Overflows, Injection, and Memory Safety

CS 161: Computer Security

Prof. Vern Paxson

TAs: Paul Bramsen, Apoorva Dornadula,
David Fifield, Mia Gil Epner, David Hahn, Warren He,
Grant Ho, Frank Li, Nathan Malkin, Mitar Milutinovic,
Rishabh Poddar, Rebecca Portnoff, Nate Wang

http://inst.eecs.berkeley.edu/~cs161/

January 24, 2017

Common Assumptions When Discussing Attacks

- (Note, these tend to be pessimistic … but prudent)
- Attackers can interact with our systems without particular notice
  - *Probing* (poking at systems) may go unnoticed …
  - … even if highly repetitive, leading to crashes, and *easy to detect*
- It’s easy for attackers to know general information about their targets
  - OS types, software versions, usernames, server ports, IP addresses, usual patterns of activity, administrative procedures

Common Assumptions, con’t

- Attackers can obtain access to a copy of a given system to measure and/or determine how it works
- Attackers can make energetic use of automation
  - They can often find clever ways to automate
- Attackers can pull off complicated coordination across a bunch of different elements/systems
- Attackers can bring large resources to bear if req’d
  - Computation, network capacity
  - But they are *not* super-powerful (e.g., control entire ISPs)

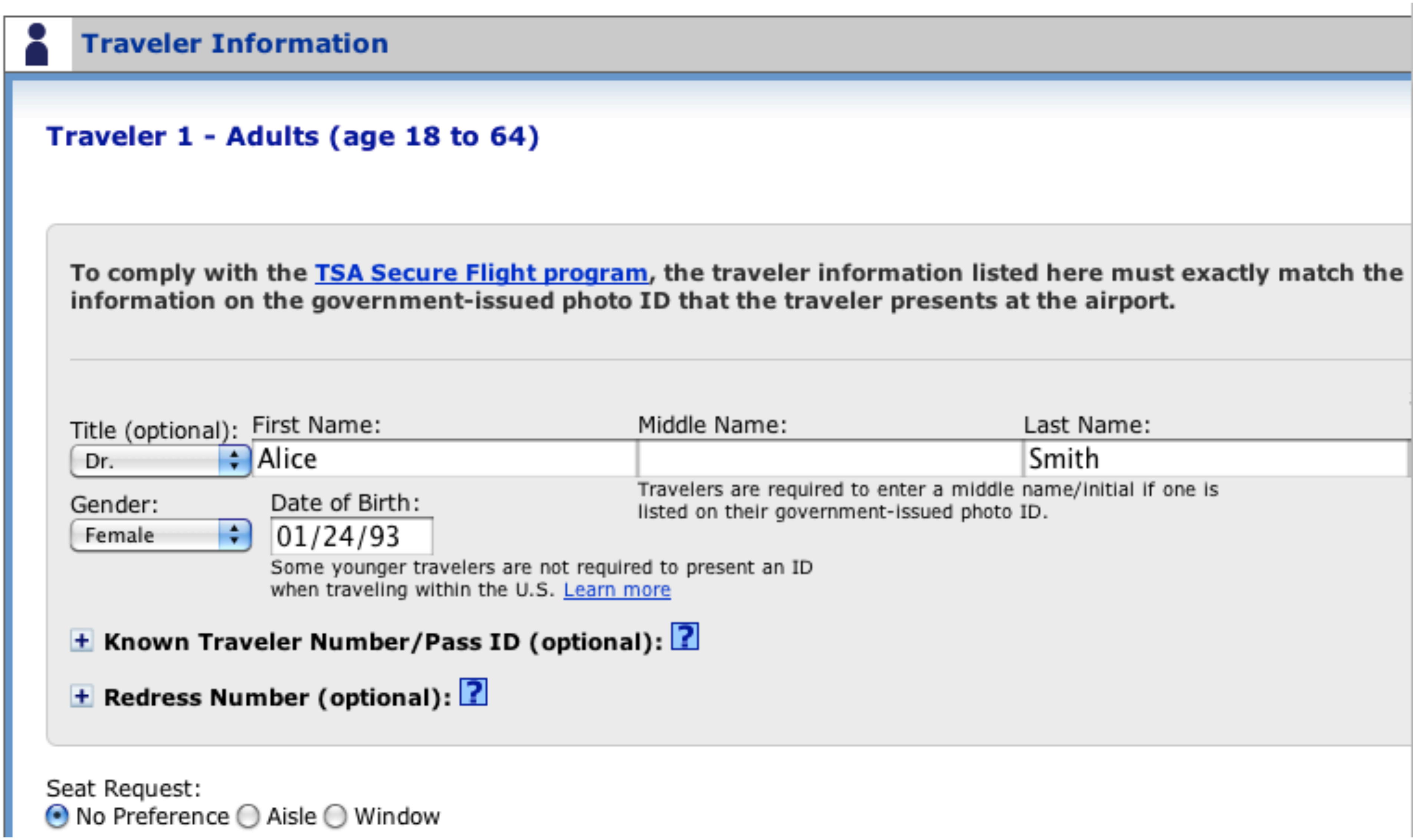
Common Assumptions, con't

- If it helps the attacker in some way, assume they can obtain privileges
  - But if the privilege gives everything away (attack becomes trivial), then we care about unprivileged attacks
- The ability to robustly *detect* that an attack has occurred does not replace desirability of preventing
- Infrastructure machines/systems are well protected (hard to directly take over)
  - So a vulnerability that requires infrastructure compromise is less worrisome than same vulnerability that doesn't

Common Assumptions, con’t

- Network routing is hard to alter … other than with physical access near clients (e.g., “coffeeshop”)
  - Such access helps fool clients to send to wrong place
  - Can enable *Man-in-the-Middle* (MITM) attacks
- We worry about attackers who are lucky
  - Since often automation/repetition can help “make luck”
- Just because a system does not have apparent value, it may still be a target
- Any others?

