THE INTERNET of ~~THINGS~~
SHIT...
OR NET OF A MILLION SPIES

Overflows & Format Strings

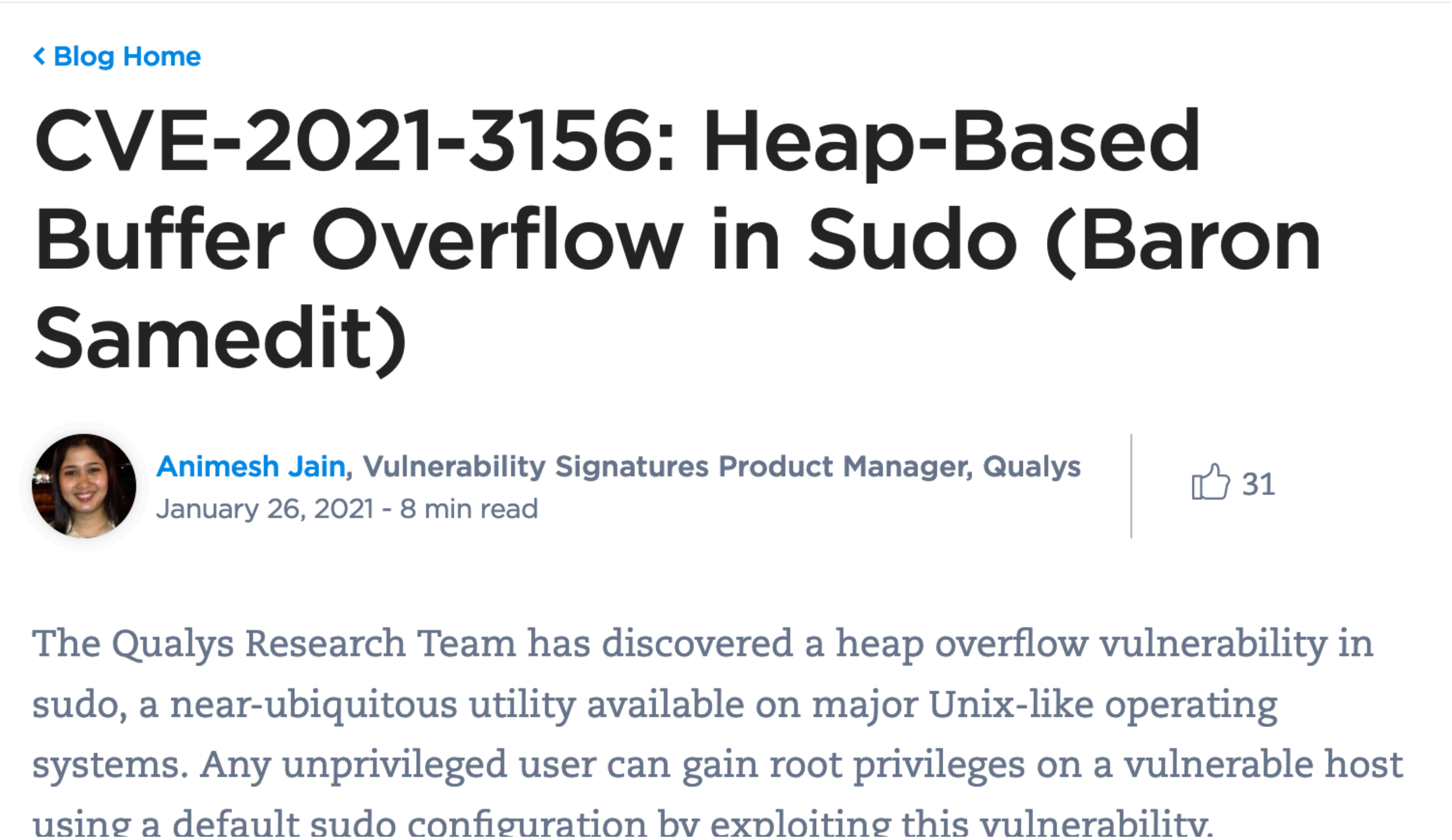
Announcements...

- CS61C (Call Frame) review session:
  Friday 1-3PM Pacific
- Project 1 and Homework 1 are both out:
  HW1 due Friday
  Project 1 due Friday 2/19
- Zoom protocol: If the chat annoys you, just scroll it off screen

Internet of Shit...

- A device produced by the lowest bidder...
  - That you then connect through the network
- This has a very wide ***attack surface***
  - Methods where an attacker might access a vulnerability
- And its often incredibly ***cost sensitive***
  - Very little support after purchase
    - So things don't get patched
  - No way for the user to tell what is "secure" or "not"
    - But they can tell what is cheaper!
    - And often it is ***insanely*** insecure:
      Default passwords on telnet of admin/admin...
      Trivial buffer overflows

And Speaking of Internet-Of-Shit Targeting Bugs: sudo...



< Blog Home

CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit)

Animesh Jain, Vulnerability Signatures Product Manager, Qualys
January 26, 2021 - 8 min read

👍 31

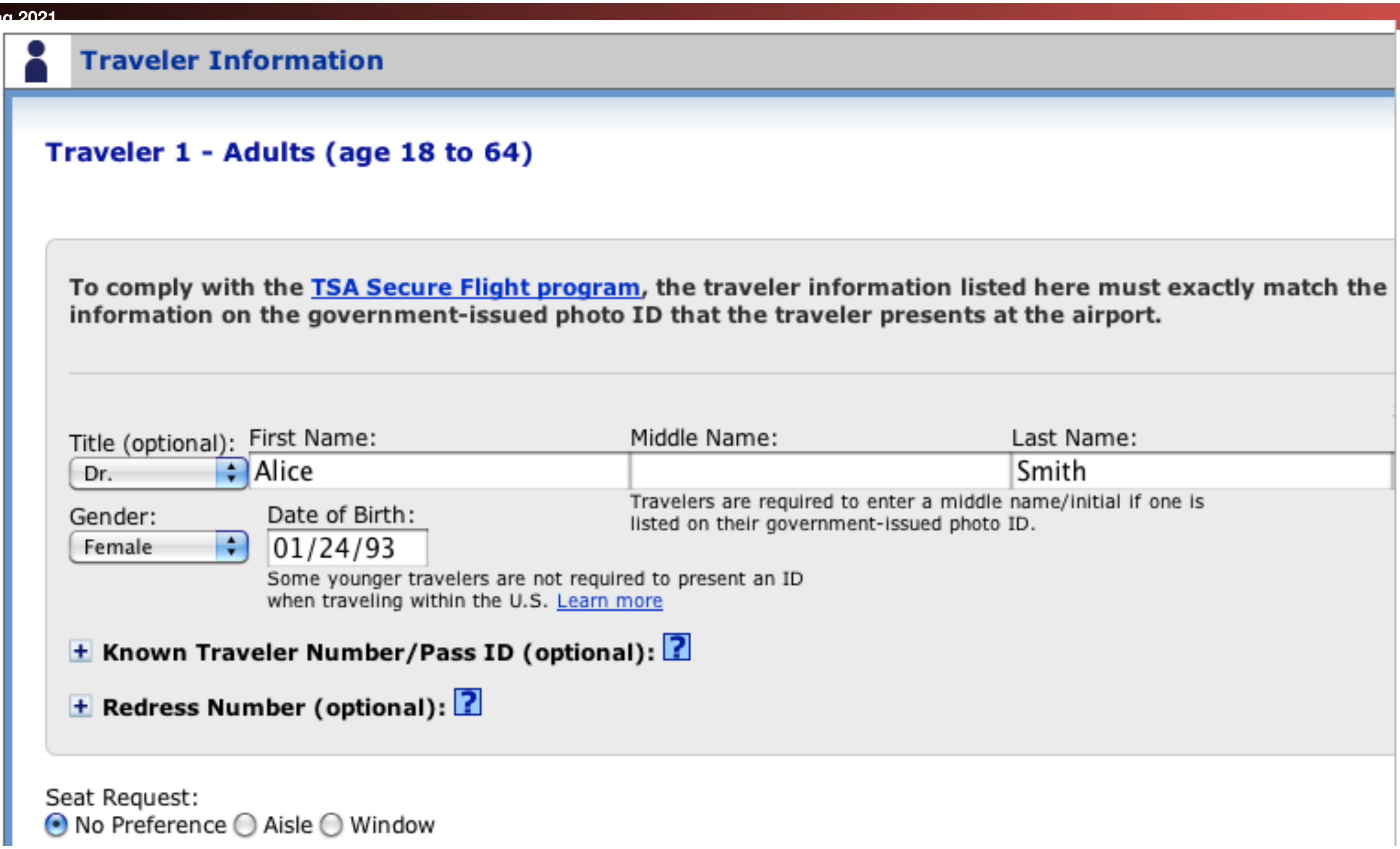
The Qualys Research Team has discovered a heap overflow vulnerability in sudo, a near-ubiquitous utility available on major Unix-like operating systems. Any unprivileged user can gain root privileges on a vulnerable host using a default sudo configuration by exploiting this vulnerability.

Net Of A Million Spies...

- Device only communicates through a central service
  - Greatly reduces the attack surface but...
- Most of the companies running the service are "Data Asset" companies
  - Make their money from advertising, not the product themselves
    - May actually subsidize the product considerably
  - Some you know about: Google, Amazon
  - Some you may not: Salesforce
- Only exception of note is Apple:
  - I may talk about HomeKit later...
    But you still have to trust that the HomeKit product doesn't report to a third party.

A Tale from the Before Times

- I (used) to fly a fair amount
  - Conferences, panels, meetings, etc...



Traveler Information

Traveler 1 - Adults (age 18 to 64)

To comply with the TSA Secure Flight program, the traveler information listed here must exactly match the information on the government-issued photo ID that the traveler presents at the airport.

Title (optional): Dr.

First Name: Alice

Middle Name:

Last Name: Smith

Travelers are required to enter a middle name/initial if one is listed on their government-issued photo ID.

