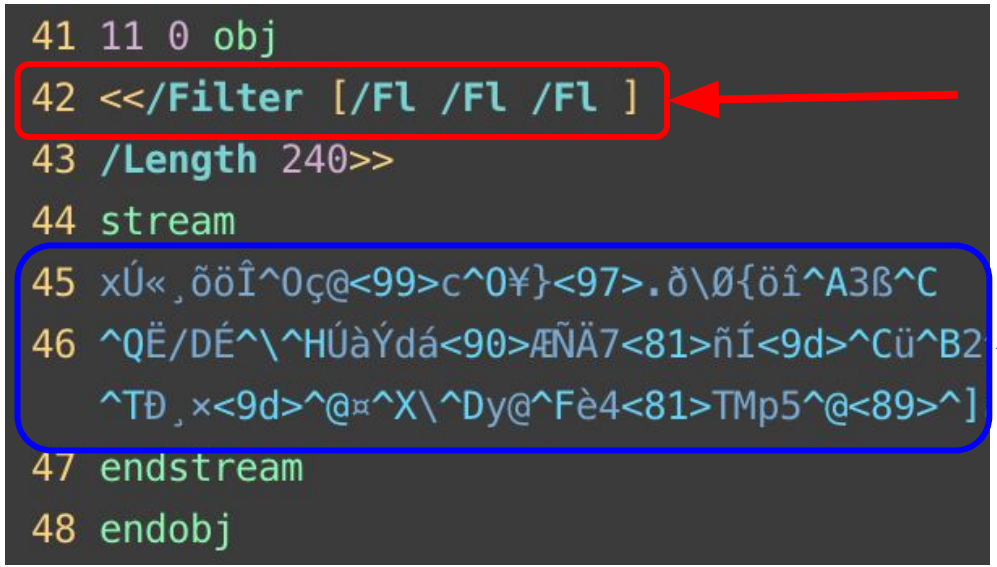
- **Effect:** AC time attack against PDF parser without going over a given memory ceiling

PDF Decompression ~~Bomb~~ Napalm?

- **Effect:** AC time attack against PDF parser without going over a given memory ceiling

We Didn't Start the Fire: Stevens' Bomb

PDFstream objects



Playing With Fire

Observations:

1. FlateDecode causes a small AC time effect

Playing With Fire

Observations:

1. FlateDecode causes a small AC time effect
2. A single PDF Page can hold multiple pdfstream objects

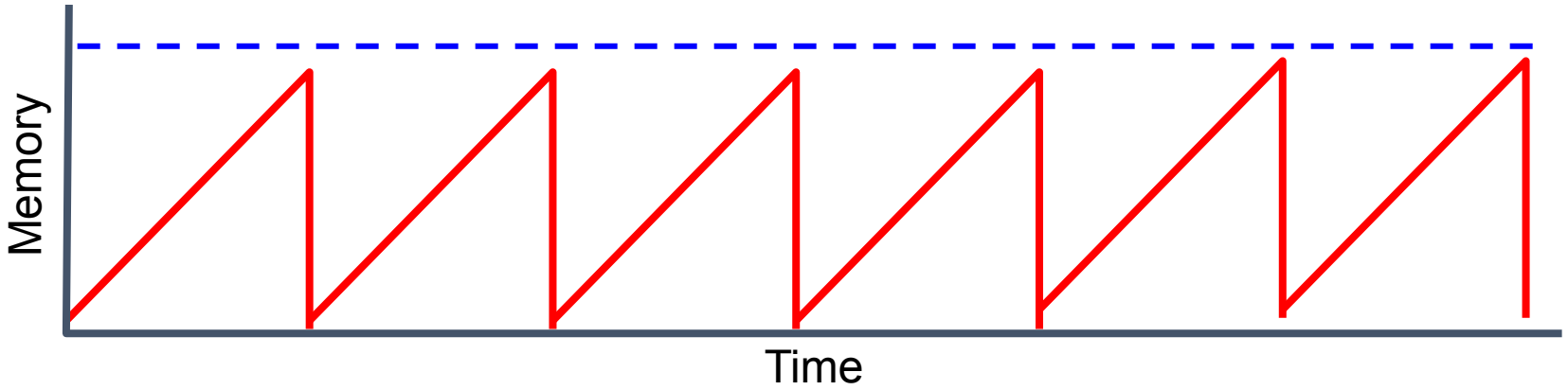
Playing With Fire

Observations:

1. FlateDecode causes a small AC time effect
2. A single PDF Page can hold multiple pdfstream objects

Challenge: Can we translate this memory (AC Space) vulnerability into an CPU (AC time) vulnerability?