Thinking about overflows



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):	First Name:	Middle Name:	Last Name:
Dr.	Alice		Smith

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

Gender: Female

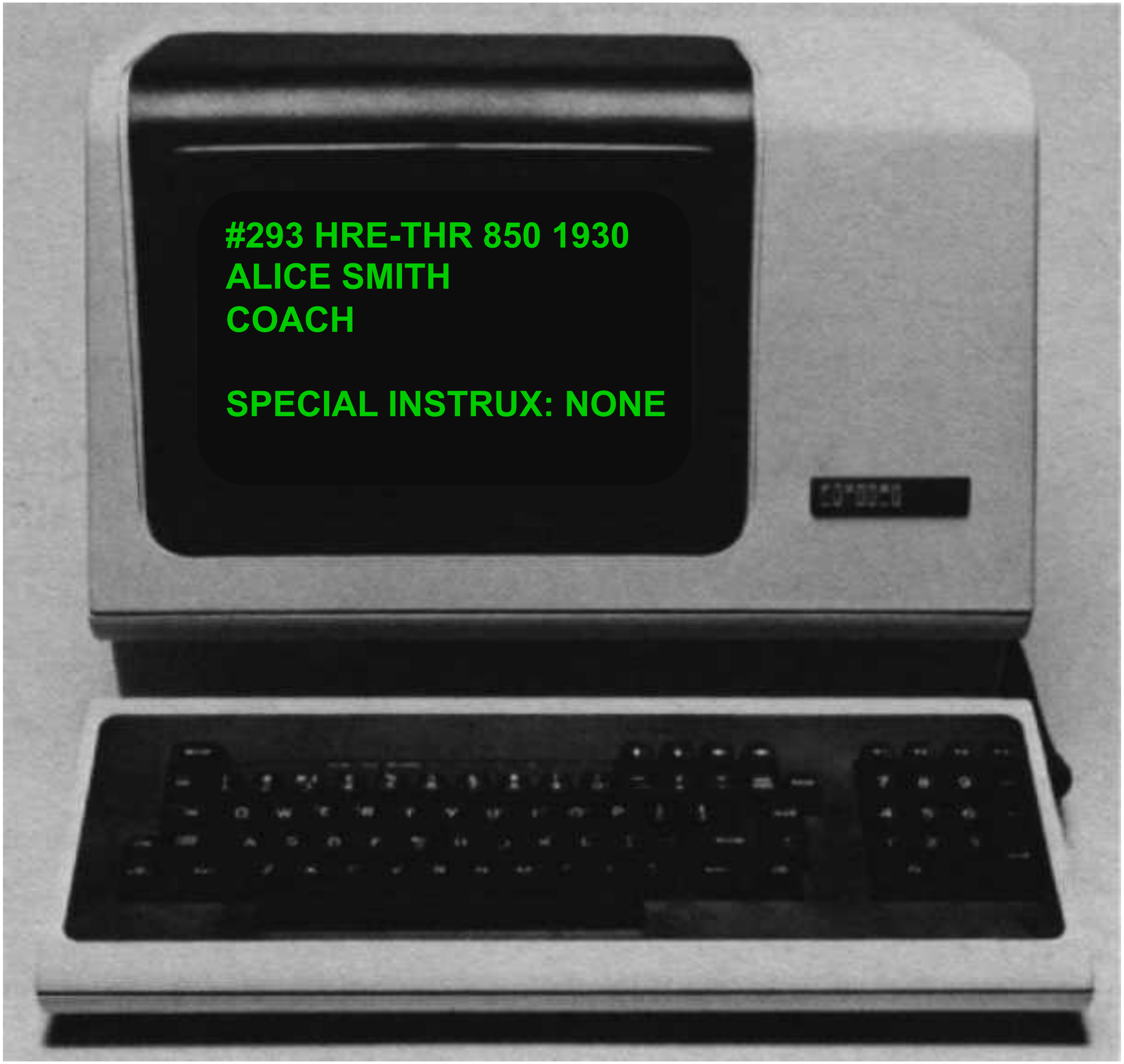
Date of Birth: 01/24/93

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

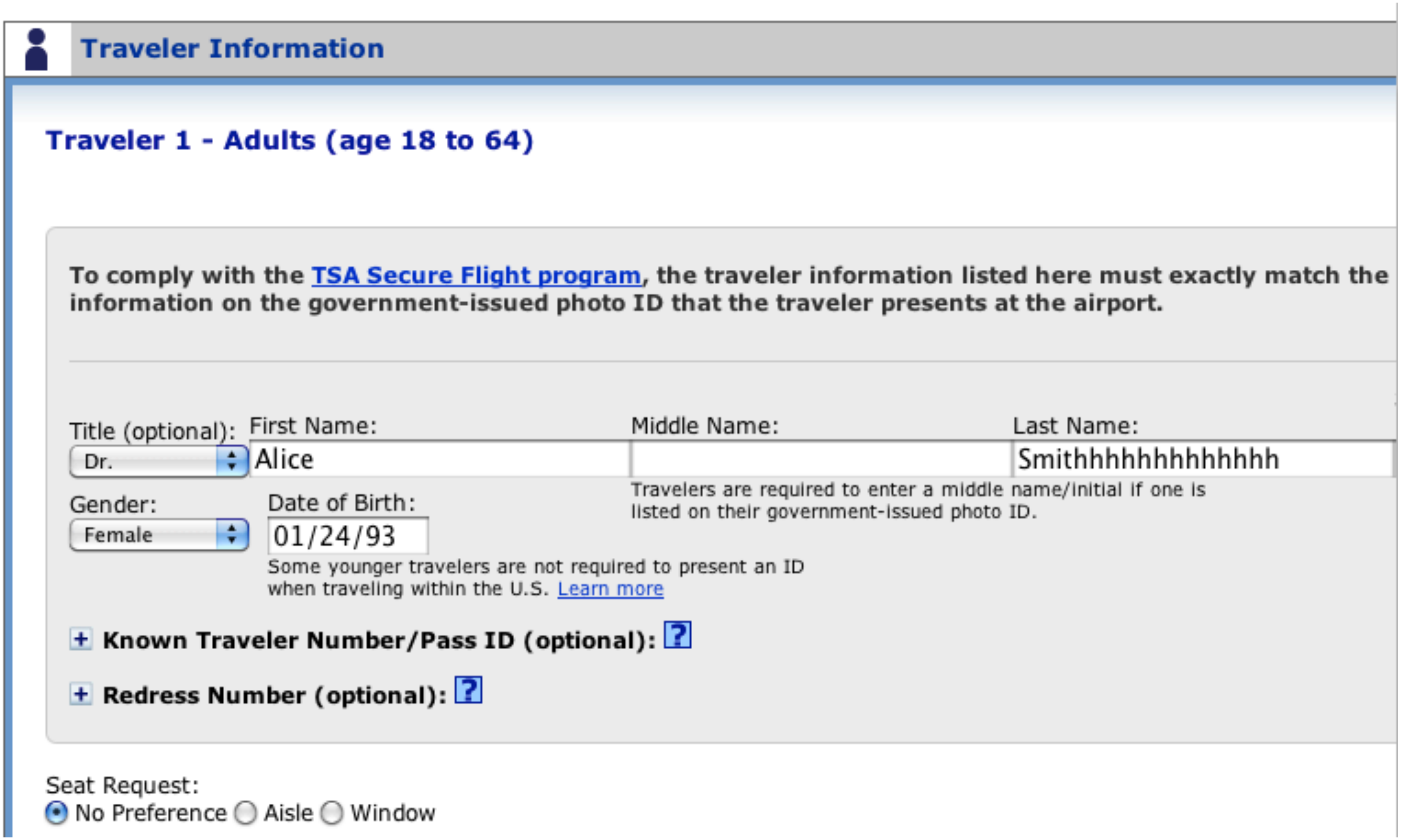
+ **Known Traveler Number/Pass ID (optional):** ?