Gender: Female

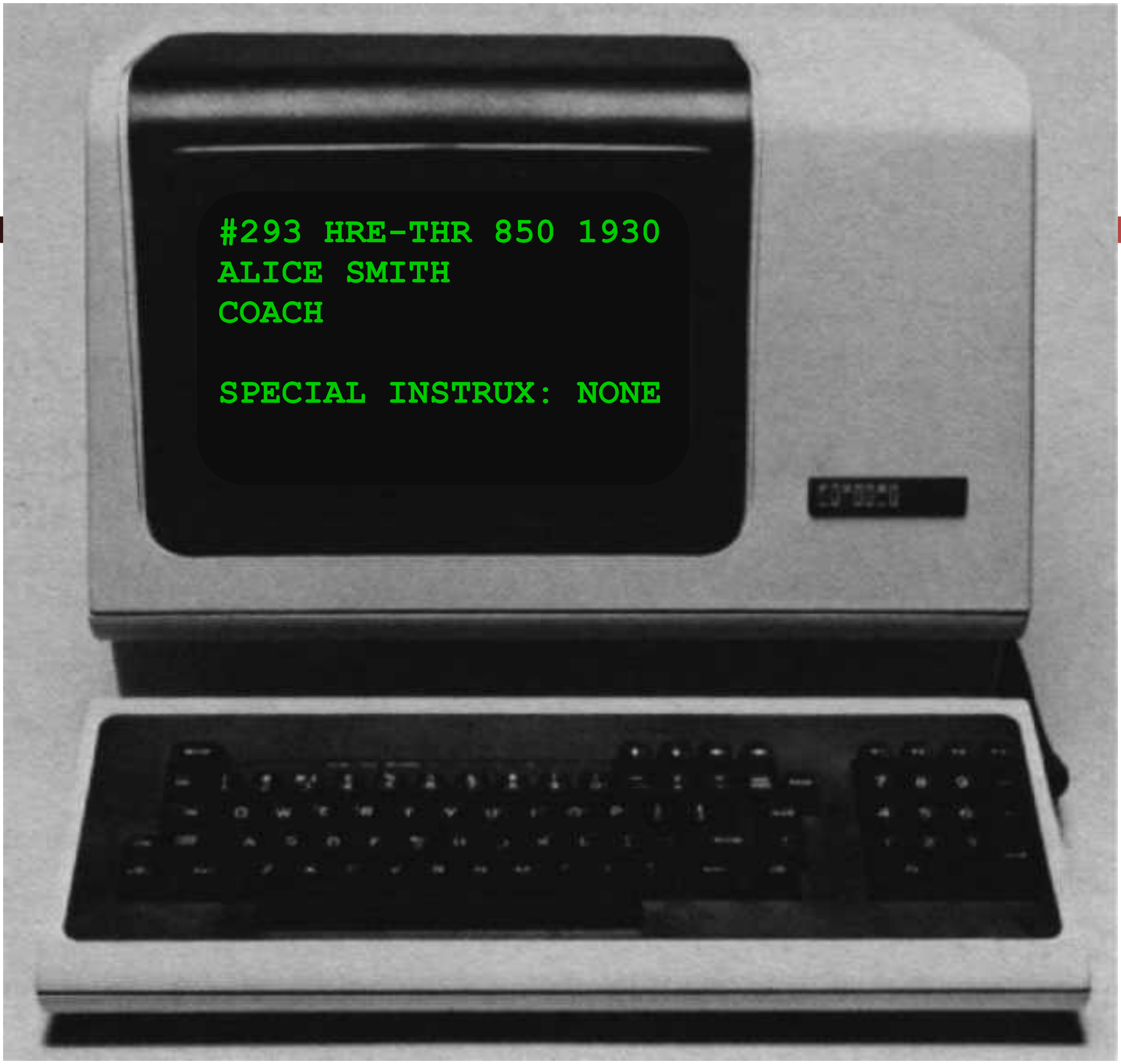
Date of Birth: 01/24/93

Some younger travelers are not required to present an ID when traveling within the U.S. Learn more

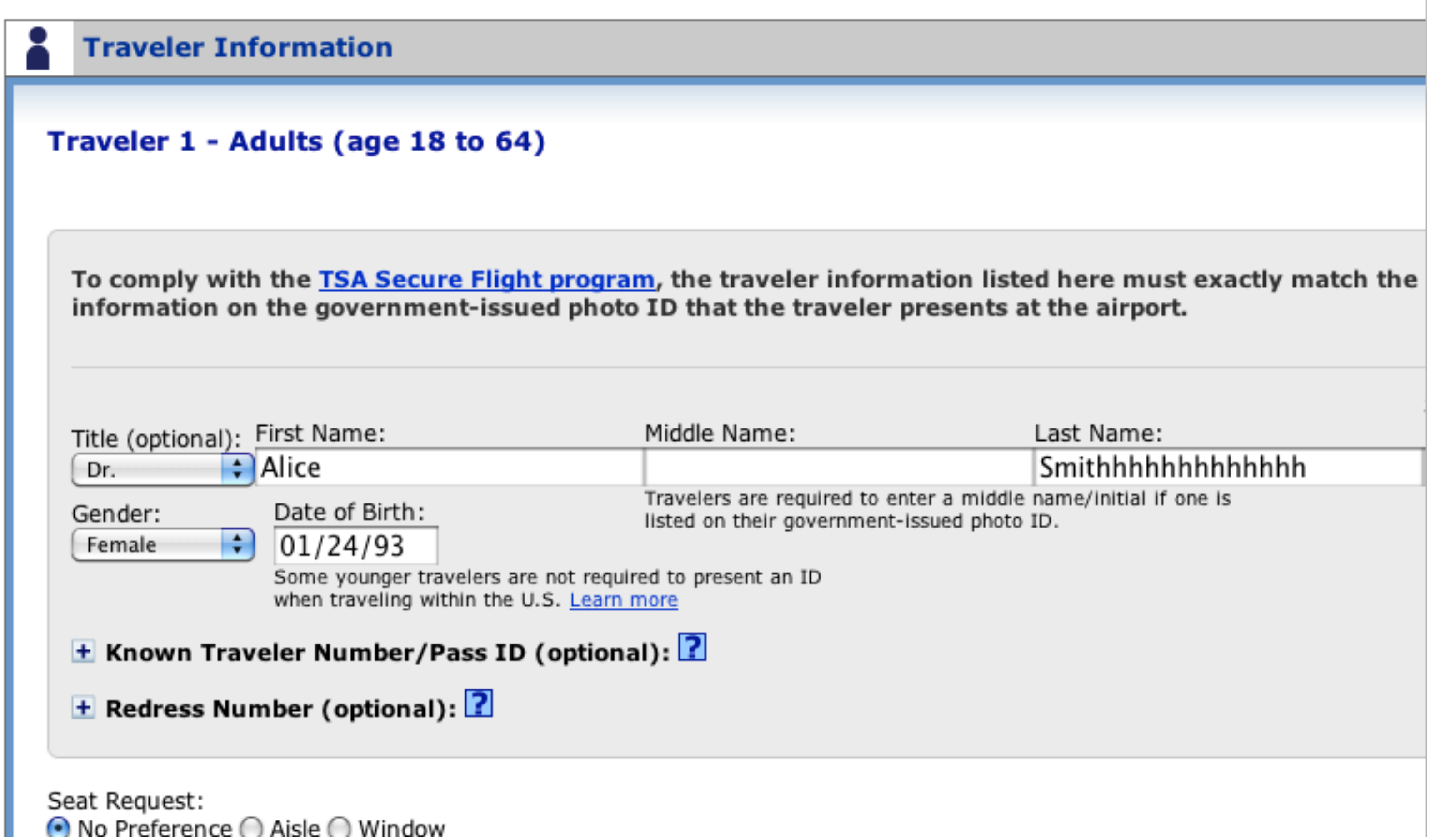
+ Known Traveler Number/Pass ID (optional): ?

+ Redress Number (optional): ?

Seat Request:
● No Preference ○ Aisle ○ Window



```
#293 HRE-THR 850 1930
ALICE SMITH
COACH

SPECIAL INSTRUX: NONE
```



Traveler Information

Traveler 1 - Adults (age 18 to 64)

To comply with the TSA Secure Flight program, the traveler information listed here must exactly match the information on the government-issued photo ID that the traveler presents at the airport.

Title (optional): Dr.

First Name: Alice

Middle Name:

Last Name: Smithhhhhhhhhhhh

Travelers are required to enter a middle name/initial if one is listed on their government-issued photo ID.

