CS 43: Computer Networks Security

Kevin Webb
Swarthmore College
December 7, 2017

Topics

- Spying on network traffic
- Classic problem: buffer overflow attack
- Monetizing botnets

Once upon a time…

- The Internet was “a group of mutually trusting users attached to a transparent network.”

- Result: Not much built-in security
  - Email headers
  - IP address spoofing
  - IP prefix hijacking
  - …

Once upon a time…

- Trust is gone, is the network still transparent?
- Switches help, cables can still be tapped…
- What about wireless?

- Wireless example/demo

Encryption

- Multiple options:
  - End to end (SSL, TLS): Browsers use this.
  - Link layer (WEP, WPA): Access point uses this.

- Facebook: enabled E2E encryption?
  - July 2013

Cryptography

- Dates back 1000's of years
- Simple substitution cipher (Caesar cipher)
  - Shift each letter by three (a -> d, b -> e, …)
  - "Hello world" becomes "khoor zruog"
- Many other, significantly better ciphers since…
  - De facto standard today: AES

(Symmetric) Cryptography

- Problem: Encrypting with a cipher requires shared “key” information. (prior to 1970’s)

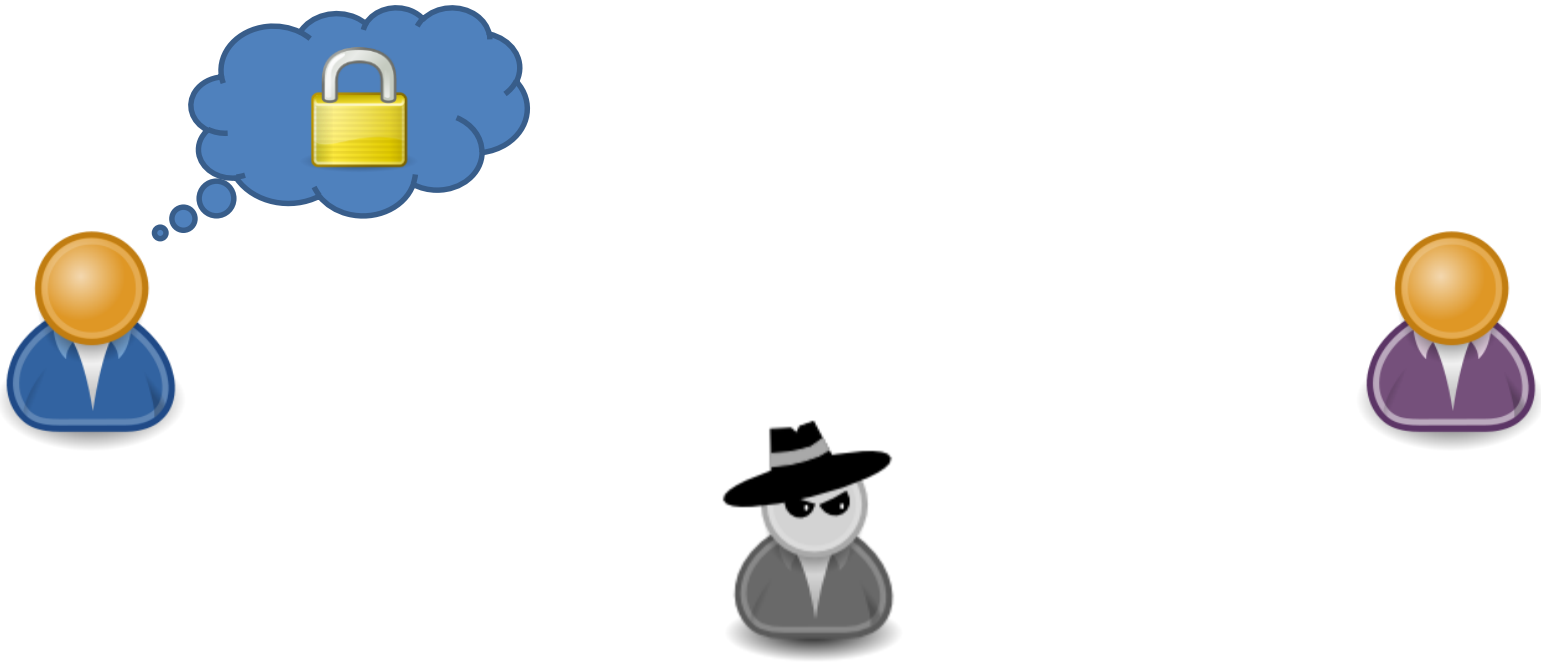
- Sophisticated cipher doesn’t help if we have to communicate the secret key!