+ **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: Smithhhhhhhhhhhhh

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

Gender: Female

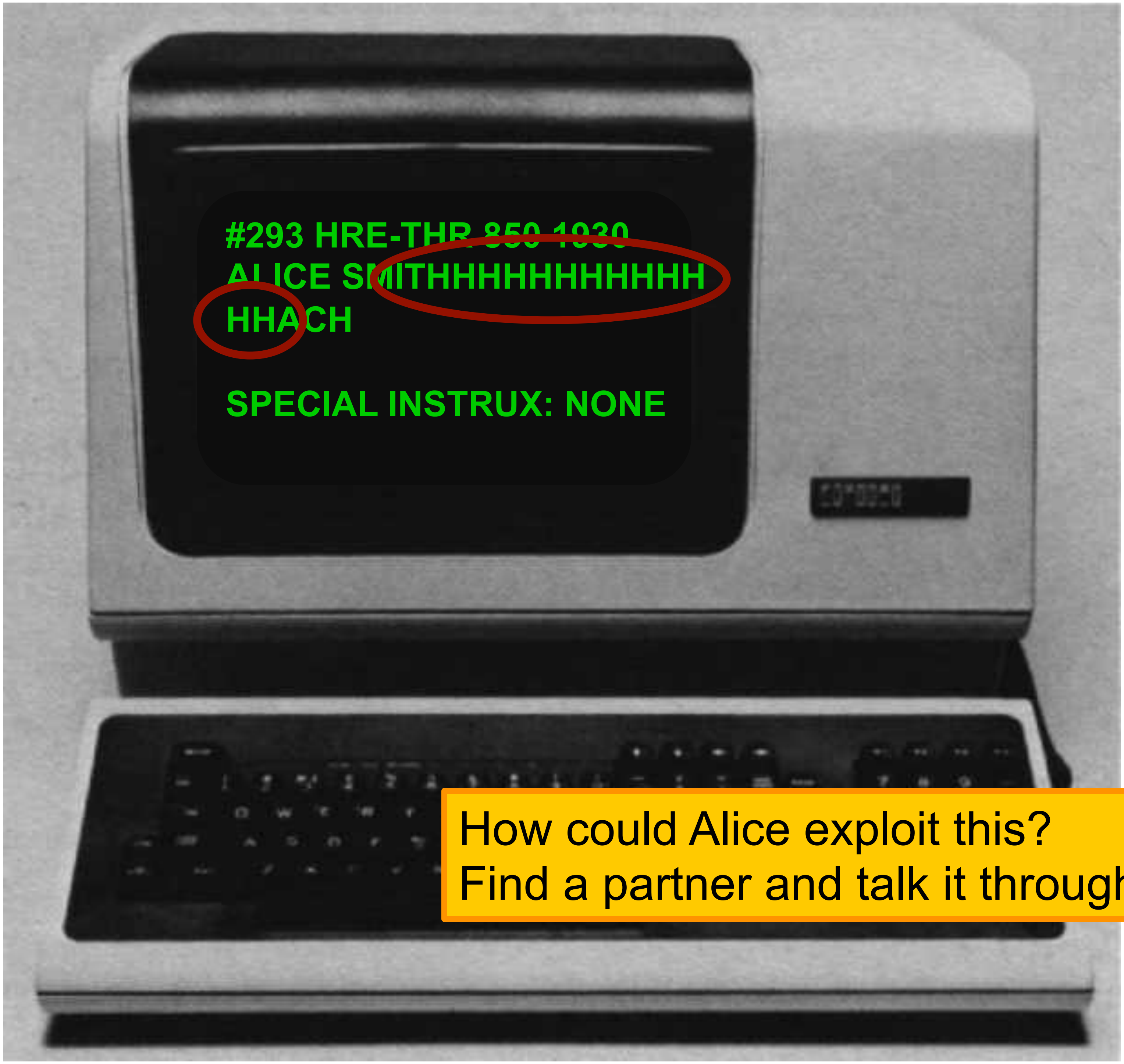
Date of Birth: 01/24/93

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

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

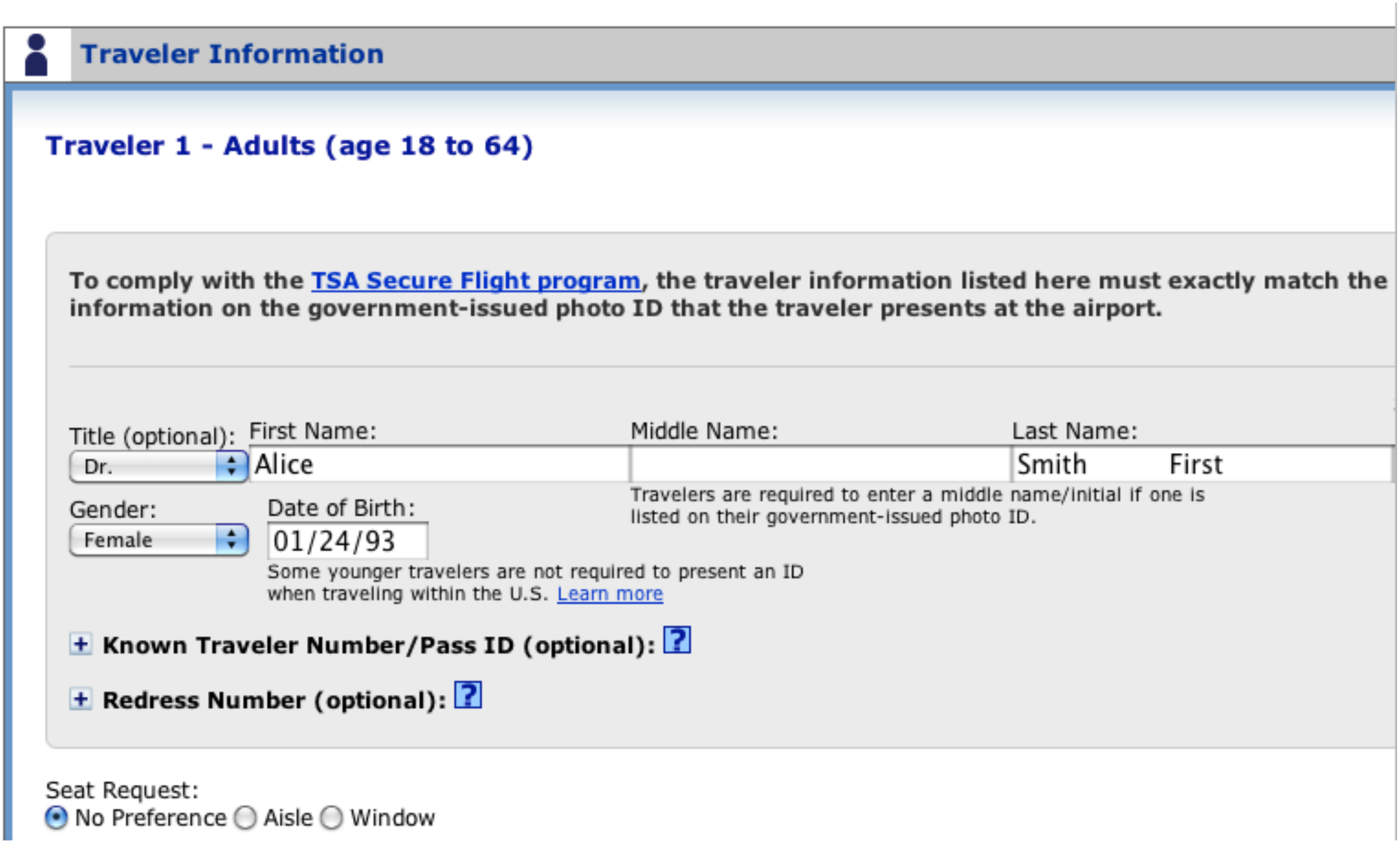
+ Redress Number (optional): ?