Gender: Female

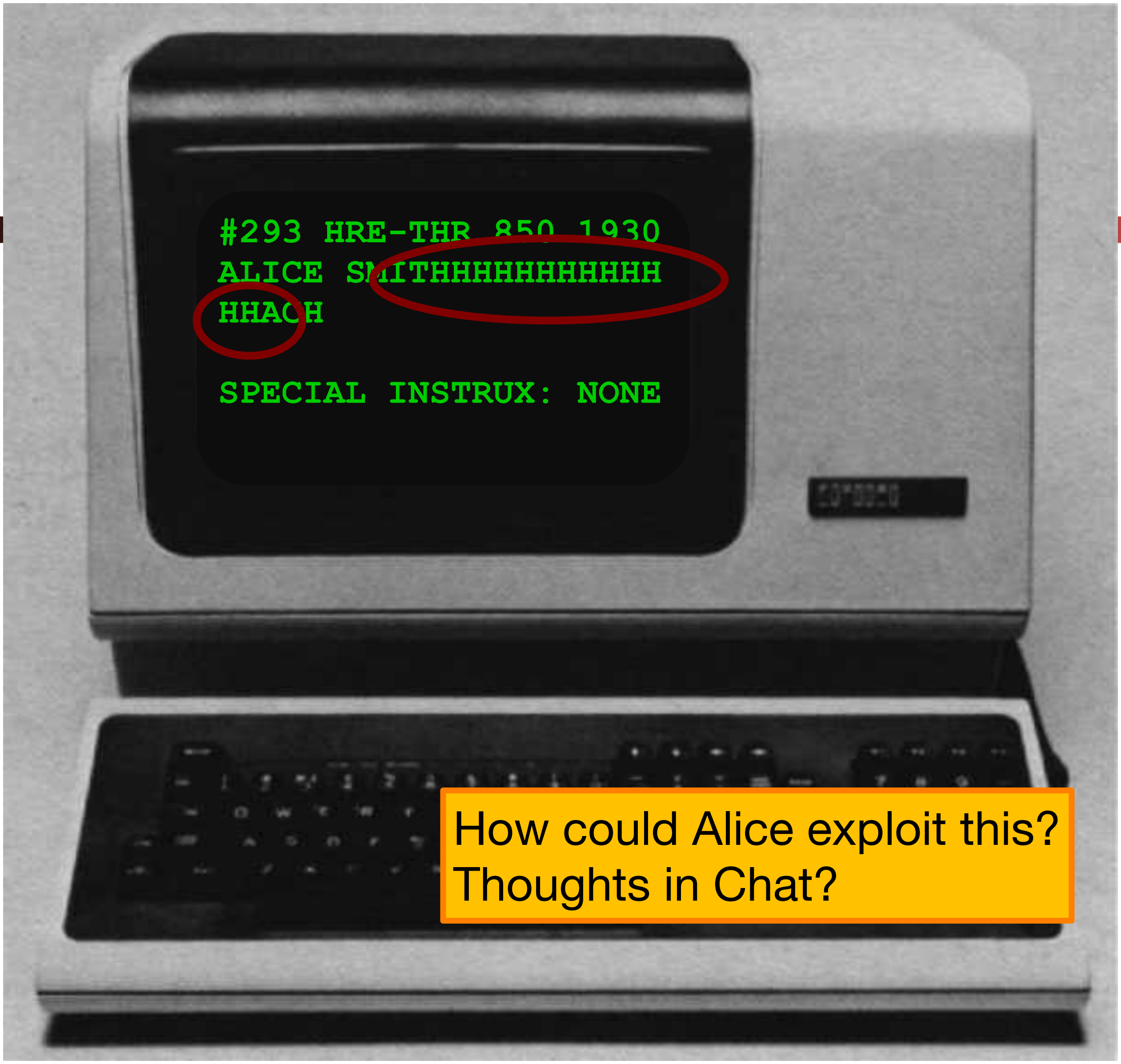
Date of Birth: 01/24/93

Some younger travelers are not required to present an ID when traveling within the U.S. Learn more

+ **Known Traveler Number/Pass ID (optional):** ?

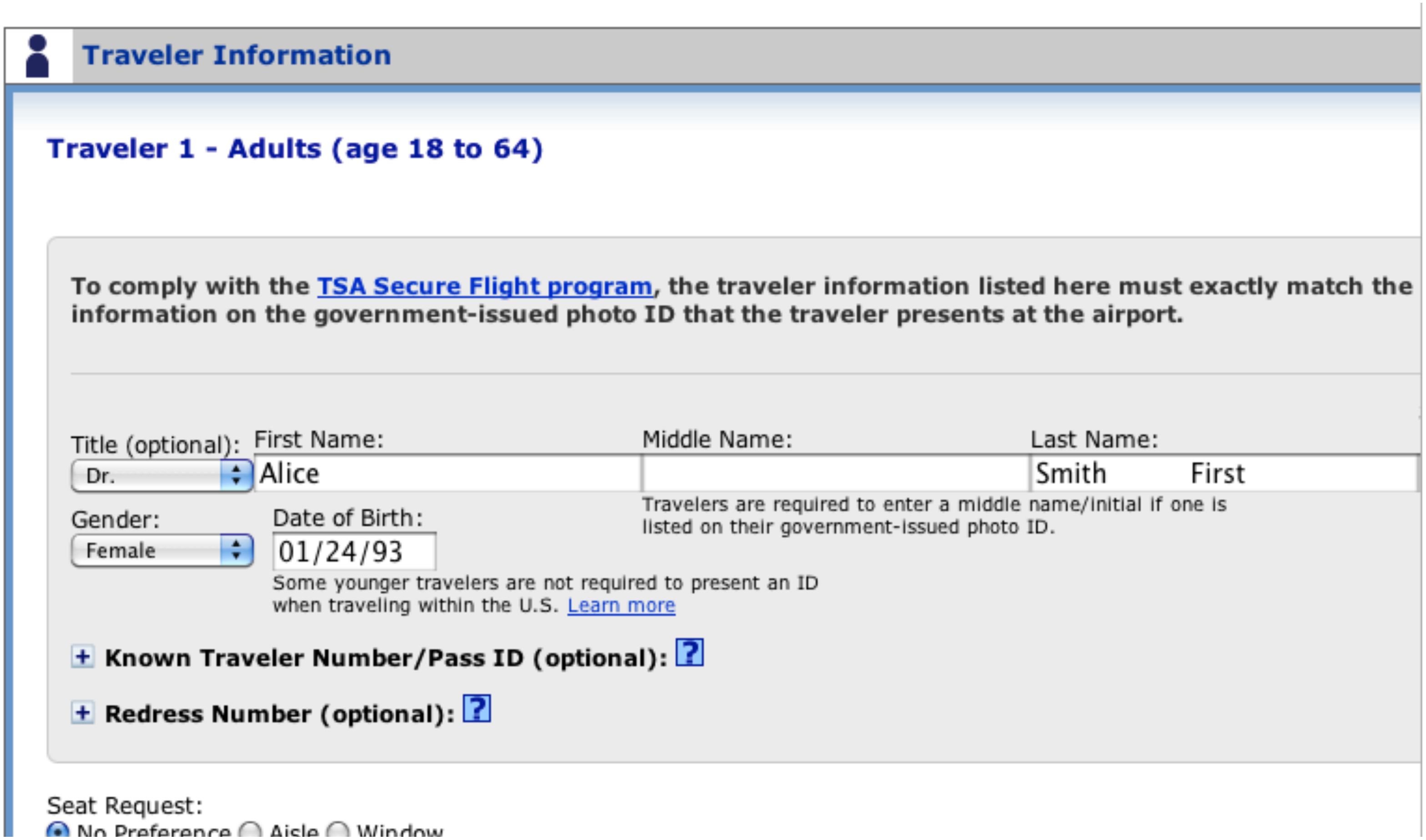
+ **Redress Number (optional):** ?

Seat Request:
● No Preference ○ Aisle ○ Window



```
#293 HRE-THR 850 1930
ALICE SMITHHHHHHHHHHH
HHACH

SPECIAL INSTRUX: NONE
```

How could Alice exploit this?
Thoughts in Chat?



Traveler Information

Traveler 1 - Adults (age 18 to 64)

To comply with the TSA Secure Flight program, the traveler information listed here must exactly match the information on the government-issued photo ID that the traveler presents at the airport.

Title (optional): Dr.

First Name: Alice

Middle Name:

Last Name: Smith First

Travelers are required to enter a middle name/initial if one is listed on their government-issued photo ID.