Box Analogy

- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.

Box Analogy

- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.



Box Analogy

- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.

Box Analogy

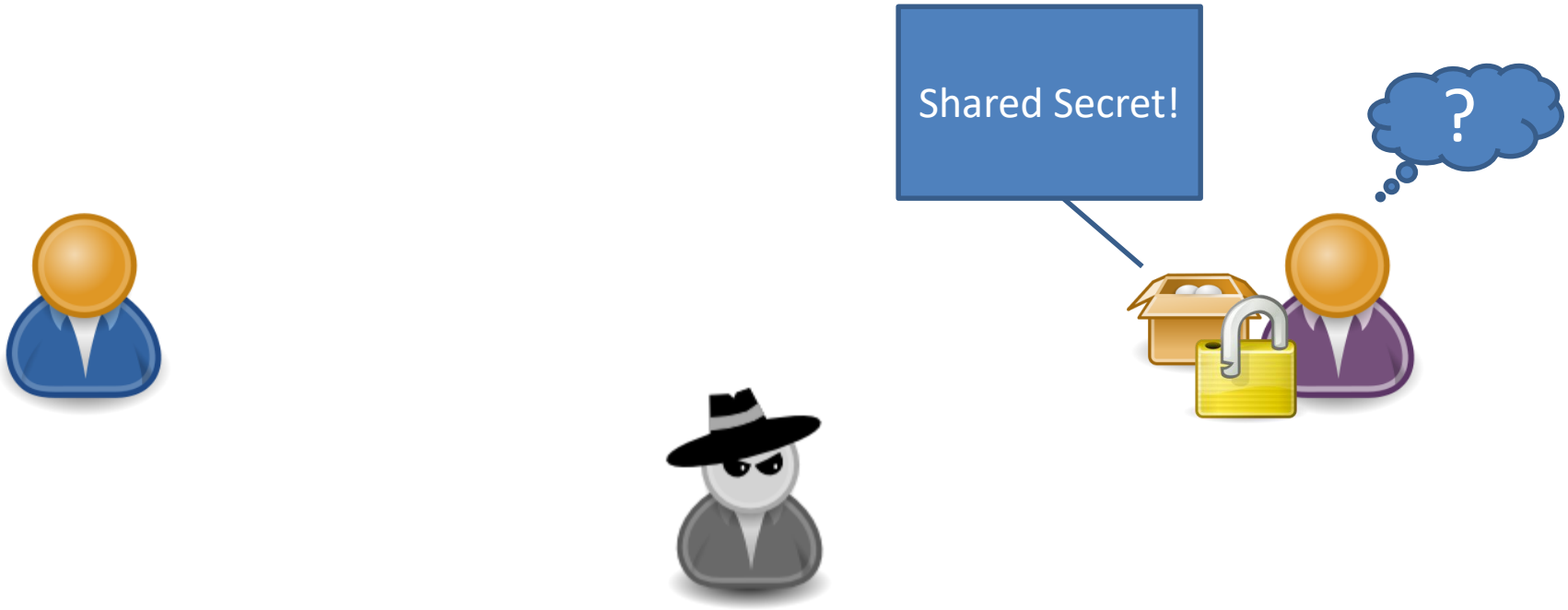
- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.

Box Analogy

- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.

Box Analogy

- You want to ship a package to someone.
- You trust it won't be stolen, but might be read.



(Asymmetric) Public Key Crypto

- Analogy: locking and unlocking are asymmetric
  - Anybody can lock
  - Very difficult to unlock without the key

- Let's apply this to data.
  - We need a function that's easy to apply in one direction and difficult in the other.

Factoring

- Multiplication is easy.
- Factoring (primes) is hard.

- 617077493 is the product of two primes.
  - What are they?

RSA Algorithm

- Rivest, Shamir, Adleman (RSA)

- Everyone computes two items:
  - Public key (kind of like a pad lock, ok if seen)
  - Private key (keep this to yourself)

- Receiver distributes public key, sender uses it to craft message that only receiver can read.