Seat Request:
● No Preference ○ Aisle ○ Window



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

SPECIAL INSTRUX: NONE
```

How could Alice exploit this?
Find a partner and talk it through.



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):	First Name:	Middle Name:	Last Name:
Dr.	Alice		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

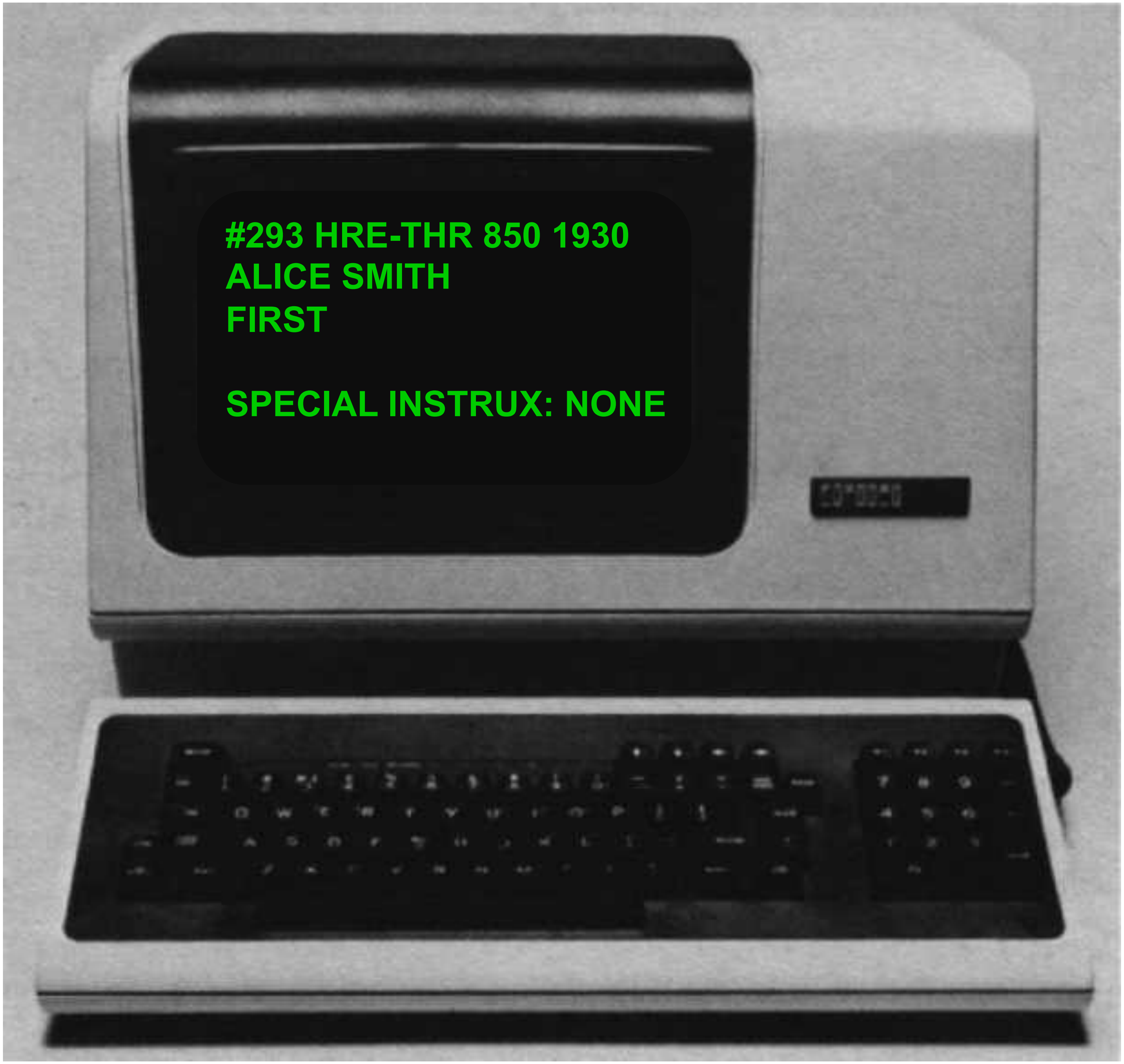
Some younger travelers are not required to present an ID when traveling within the U.S. Learn more