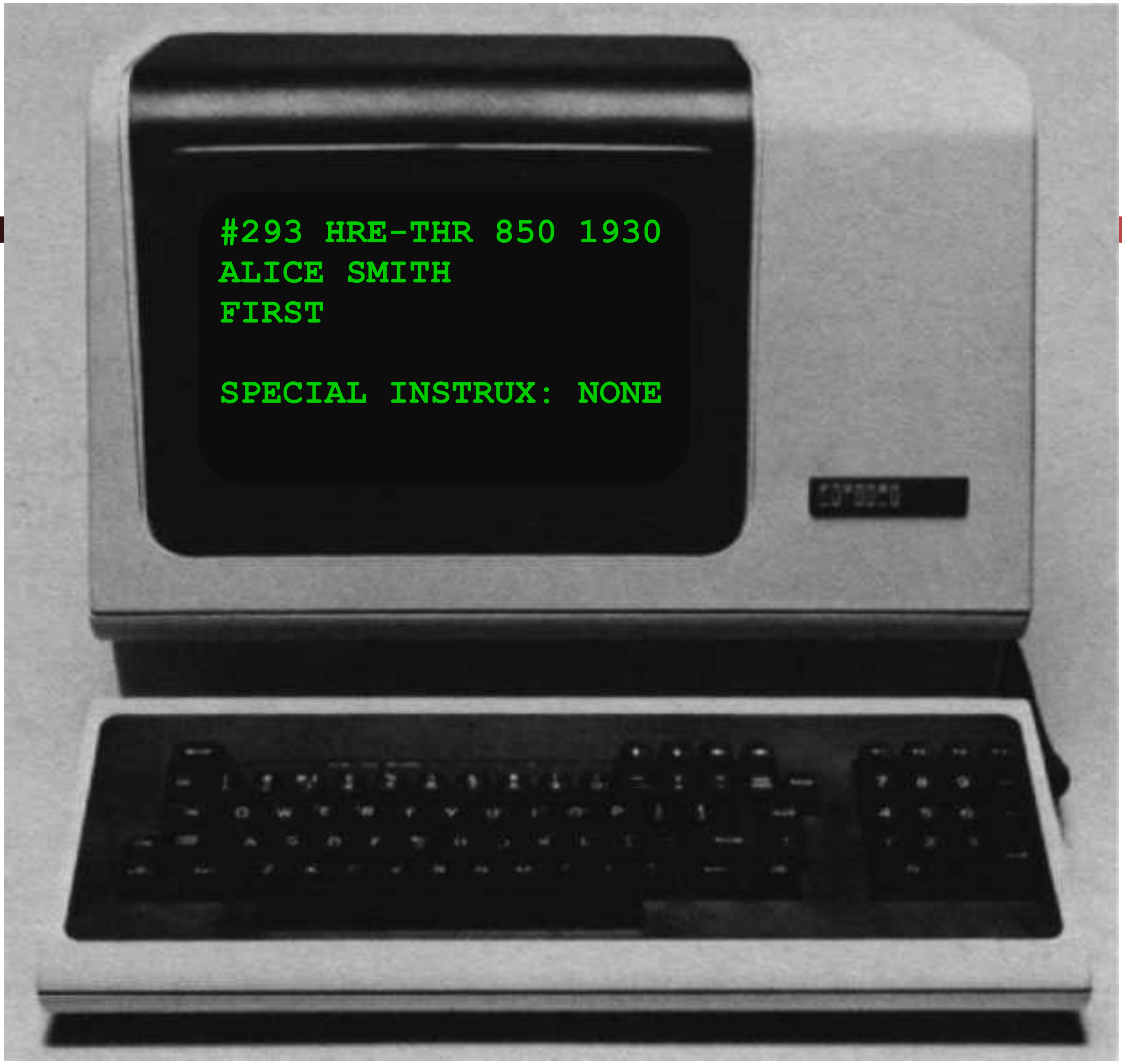
Gender: Female

Date of Birth: 01/24/93

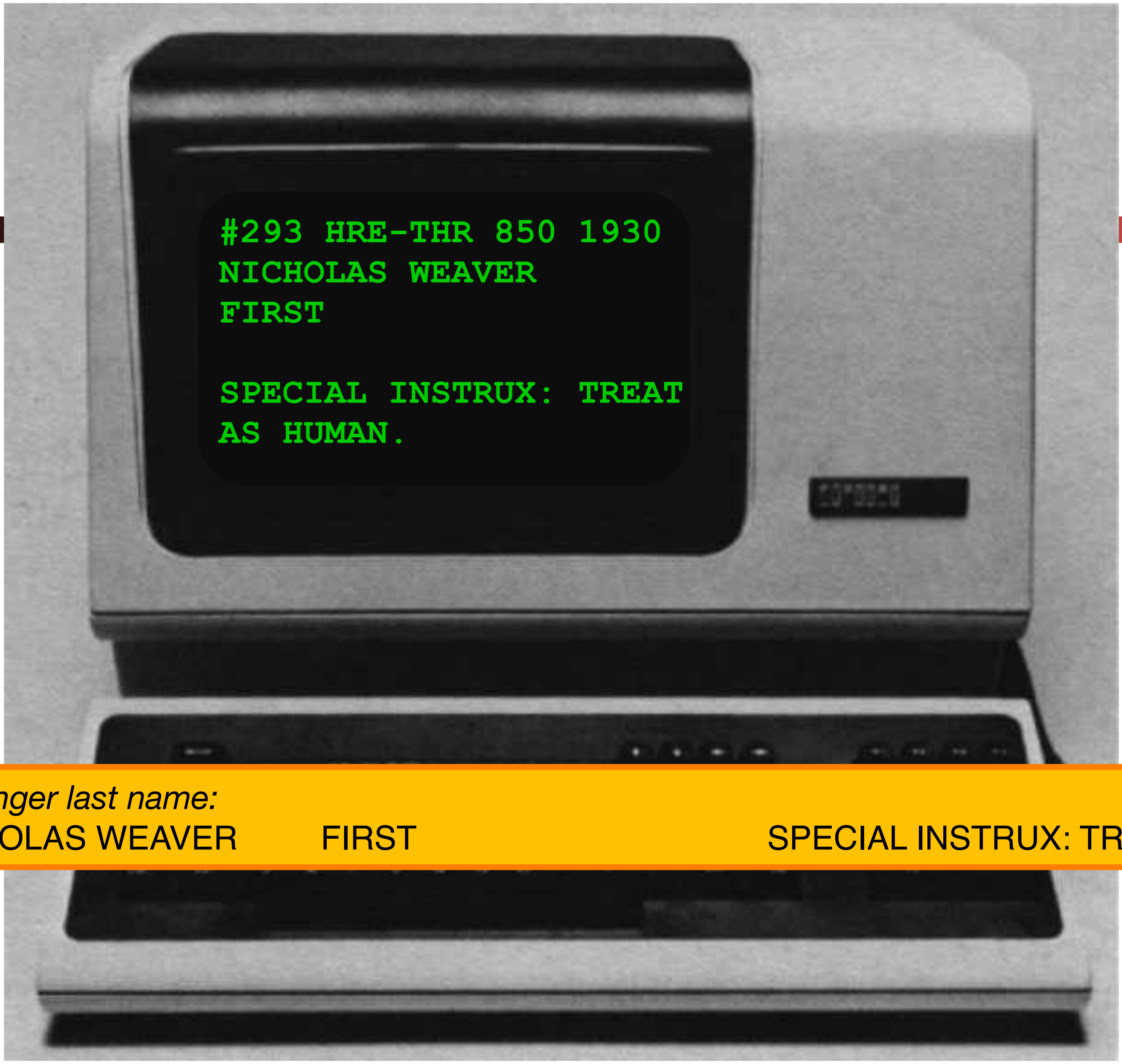
Some younger travelers are not required to present an ID when traveling within the U.S. Learn more

+ Known Traveler Number/Pass ID (optional): ?
+ Redress Number (optional): ?

Seat Request:
● No Preference ○ Aisle ○ Window



```
#293 HRE-THR 850 1930
ALICE SMITH
FIRST

SPECIAL INSTRUX: NONE
```



```
#293 HRE-THR 850 1930
NICHOLAS WEAVER
FIRST

SPECIAL INSTRUX: TREAT
AS HUMAN.
```

Passenger last name:
“NICHOLAS WEAVER   FIRST   SPECIAL INSTRUX: TREAT AS HUMAN.”

```
char name[20];

void vulnerable() {
  ...
  gets(name);
  ...
}
```

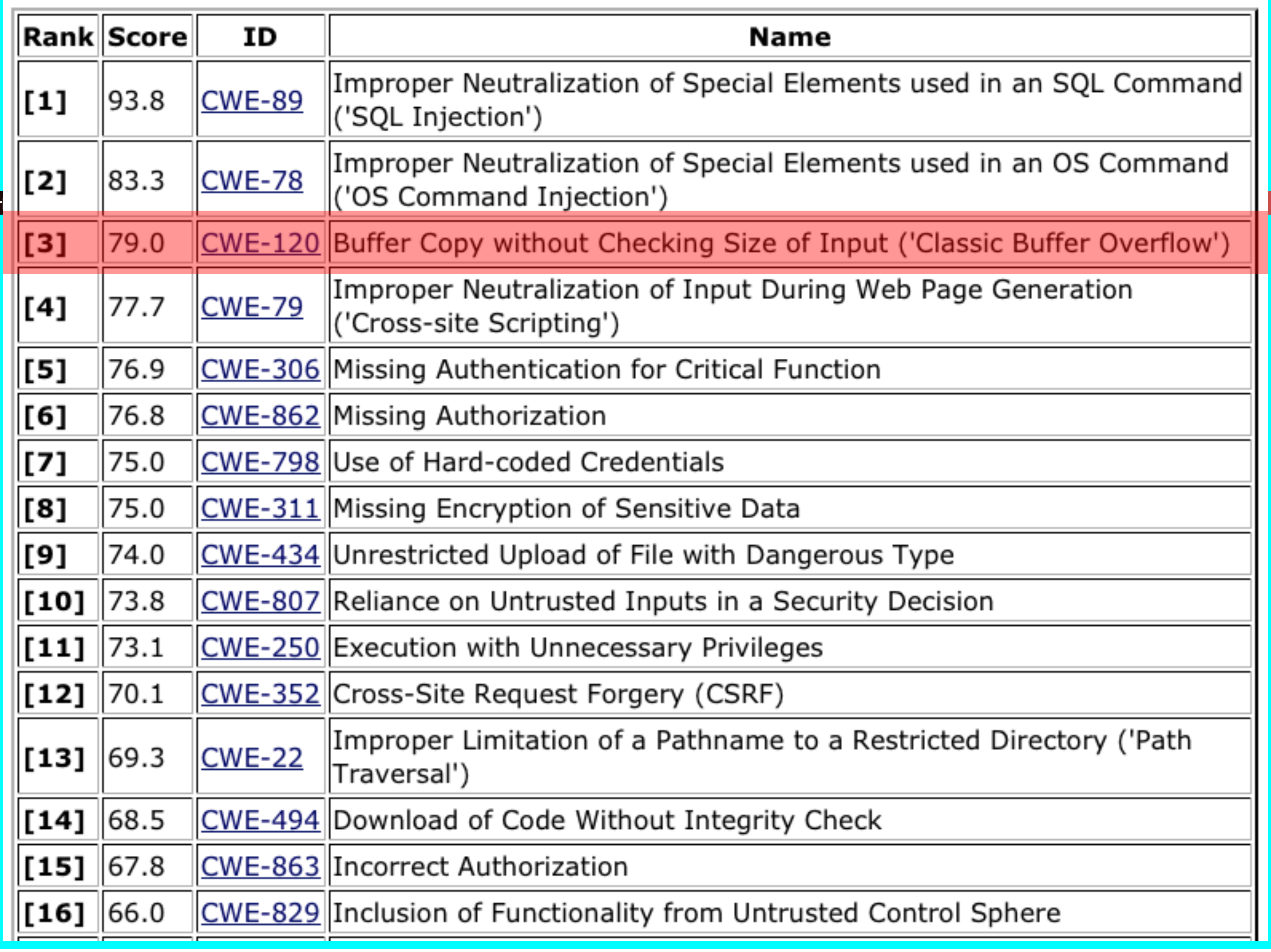
```
char name[20];
char instrux[80] = "none";

void vulnerable() {
   ...
   gets(name);
   ...
}
```

```
char name[20];
int  seatinfirstclass = 0;

void vulnerable() {
   ...
   gets(name);
   ...
}
```

```
char name[20];
int  authenticated = 0;

void vulnerable() {
  ...
  gets(name);
  ...
}
```

```
char line[512];
char command[] = "/usr/bin/finger";

void main() {
  ...
  gets(line);
  ...
  execv(command, ...);
}
```