Desired Effect



Only You Can Prevent OOM Errors

- Some filters shrink data: ASCIIHexDecode

  “53 6d 6f 6b 65 79” → Smokey

Only You Can Prevent OOM Errors

- Some filters shrink data: ASCIIHexDecode

  “53 6d 6f 6b 65 79” → Smokey

- **Idea**: FlateDecode to grow, and then ASCIIHexDecode to shrink.

Only You Can Prevent OOM Errors

- Some filters shrink data: ASCIIHexDecode

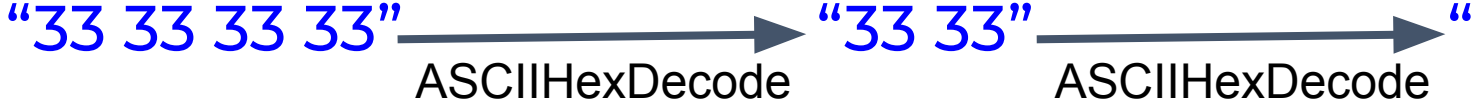
  “53 6d 6f 6b 65 79” → Smokey

- **Idea**: FlateDecode to grow, and then ASCIIHexDecode to shrink.
- **Problem**: ASCIIHexDecode needs valid hex

ASCIIHexDecode and a Trick

Trick:

0x33 is the ASCII encoding for the character “3”



A Small Fire

```
42 <</Filter [/Fl /Fl /Fl /AHx /AHx /AHx /AHx /AHx /AH
43 /Length 160 >>
44 stream
45 xÚ«¸õö¶5^_#£^HkÀá^P®^P©³G´L<84>f 4^Mîf}QÚÑ^Q<96>Ìo<
   6ì^T?kî'Àu^A^@^E<89>A^K
46 endstream
47 endobj
```

Recipe for Making PDF Napalm

1. Find or guess RAM limits

Recipe for Making PDF Napalm

1. Find or guess RAM limits
2. Deflate a bunch* of “3”s

Recipe for Making PDF Napalm

1. Find or guess RAM limits
2. Deflate a bunch* of “3”s
3. FlateDecode + ASCIIHexDecode filters

Recipe for Making PDF Napalm

1. Find or guess RAM limits
2. Deflate a bunch* of “3”s
3. FlateDecode + ASCIIHexDecode filters
4. Fill a PDF page with these mini bomb pdfstreams

PDF Napalm Demo

Impact

- Affects spec-compliant implementations

- Vulnerable targets include OCR apps

Mitigations

- Input sanitization:
  - Don’t allow repeated filters
  - Limit the number of pdfstream objects per page
- Resource controls:
  - Limit the memory / processing time

Vulnerability 2: Unauthenticated VNC Server Disk Space Consumption

What is a VNC Server?

- Remotely access computer
- Graphical view of desktop
- Compare with Remote Desktop Protocol (RDP)

```
19/03/2019 13:05:52 Got connection from client 10.3.3.134

19/03/2019 13:05:52 Got connection from client 10.3.3.134
19/03/2019 13:05:52   (other clients 10.3.3.134)

19/03/2019 13:05:52 Got connection from client 10.3.3.134
19/03/2019 13:05:52   (other clients 10.3.3.134 10.3.3.134)

19/03/2019 13:05:52 Got connection from client 10.3.3.134
19/03/2019 13:05:52   (other clients 10.3.3.134 10.3.3.134 10.3.3.134)

19/03/2019 13:05:53 Got connection from client 10.3.3.134
19/03/2019 13:05:53   (other clients 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134)

19/03/2019 13:05:53 Got connection from client 10.3.3.134
19/03/2019 13:05:53   (other clients 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134)

19/03/2019 13:05:53 Got connection from client 10.3.3.134
19/03/2019 13:05:53   (other clients 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134)

19/03/2019 13:05:53 Got connection from client 10.3.3.134
19/03/2019 13:05:53   (other clients 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134 10.3.3.134)
```