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

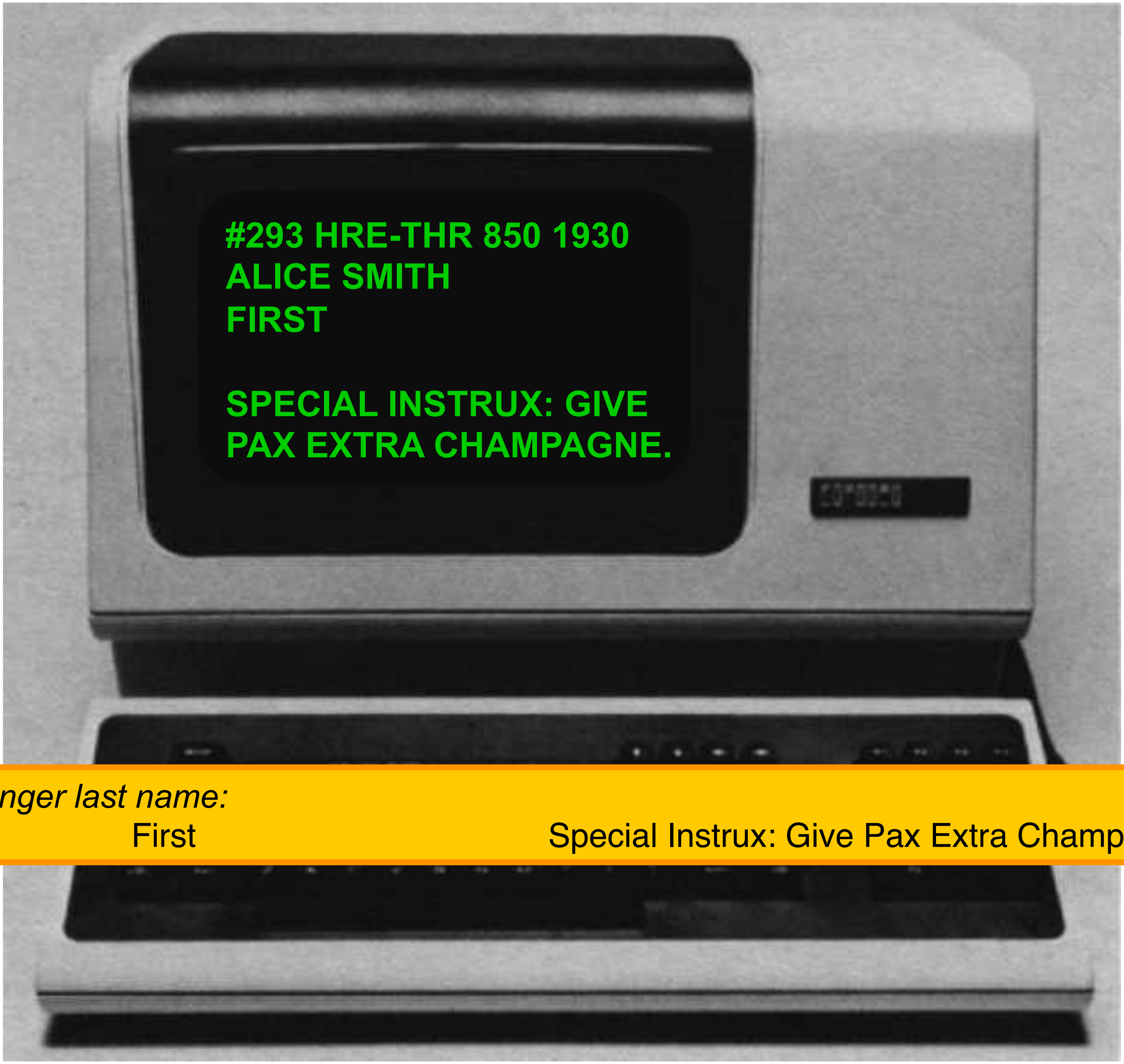
+ Redress Number (optional): ?

Seat Request:

● No Preference ○ Aisle ○ Window



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

SPECIAL INSTRUX: NONE
```



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

SPECIAL INSTRUX: GIVE
PAX EXTRA CHAMPAGNE.
```

Passenger last name:
“Smith First Special Instrux: Give Pax Extra Champagne.”