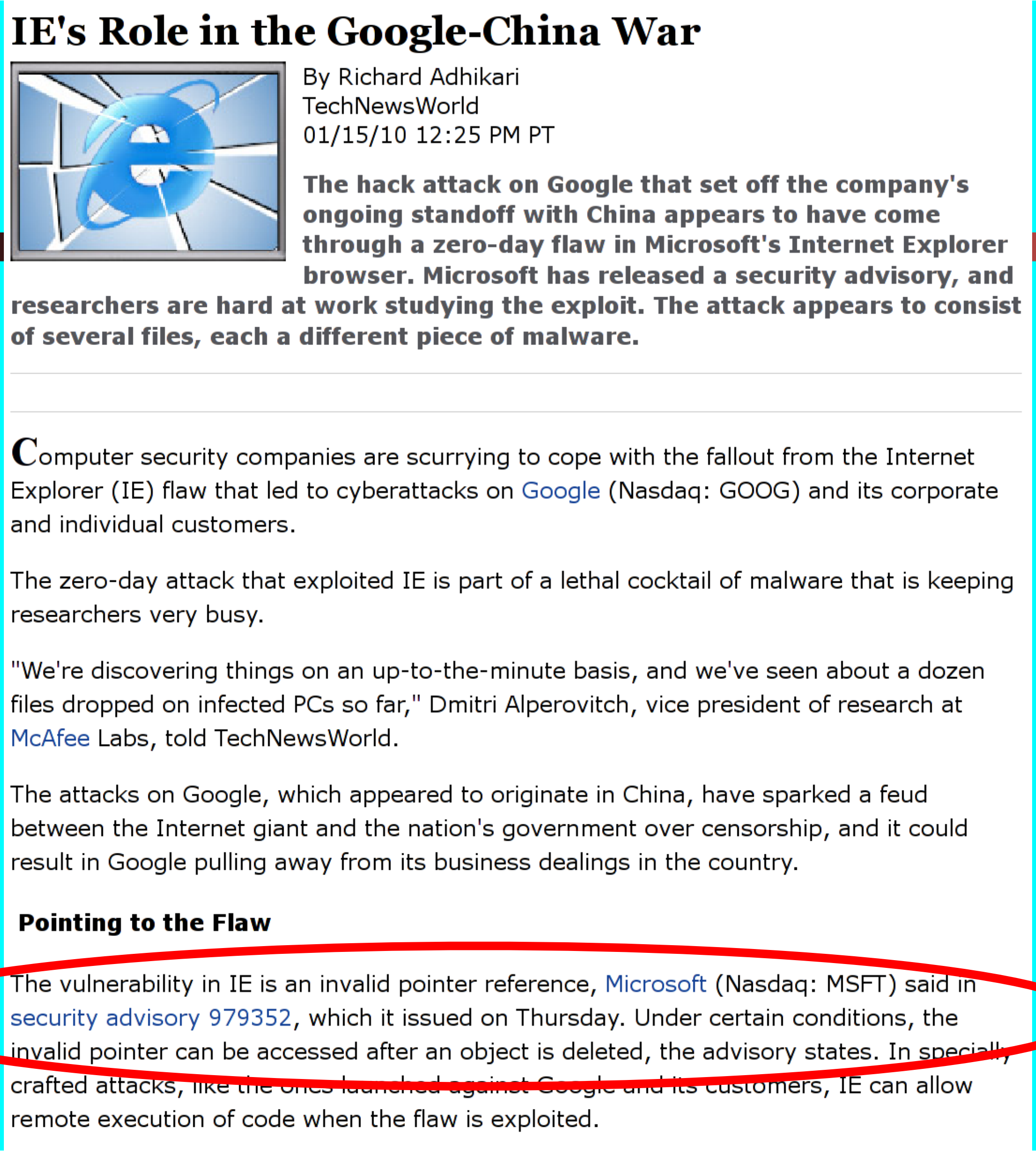
```
char name[20];
int (*fnptr)();

void vulnerable() {
   ...
   gets(name);
   ...
}
```



Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere

```
void vulnerable() {
  char buf[64];
  ...
  gets(buf);
  ...
}
```

```
void still_vulnerable?() {
  char *buf = malloc(64);
  ...
  gets(buf);
  ...
}
```



IE's Role in the Google-China War

By Richard Adhikari
TechNewsWorld
01/15/10 12:25 PM PT

The hack attack on Google that set off the company's ongoing standoff with China appears to have come through a zero-day flaw in Microsoft's Internet Explorer browser. Microsoft has released a security advisory, and researchers are hard at work studying the exploit. The attack appears to consist of several files, each a different piece of malware.

Computer security companies are scurrying to cope with the fallout from the Internet Explorer (IE) flaw that led to cyberattacks on Google (Nasdaq: GOOG) and its corporate and individual customers.

The zero-day attack that exploited IE is part of a lethal cocktail of malware that is keeping researchers very busy.

"We're discovering things on an up-to-the-minute basis, and we've seen about a dozen files dropped on infected PCs so far," Dmitri Alperovitch, vice president of research at McAfee Labs, told TechNewsWorld.

The attacks on Google, which appeared to originate in China, have sparked a feud between the Internet giant and the nation's government over censorship, and it could result in Google pulling away from its business dealings in the country.

Pointing to the Flaw

The vulnerability in IE is an invalid pointer reference, Microsoft (Nasdaq: MSFT) said in security advisory 979352, which it issued on Thursday. Under certain conditions, the invalid pointer can be accessed after an object is deleted, the advisory states. In specially crafted attacks, like the ones launched against Google and its customers, IE can allow remote execution of code when the flaw is exploited.

Disclaimer: x86-32

- For this class, we are going to use 32b x86...
  - Why?
- It is both common and weak...
  - Almost everyone in this class has access to an x86 system:
    Mac, Linux, Windows...
    And can run a 32b x86 virtual machine
  - 64b x86 systems generally include a lot better "mitigations":
    System defenses designed to limit exploitation in this manner
- But these attacks do apply to other microarchitectures
  - Phones are 64b ARM: Can still be exploited in this manner
  - The Internet of Things is mostly 32b or 64b ARM...
    and often neglects to include the mitigations!

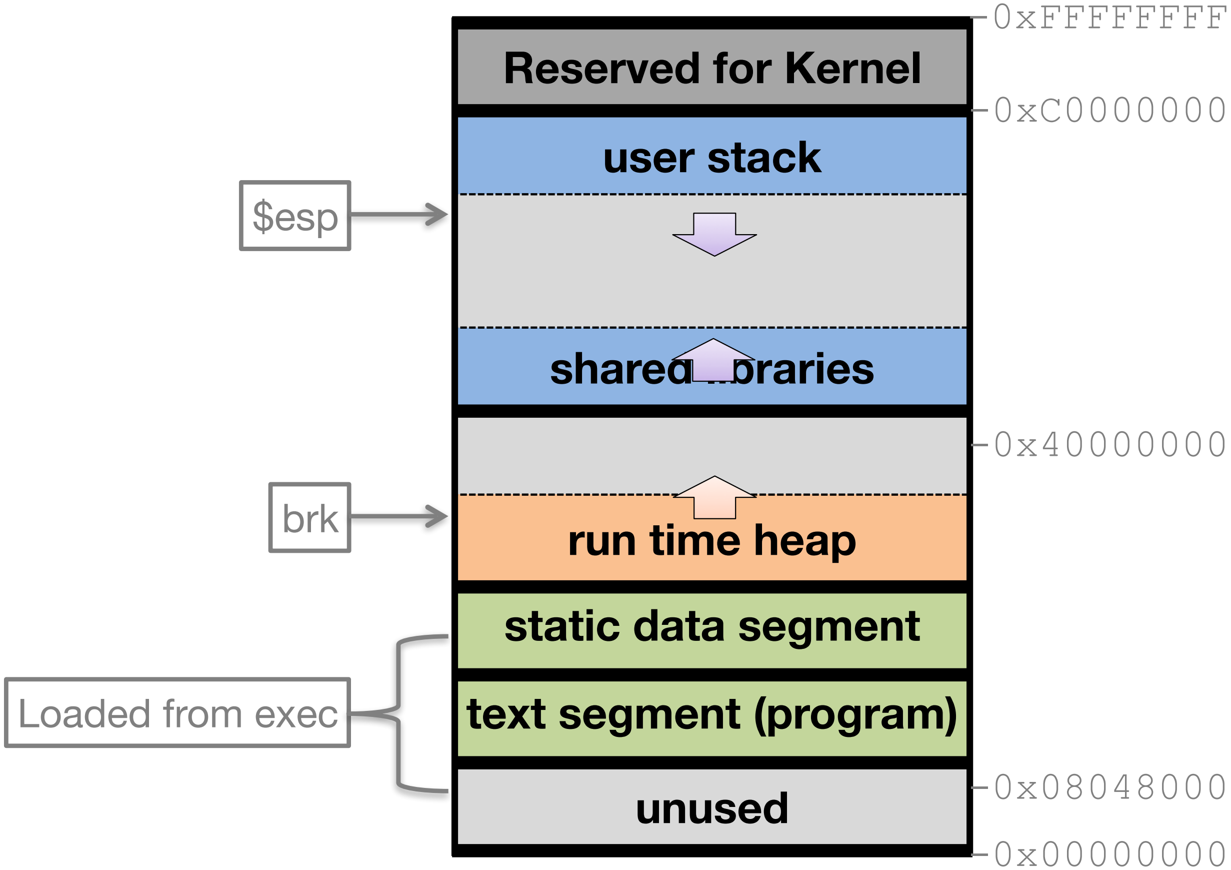
x86 vs RISC-V

- All RISC architectures are the same except for one or two ‘seems like a good idea at the time’ design decisions
  … But x86 is a very different beast from a programing viewpoint
- RISC-V: 32 general purpose registers (well, 31 + x0…)
  - All operations are on data in registers apart from loads & stores
- x86: only a few registers
  - Operations can be directly on data in memory, including a large number relative to the stack
  - EG, add takes two operands, adds them together, and stores the result in the first
    - The first can be a register or memory location
    - The second can be a register, a memory location, or an immediate…
      - But the first and second can’t both be a memory location?!?