VNC Server Disk Space Consumption

```
348     if (rfbClientHead == NULL) {
349       /* no other clients - make sure we don't think any keys are pressed */
350       KbdReleaseAllKeys();
351     } else {
352       rfbLog("  (other clients");
353       for (cl = rfbClientHead; cl; cl = cl->next) {
354         fprintf(stderr, " %s", cl->host);
355       }
356       fprintf(stderr, ")\n");
357     }
```

VNC Server Disk Space Consumption

```
348     if (rfbClientHead == NULL) {
349       /* no other clients - make sure we don't think any keys are pressed */
350       KbdReleaseAllKeys();
351     } else {
352       rfbLog("  (other clients");
353       for (cl = rfbClientHead; cl; cl = cl->next) {
354         fprintf(stderr, " %s", cl->host);
355       }
356       fprintf(stderr, ")\n");
357     }
```

Print the IP address of every connected client

Recipe for Exploiting Disk Space

Recipe for Exploiting Disk Space

1. Create multiple TCP connections to the VNC server

Recipe for Exploiting Disk Space

1. Create multiple TCP connections to the VNC server
2. Keep connections open

Recipe for Exploiting Disk Space

1. Create multiple TCP connections to the VNC server
2. Keep connections open
3. Every connection adds a longer line to the log file

Recipe for Exploiting Disk Space

1. Create multiple TCP connections to the VNC server
2. Keep connections open
3. Every connection adds a longer line to the log file
4. Log file size is $O(n^2)$ where n is the number of connections

VNC Demo #1

Vulnerability 2 Bonus: Infinite Logging & Denial of Service

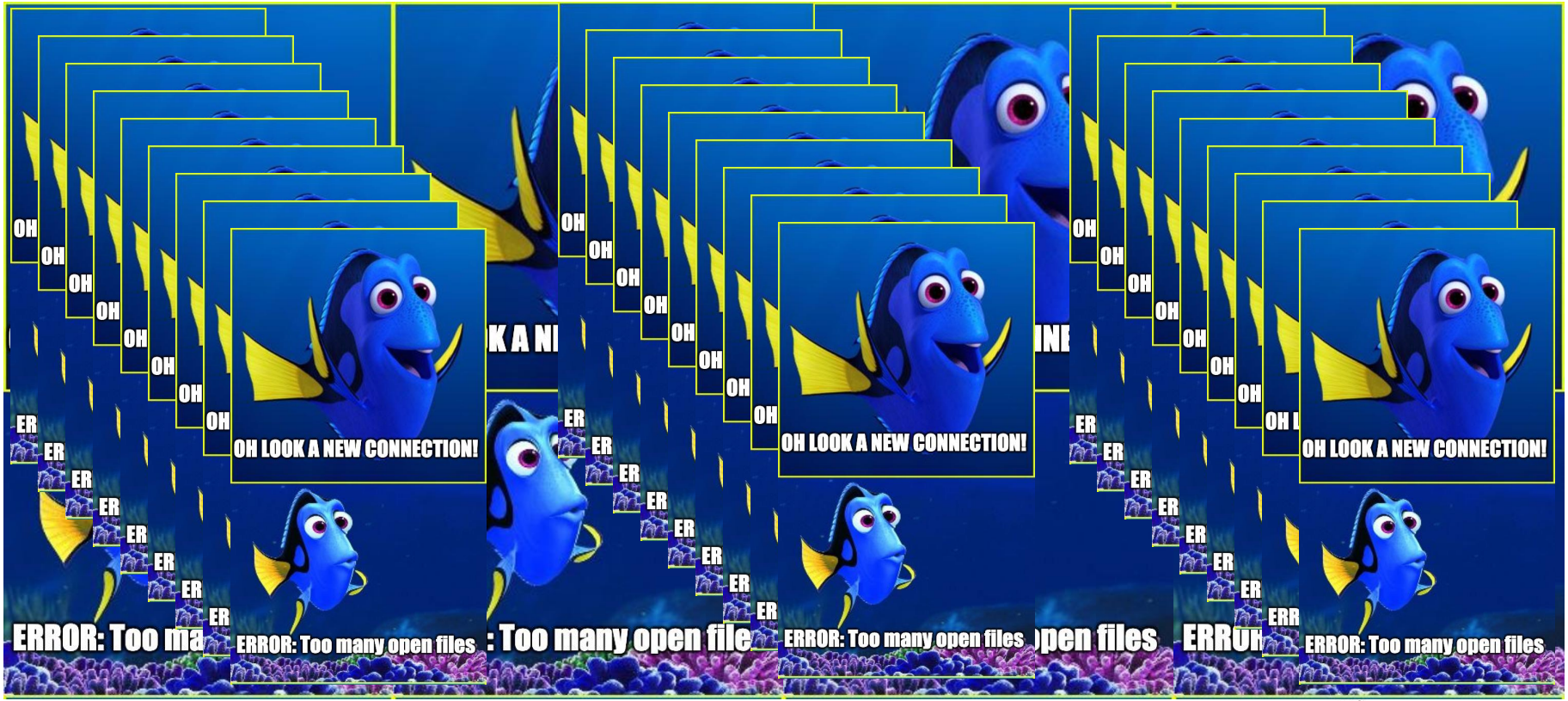
Some Innocuous Code

```
178     if ((sock = accept(rfbListenSock, &addr.u.sa, &addrlen)) < 0) {
179       rfbLogPerror("rfbSockNotify: accept");
180       return;
181     }
```