```
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);
  ...
}
```

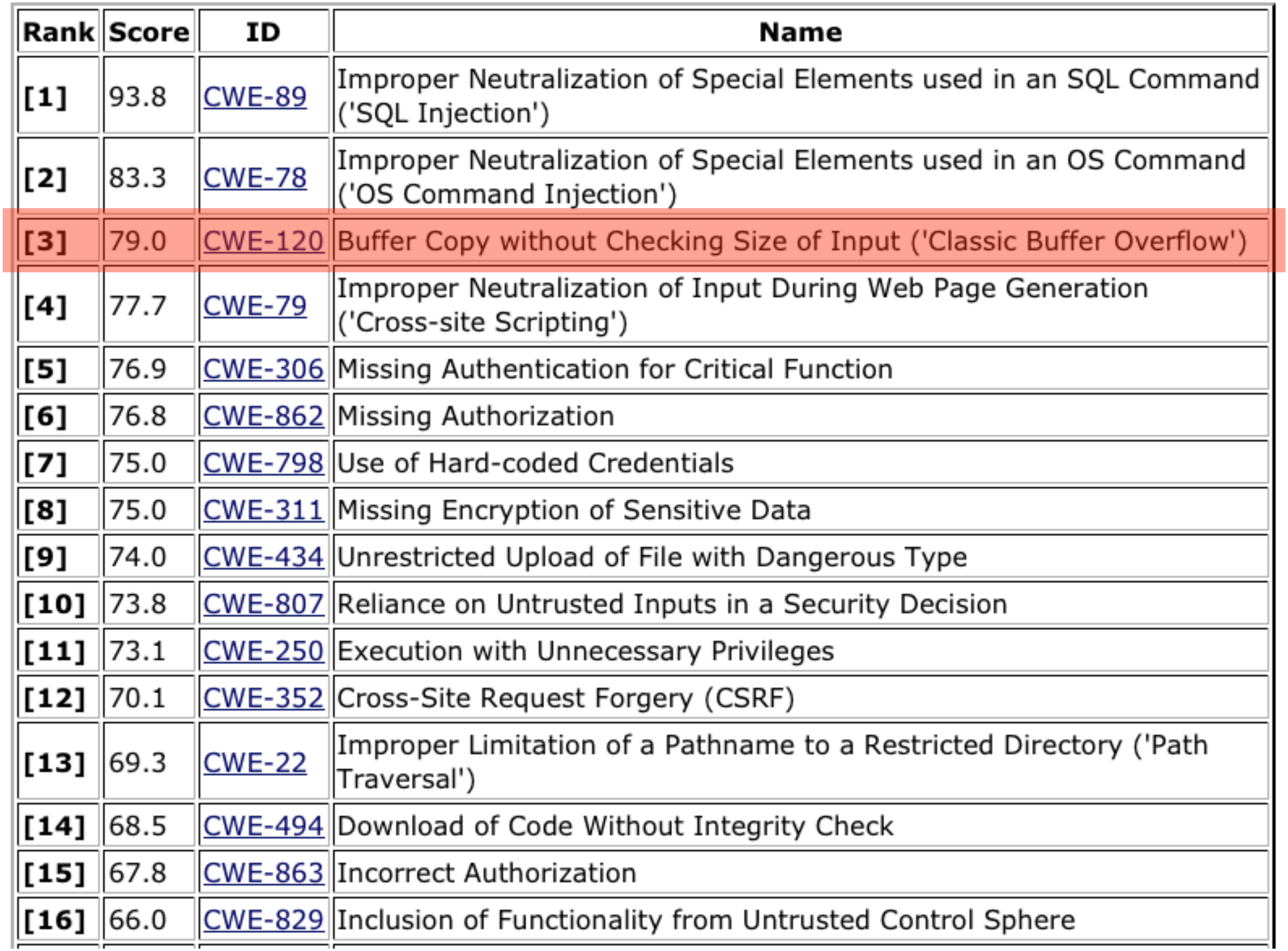
```
char name[20];
int  authenticated = 0;

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);
  ...
}
```

Walking Through Overflow Vulnerabilities

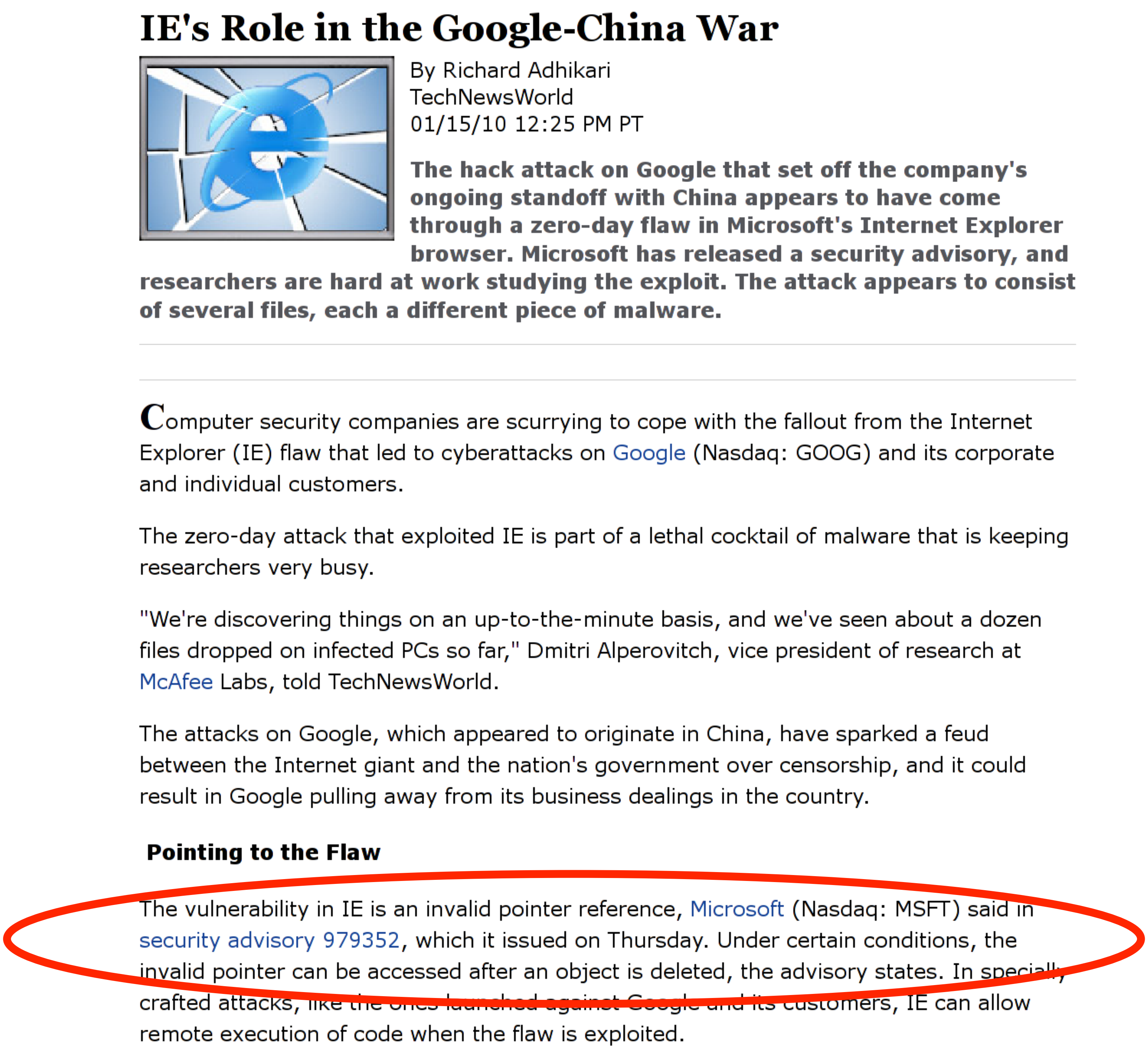
(See separate slides)



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.