The main x86 registers…

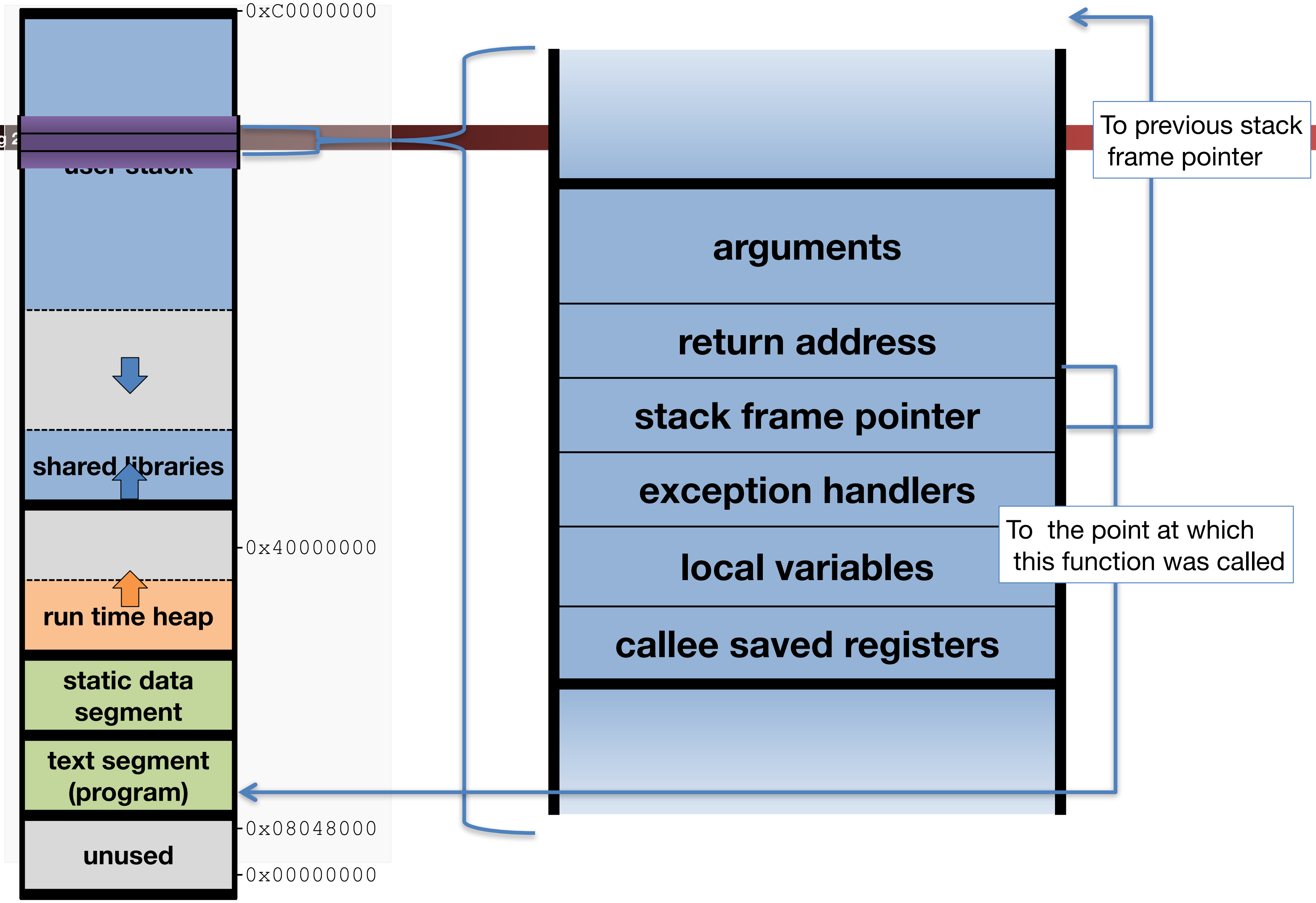
- General purpose: EAX-EDX
  - What you use for computing and other stuff, sorta…
- Indexes & Pointers
  - EBP: “Frame pointer”: points to the top/start of the current call frame on the stack
  - ESP: “Stack pointer”: points to the current stack
    (Remember, stack grows down!)
    - PUSH and POP
      - Decrement the stack pointer and store something there
      - Load something and increment the stack pointer
    - Most operations are done with data on the stack…

Linux (32-bit) process memory layout



x86 function calling

- Place the arguments on the stack
    - Compare with RISC-V where the first arguments are in registers
- CALL the function
    - Which pushes the return address onto the stack (RIP == Return Instruction Pointer)
- do your stuff…
    - Start by saving the old EP on the stack (SFP == Saved Frame Pointer)
- Restore everything
    - Reload EBP, pop ESP as necessary
- RET
    - Which jumps to the return address that is currently pointed to by ESP
    - And can optionally pop the stack a lot further…



0xC0000000

user stack

shared libraries

0x40000000

run time heap

static data segment

text segment (program)

0x08048000

unused

0x00000000

arguments

return address

stack frame pointer

exception handlers

local variables

callee saved registers

To previous stack frame pointer

To the point at which this function was called

```
void safe() {
  char buf[64];
  ...
  fgets(buf, 64, stdin);
  ...
}
```

```
void safer() {
  char buf[64];
  ...
  fgets(buf,sizeof(buf),stdin);
  ...
}
```

Assume these are both under the control of an attacker.

```
void vulnerable(int len, char *data) {
  char buf[64];
  if (len > 64)
    return;
  memcpy(buf, data, len);
}
```

```
memcpy(void *s1, const void *s2, size_t n);
```

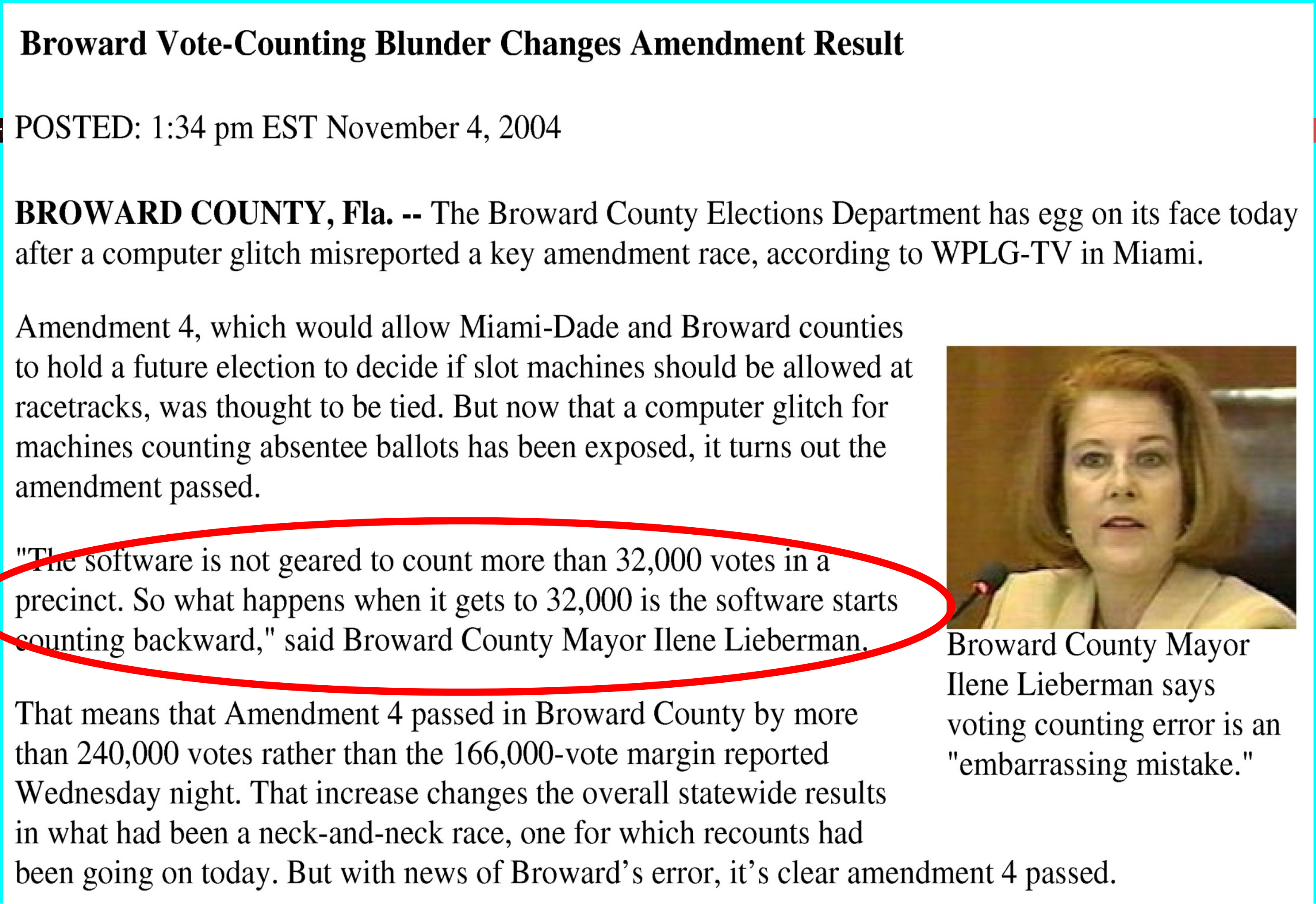
size_t is ***unsigned***:
What happens if len == -1?

```
void safe(size_t len, char *data) {
  char buf[64];
  if (len > 64)
    return;
  memcpy(buf, data, len);
}
```

```
void f(size_t len, char *data) {
  char *buf = malloc(len+2);
  if (buf == NULL) return;
  memcpy(buf, data, len);
  buf[len] = '\n';
  buf[len+1] = '\0';
}
```

Is it safe? Spam the Chat!

Vulnerable!
If **`len`** = **`0xffffffff`**, *allocates only 1 byte*



Broward Vote-Counting Blunder Changes Amendment Result

POSTED: 1:34 pm EST November 4, 2004

BROWARD COUNTY, Fla. -- The Broward County Elections Department has egg on its face today after a computer glitch misreported a key amendment race, according to WPLG-TV in Miami.

Amendment 4, which would allow Miami-Dade and Broward counties to hold a future election to decide if slot machines should be allowed at racetracks, was thought to be tied. But now that a computer glitch for machines counting absentee ballots has been exposed, it turns out the amendment passed.