Or is it?

- What happens if we run out of file descriptors?
- EMFILE error
- New connection still needs to be processed


OH LOOK A NEW CONNECTION!
OH LOOK A NEW CONNECTION!
OH LOOK A NEW CONNECTION!
OH LOOK A NEW CONNECTION!
ERROR: Too many open files
ERROR: Too many open files
ERROR: Too many open files
ERROR: Too many open files


OH
ER
OH LOOK A NEW CONNECTION!
ERROR: Too many open files

Recipe for Exploiting Disk Space & Time

1. Create multiple TCP connections to the VNC server
2. Keep connections open

Recipe for Exploiting Disk Space & Time

1. Create multiple TCP connections to the VNC server
2. Keep connections open
3. Repeat until the server process is out of file descriptors (~1024)

Recipe for Exploiting Disk Space & Time

1. Create multiple TCP connections to the VNC server
2. Keep connections open
3. Repeat until the server process is out of file descriptors (~1024)
4. Next connection attempt triggers infinite loop

VNC Demo #2

Impact

- Multiple affected servers:
  - TightVNC
  - TurboVNC
  - Vino
  - LibVNCServer
  - x11VNC

- No authentication required

Mitigations

TurboVNC / LibVNCServer / x11VNC:

- Don’t log the list of other clients

- Limit the maximum number of client connections

Vulnerability 3: Unauthenticated Denial of Service in Dropbox’s zxcvbn

What is zxcvbn?

- Estimate difficulty for an attacker to guess your password

- Designed to replace archaic password policy

How does zxcvbn work?

How does zxcvbn work?

- n@thanPassword080819

How does zxcvbn work?

- n@thanPassword080819

How does zxcvbn work?

- n@thanPassword080819

```
'n@than'
pattern:                dictionary
guesses_log10:          2.32634
dictionary_name:        male_names
rank:                   106
reversed:               false
l33t subs:              @ -> a
un-l33ted:              nathan
base-guesses:           106
uppercase-variations:   1
l33t-variations:        2
```

```
'Password'
pattern:                dictionary
guesses_log10:          1.69897
dictionary_name:        passwords
rank:                   2
reversed:               false
base-guesses:           2
uppercase-variations:   2
l33t-variations:        1
```

```
'080819'
pattern:        date
guesses_log10:  3.86332
day:            8
month:          8
year:           2019
separator:      ''
```

How does zxcvbn work?