```
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);
```

```c
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? Talk to your partner.

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.

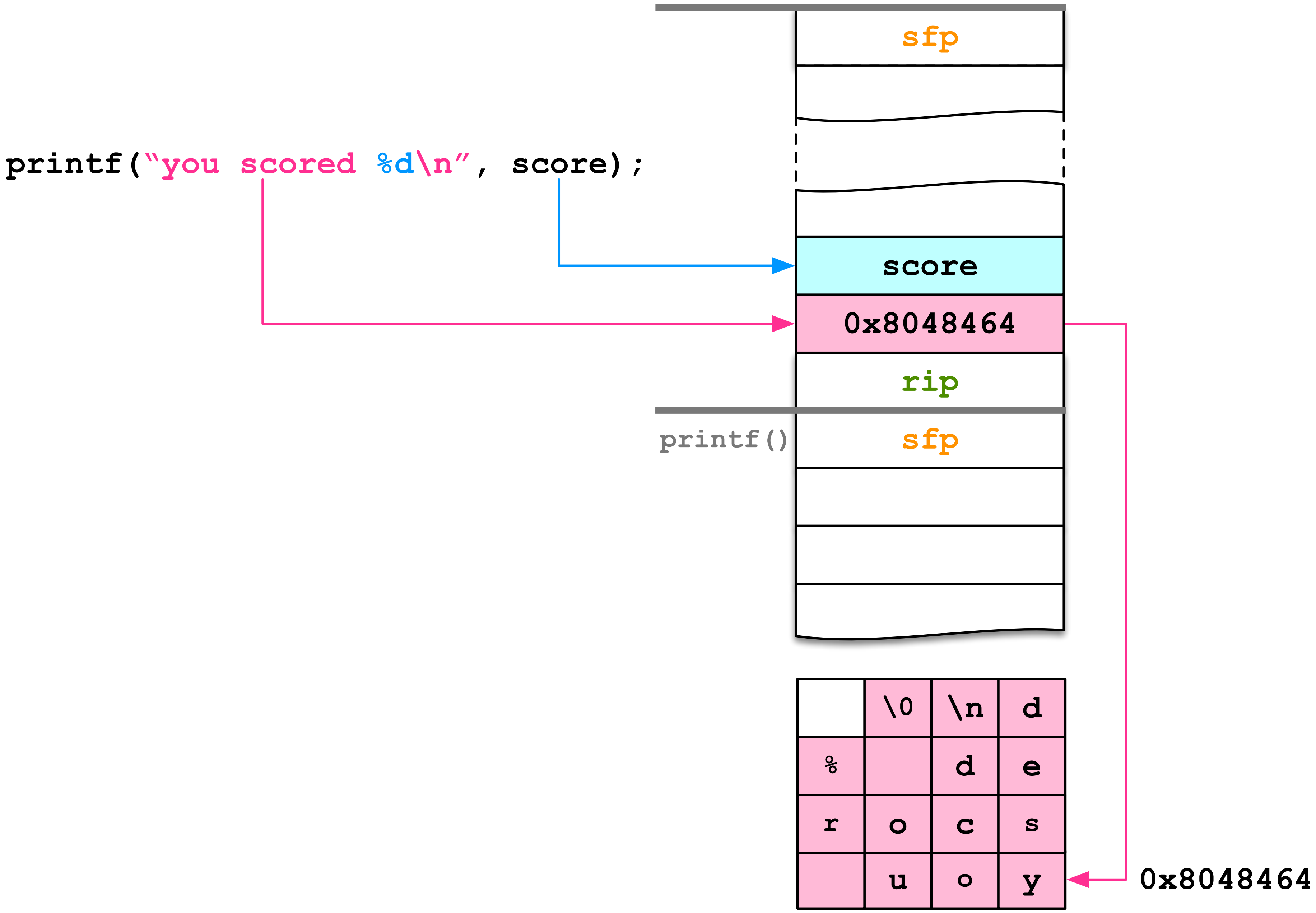
"The software is not geared to count more than 32,000 votes in a precinct. So what happens when it gets to 32,000 is the software starts counting backward," said Broward County Mayor Ilene Lieberman.