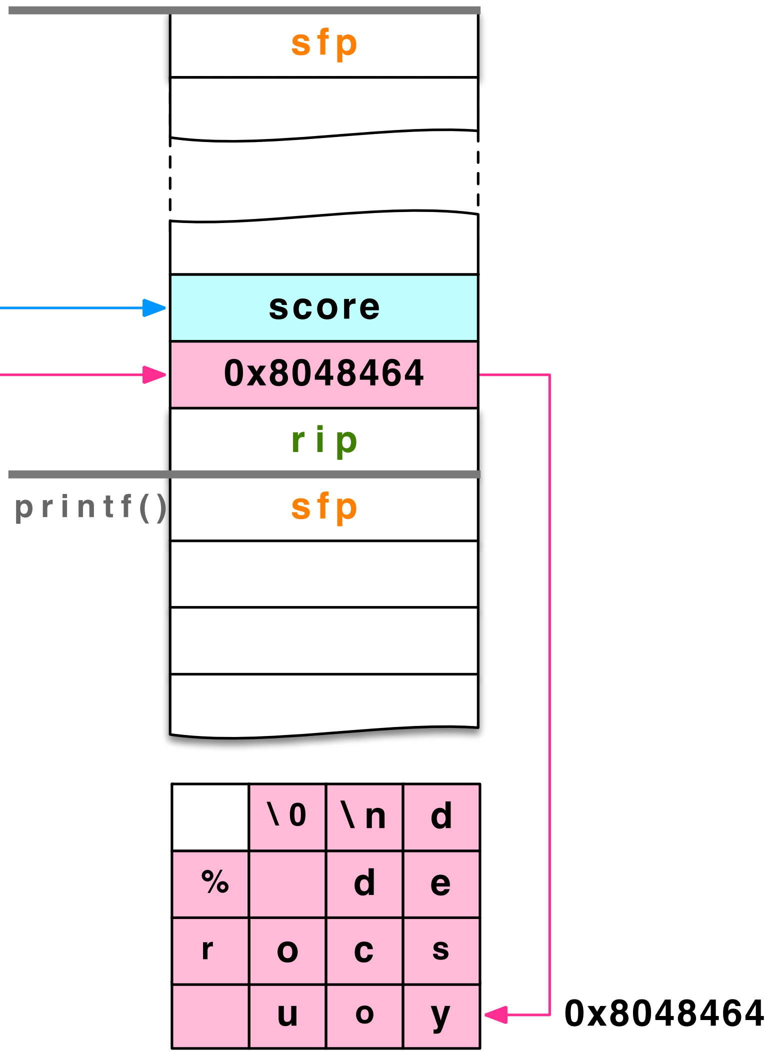
"The software is not geared to count more than 32,000 votes in a precinct. So what happens when it gets to 32,000 is the software starts counting backward," said Broward County Mayor Ilene Lieberman.

Broward County Mayor Ilene Lieberman says voting counting error is an "embarrassing mistake."

That means that Amendment 4 passed in Broward County by more than 240,000 votes rather than the 166,000-vote margin reported Wednesday night. That increase changes the overall statewide results in what had been a neck-and-neck race, one for which recounts had been going on today. But with news of Broward's error, it's clear amendment 4 passed.

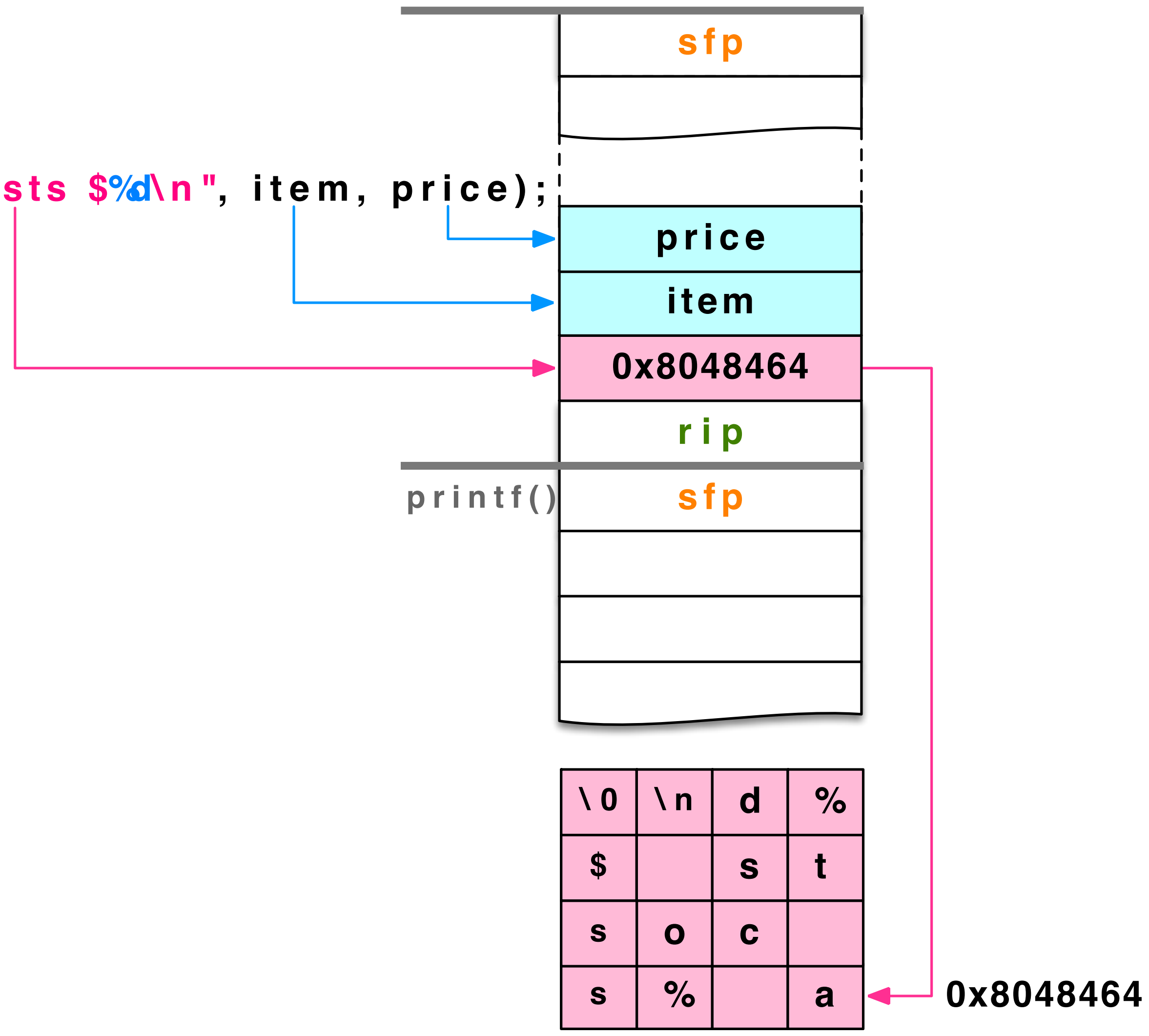
```
void vulnerable() {
  char buf[64];
  if (fgets(buf, 64, stdin) == NULL)
    return;
  printf(buf);
}
```

```
printf("you scored %d\n", score);
printf(“you scored %d\n”, score);
```



	\0	\n	d
%		d	e
r	o	c	s
	u	o	y

```
printf("a %s costs $%d\n", item, price);
printf("a %s costs $%d\n", item, price);
```

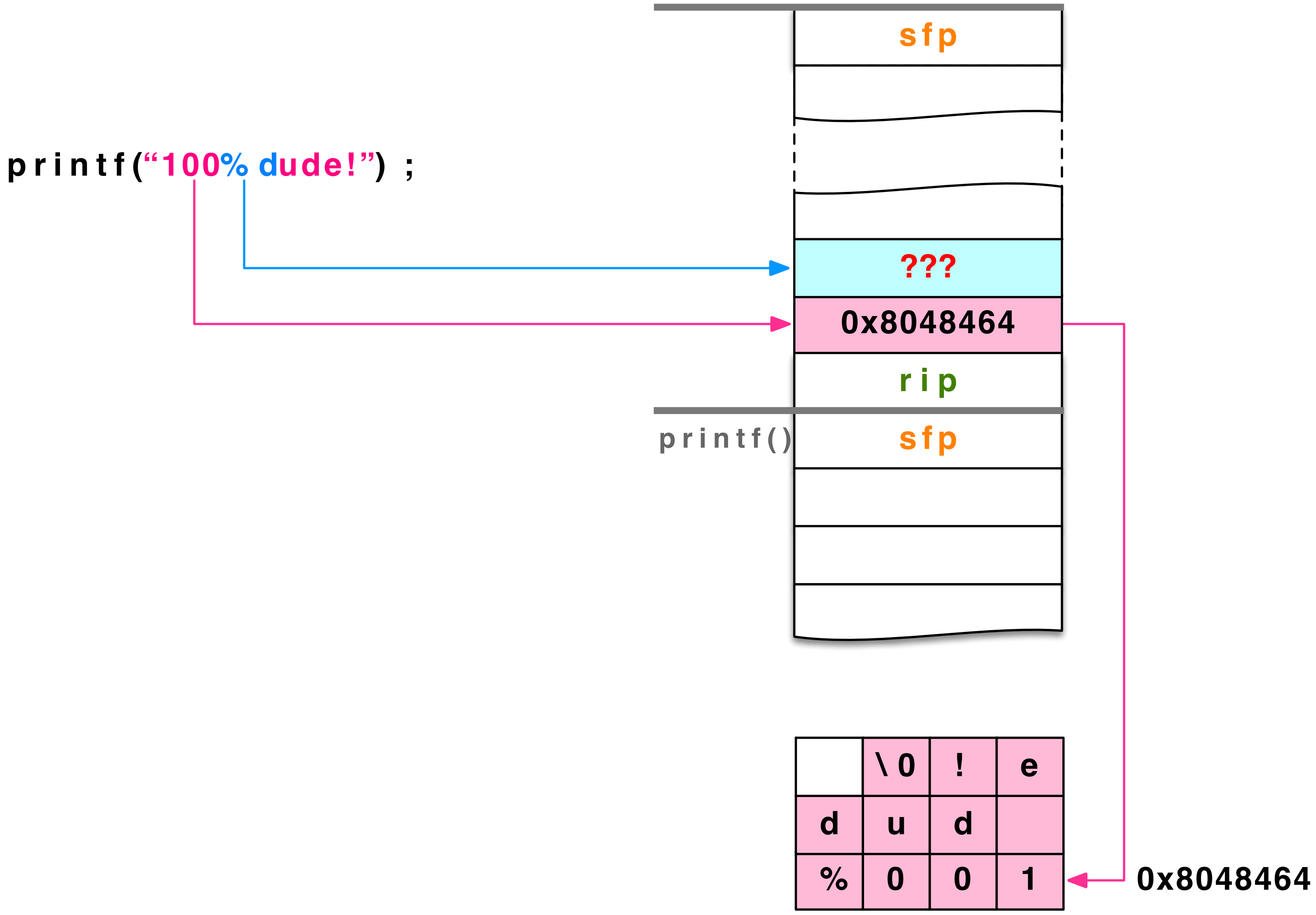


\0	\n	d	%
$		s	t
s	o	c	
s	%		a

Fun With **`printf`** format strings...

```
printf("100% dude!");
```

Format argument is missing!



```
printf("100% dude!") ;
```

	\0	!	e
d	u	d	
%	0	0	1

More Fun With `printf` format strings...

```
printf("100% dude!");
```

⇒ ***prints value 4 bytes above retaddr as integer***

```
printf("100% sir!");
```