- n@thanPassword080819

```
'n@than'
pattern:                dictionary
guesses_log10:          2.32634
dictionary_name:        male_names
rank:                   106
reversed:               false
l33t subs:              @ -> a
un-l33ted:              nathan
base-guesses:           106
uppercase-variations:   1
l33t-variations:        2
```

```
'Password'
pattern:                dictionary
guesses_log10:          1.69897
dictionary_name:        passwords
rank:                   2
reversed:               false
base-guesses:           2
uppercase-variations:   2
l33t-variations:        1
```

```
'080819'
pattern:        date
guesses_log10:  3.86332
day:            8
month:          8
year:           2019
separator:      ''
```

L33T Substitution

p@ssw0rd

L33T Substitution

p@ssw0rd

L33T Substitution

p@ssw0rd ⟶ {‘@’: ‘a’, ‘0’: ‘o’}

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

b|ackh@t

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

b|ackh@t → {'|': 'i', '@': 'a'}

b|ackh@t → {'|': 'l', '@': 'a'}

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

b|ackh@t → {'|': 'i', '@': 'a'} → biackhat

→ {'|': 'l', '@': 'a'}

L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

b|ackh@t → {'|': 'i', '@': 'a'} → biackhat → ●

{'|': 'l', '@': 'a'}

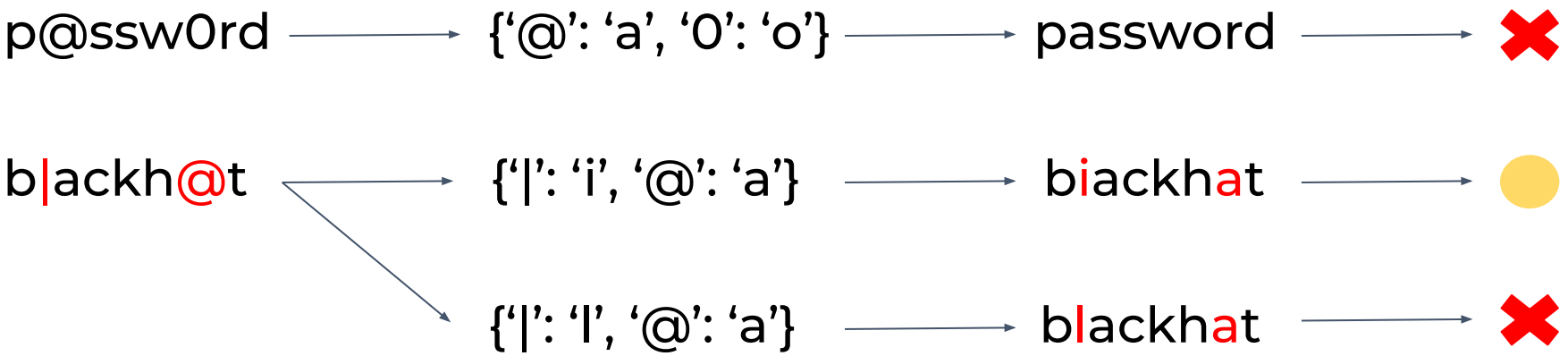
L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✖

b|ackh@t → {'|': 'i', '@': 'a'} → biackhat → ●

b|ackh@t → {'|': 'l', '@': 'a'} → blackhat

L33T Substitution



L33T Substitution

p@ssw0rd → {'@': 'a', '0': 'o'} → password → ✗

b|ackh@t → {'|': 'i', '@': 'a'} → biackhat → ●

b|ackh@t → {'|': 'l', '@': 'a'} → blackhat → ✗

Ambiguous characters:

| 1 7

L33T Substitution

1o77|pop

L33T Substitution

1o77|pop

- {'1': 'i', '7': 'l', '|': 'i'}
- {'1': 'i', '7': 'l', '|': 'l'}
- {'1': 'i', '7': 't', '|': 'i'}
- {'1': 'i', '7': 't', '|': 'l'}
- {'1': 'l', '7': 'l', '|': 'i'}
- {'1': 'l', '7': 'l', '|': 'l'}
- {'1': 'l', '7': 't', '|': 'i'}
- {'1': 'l', '7': 't', '|': 'l'}