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);
```

sfp

score

0x8048464

rip

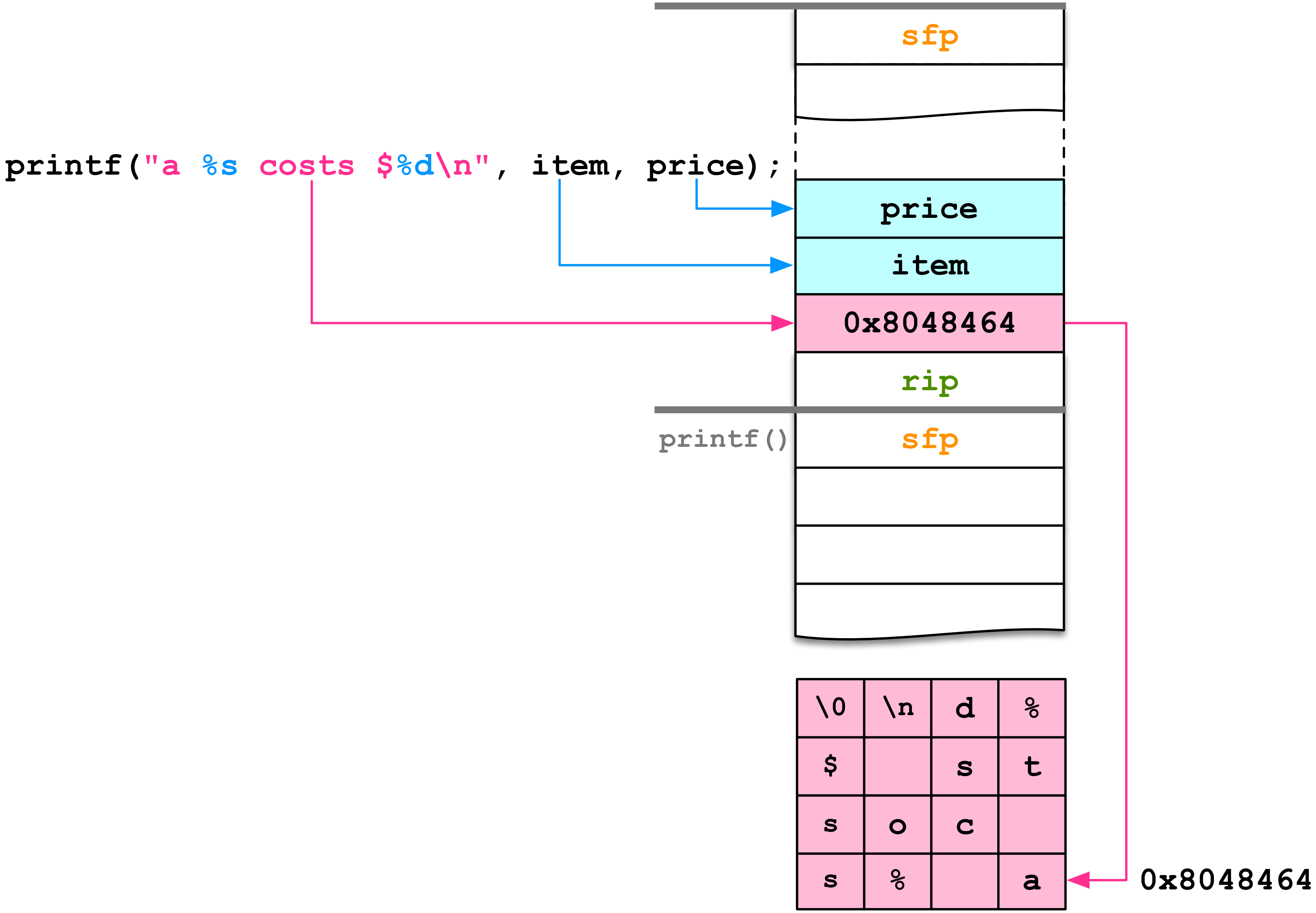
printf()

sfp

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

0x8048464

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



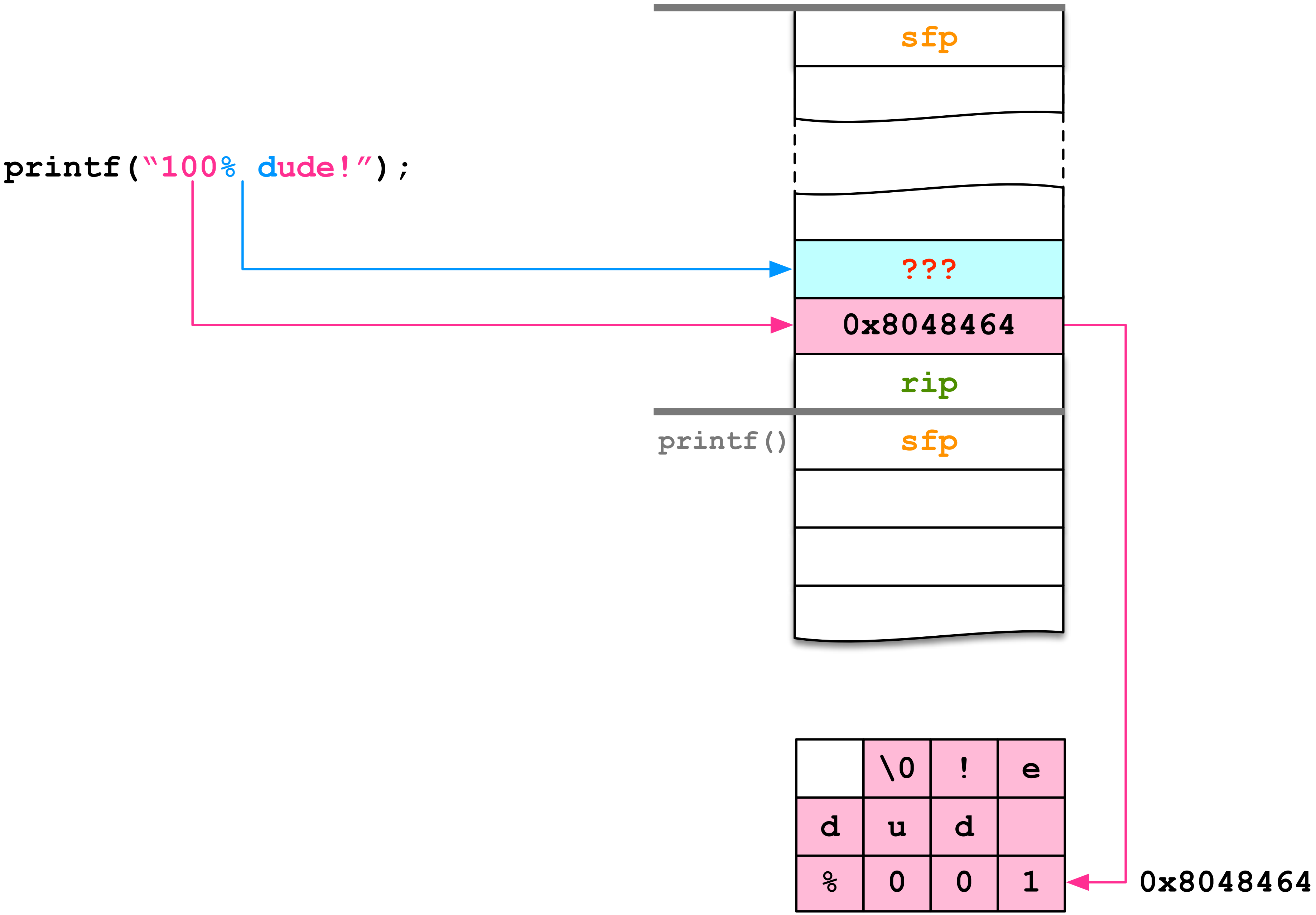
```
printf("a %s costs $%d\n", item, price);
```

\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!");
```

sfp

???

0x8048464

rip

printf()

sfp

	\0	!	e
d	u	d	
%	0	0	1

0x8048464

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!");
```

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);
}
```