⇒ ***prints bytes <u>pointed to</u> by that stack entry up through first NUL***

```
printf("%d %d %d %d ...");
```

⇒ ***prints series of stack entries as integers***

```
printf("%d %s");
```

⇒ ***prints value 4 bytes above retaddr plus bytes*** *pointed to by <u>preceding</u> stack entry*

```
printf("100% nuke'm!");
```

What does the %n format do??

%n *writes* the number of characters printed so far into the corresponding format argument.

```
int report_cost(int item_num, int price) {
  int colon_offset;
  printf("item %d:%n $%d\n", item_num,
                 &colon_offset, price);
  return colon_offset;
}
```

`report_cost(3, 22)` **prints** `"item 3: $22"`
and returns the value 7

`report_cost(987, 5)` **prints** `"item 987: $5"`
and returns the value 9

Fun With `printf` format strings...

```
printf("100% dude!");
```

⇒ ***prints value 4 bytes above retaddr as integer***

```
printf("100% sir!");
```

⇒ ***prints bytes pointed to by that stack entry up through first NUL***

```
printf("%d %d %d %d ...");
```

⇒ ***prints series of stack entries as integers***

```
printf("%d %s");
```

⇒ ***prints value 4 bytes above retaddr plus bytes*** *pointed to by preceding stack entry*

```
printf("100% nuke’m!");
```

⇒ ***writes the value 3 to the address pointed to by stack entry***

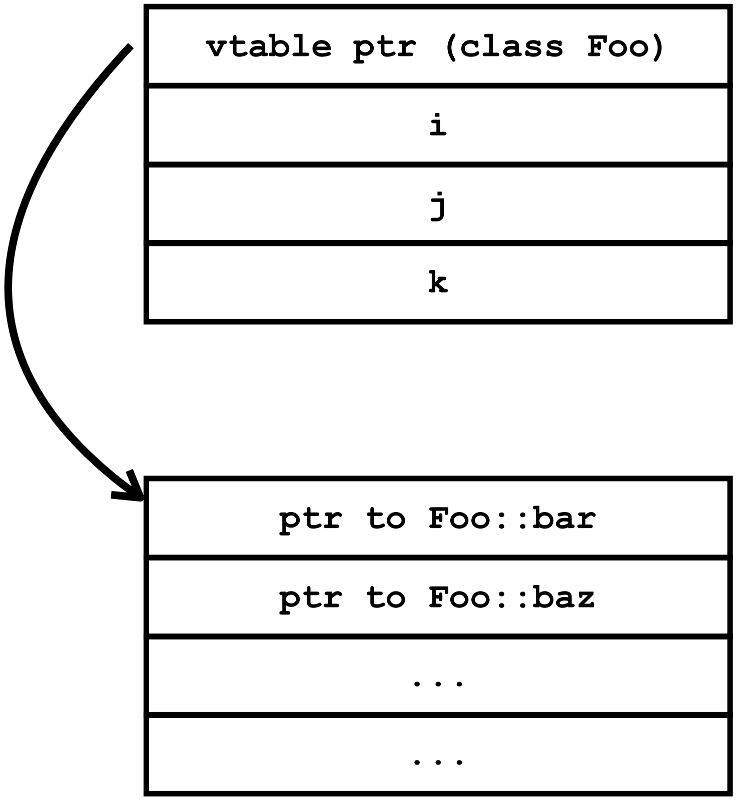
```
void safe() {
  char buf[64];
  if (fgets(buf, 64, stdin) == NULL)
    return;
  printf("%s", buf);
}
```

It isn't just the stack...

- Control flow attacks require that the attacker overwrite a piece of memory that contains a pointer for future code execution
  - The return address on the stack is just the easiest target
- You can cause plenty of mayhem overwriting memory in the heap...
  - And it is made easier when targeting C++
- Allows alternate ways to hijack control flow of the program

Compiler Operation:
Compiling Object Oriented Code

```
class Foo {
   int i, j, k;
   public virtual void bar(){ ... }
   public virtual void baz(){ ... }
....
```



vtable ptr (class Foo)

i

j

k

ptr to Foo::bar

ptr to Foo::baz

...

...

So Targets For Overwriting...

- If you can overwrite a vtable pointer…
  - It is effectively the same as overwriting the return address pointer on the stack:
    When the function gets invoked the control flow is hijacked to point to the attacker's code
    - The only difference is that instead of overwriting with a pointer you overwrite it with a pointer to a table of pointers...
- Heap Overflow:
  - A buffer in the heap is not checked:
    Attacker writes beyond and overwrites the vtable pointer of the next object in memory
- Use-after-free:
  - An object is deallocated too early:
    Attacker writes new data in a newly reallocated block that overwrites the vtable pointer
  - Object is then invoked

Magic Numbers & Exploitation…

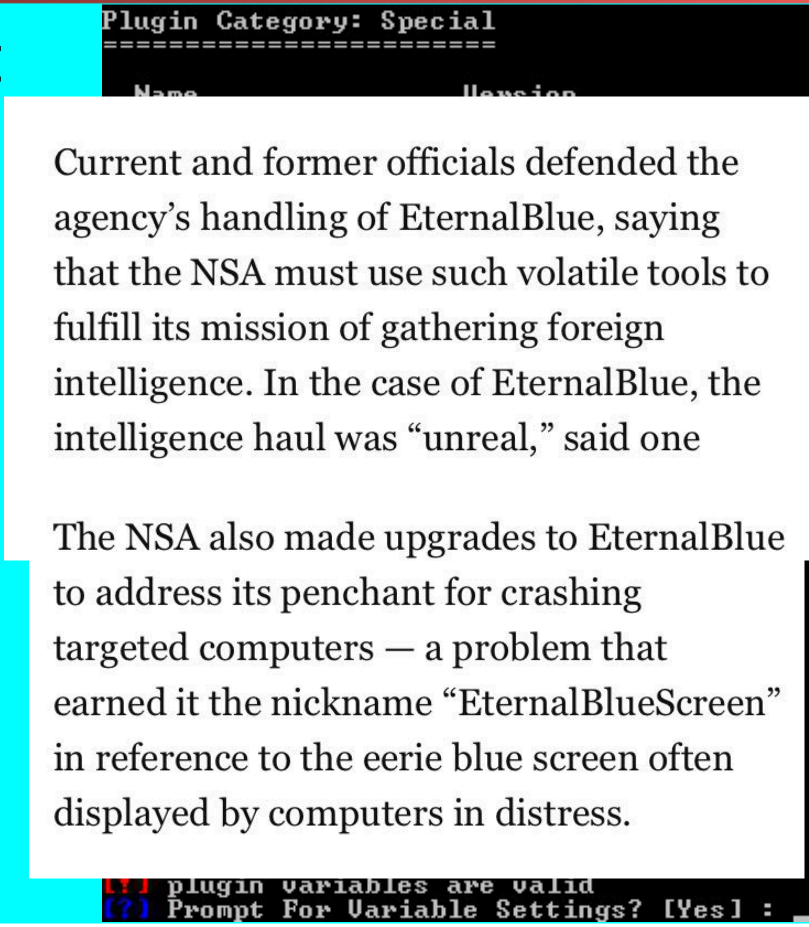
- Exploits can often be ***very*** brittle
  - You see this on your Project 1: Your ./egg will not work on someone else's VM because the memory layout is different
- Making an exploit robust is an art unto itself: e.g. EXTRABACON…
- EXTRABACON is an NSA exploit for Cisco ASA "Adaptive Security Appliances"
  - It had an exploitable stack-overflow vulnerability in the SNMP read operation
  - But actual exploitation required two steps:
    Query for the particular version (with an SMTP read)
    Select the proper set of magic numbers for that version

A hack that helps:
NOOP sled...

- Don't just overwrite the pointer and then provide the code you want to execute...
- Instead, write a large number of NOOP operations
  - Instructions that do nothing
- Now if you are a ***little*** off, it doesn't matter
  - Since if you are close enough, control flow will land in the sled and start running...

ETERNALBLUE(screen)

- ETERNALBLUE is another NSA exploit
  - Stolen by the same group ("ShadowBrokers") which stole EXTRABACON
- Eventually it was very robust...
  - This was "god mode":
    remote exploit Windows through SMBv1 (Windows File sharing)
- But initially it was jokingly called ETERNALBLUESCREEN
  - Because it would crash Windows computers more reliably than exploitation.



```
Plugin Category: Special
========================

  Name                    Version
```

Current and former officials defended the agency's handling of EternalBlue, saying that the NSA must use such volatile tools to fulfill its mission of gathering foreign intelligence. In the case of EternalBlue, the intelligence haul was "unreal," said one

The NSA also made upgrades to EternalBlue to address its penchant for crashing targeted computers — a problem that earned it the nickname "EternalBlueScreen" in reference to the eerie blue screen often displayed by computers in distress.

```
[!] plugin variables are valid
[?] Prompt For Variable Settings? [Yes] :
```

And Now A More Detailed Example...

- Walking through a function call in detail...
  - Slides from Matthias Vallentin