L33T Substitution

1o77|pop

{'1': 'i', '7': 'l', '|': 'i'} → iollipop

{'1': 'i', '7': 'l', '|': 'l'} → iolllpop

{'1': 'i', '7': 't', '|': 'i'} → iottipop

{'1': 'i', '7': 't', '|': 'l'} → iottlpop

{'1': 'l', '7': 'l', '|': 'i'} → lollipop

{'1': 'l', '7': 'l', '|': 'l'} → lolllpop

{'1': 'l', '7': 't', '|': 'i'} → lottipop

{'1': 'l', '7': 't', '|': 'l'} → lottlpop

L33T Substitution

1o77|pop

{'1': 'i', '7': 'l', '|': 'i'} → iollipop → ●
{'1': 'i', '7': 'l', '|': 'l'} → iolllpop → ●
{'1': 'i', '7': 't', '|': 'i'} → iottipop → ●
{'1': 'i', '7': 't', '|': 'l'} → iottlpop → ●
{'1': 'l', '7': 'l', '|': 'i'} → lollipop → ✖
{'1': 'l', '7': 'l', '|': 'l'} → lolllpop → ●
{'1': 'l', '7': 't', '|': 'i'} → lottipop → ●
{'1': 'l', '7': 't', '|': 'l'} → lottlpop → ●

What's the worst that could happen?

Recipe for extended zxcvbn runtime:

What's the worst that could happen?

Recipe for extended zxcvbn runtime:

1. Make the password as long as possible

What's the worst that could happen?

Recipe for extended zxcvbn runtime:

1. Make the password as long as possible

2. Use the l33t characters that have multiple possible substitutions | 1 7

What's the worst that could happen?

Recipe for extended zxcvbn runtime:

1. Make the password as long as possible

2. Use the l33t characters that have multiple possible substitutions `| 1 7`

3. Use every l33t character `4@8({[<369!0$5{%2`

What's the worst that could happen?

4@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<36
91!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$
5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@
8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!
|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7
%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[
<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|7
0$5{7%24@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24
@8({[<3691!|1|70$5{7%24@8({[<3691!|1|70$5{7%24@8({[<369

What's the worst that could happen?

Password length (chars)	Worst-case password	DropBox says
100		0.1 s
200		N/A
1000		N/A

What's the worst that could happen?

Password length (chars)	Worst-case password	DropBox says
100	5.7 s	0.1 s
200	24.4 s	N/A
1000	22.1 min	N/A

Impact

- Implementations in many different programming languages

Impact

- Implementations in many different programming languages
- Used in enterprise software

Impact

- Implementations in many different programming languages
- Used in enterprise software
- Attacks user signup page

zxcvbn Demo

Mitigations

- Input sanitization
  - Evaluate first n bytes of password

- Better algorithms
  - Improve quadratic time dictionary match algorithm

Conclusion

Defensive Measures and Mitigations

- Select better algorithms
- Don’t just design for the average case
- Use proper input sanitization

ACsploit

- Generate worst-case inputs to common algorithms
- REDoS identification
- PoCs releasing today, open source:

https://github.com/twosixlabs/acsploit

- Check it out at Arsenal at 11:30 Business Hall (Oceanside), Arsenal Station 3!

Black Hat Sound Bytes

- Pen-testers: Incorporate AC vulnerabilities as part of your testing.
- Developers: Develop with worst-case inputs in mind.
- Researchers: “See something. Say something.”

Questions?

Blog: https://www.twosixlabs.com/blog/

Contact:

- david.renardy@twosixlabs.com
- nathan.hauke@twosixlabs.com

ACsploit Arsenal 11:30 Business Hall (Oceanside), Arsenal Station 3!