RSA Algorithm

- Choose two prime numbers of similar length.
  - P = 41 and Q = 29
- Compute N = P * Q
  - N = 41 * 29 = 1189
- Compute phi(N) = (P – 1) * (Q – 1)
  - phi(N) = (41 – 1) * (29 – 1) = 40 * 28 = 1120
- Choose a value e, 1 < e < 1120
  - e must not divide 1120, we'll pick 13

RSA Algorithm

P = 41 and Q = 29

N = 1189

phi(N) = 1120

e = 13

- Compute “modular multiplicative inverse” of e
  - Need: (d * e) % phi(N) = 1
  - d = 517

RSA Algorithm

P = 41 and Q = 29

N = 1189, phi(N) = 1120, e = 13, d = 517

- Public key is (n, e), private key is (n, d)

- To encrypt message m = 1000:
  - Take 1000^{13} % 1189 = 611
- To decrypt message 611:
  - 611^{517} % 1189 = 1000

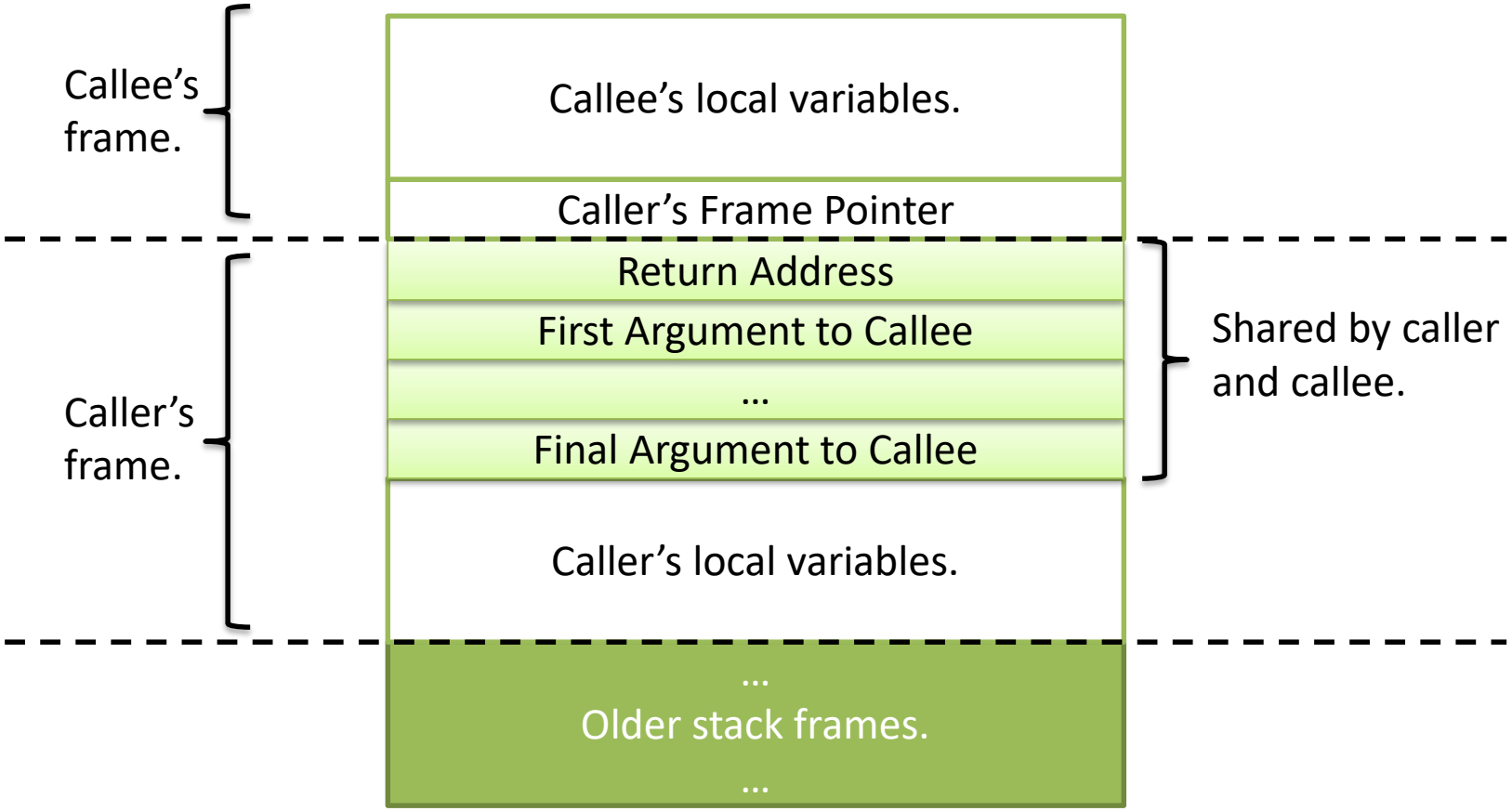
In Practice…

- Result: we can exchange secure messages with parties we've never talked to before!
  - (e.g., your bank)
- Exchange a secure message containing shared secret via RSA (asymmetric crypto)
- Subsequently use shared secret for conventional symmetric crypto (e.g., AES)

Classic Attack: Buffer Overflow

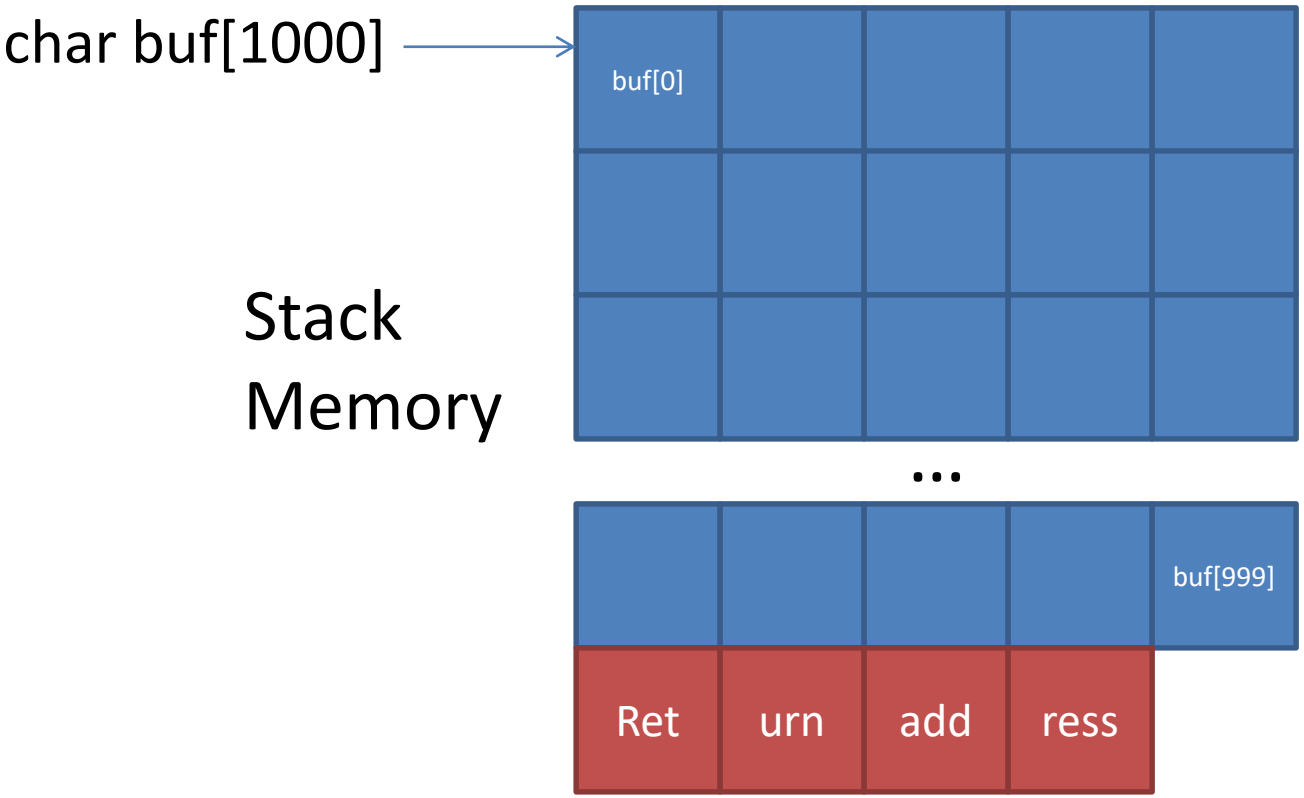
- Encryption ruining your (evil) day?
- Let’s try taking control instead!

Recall: The Stack



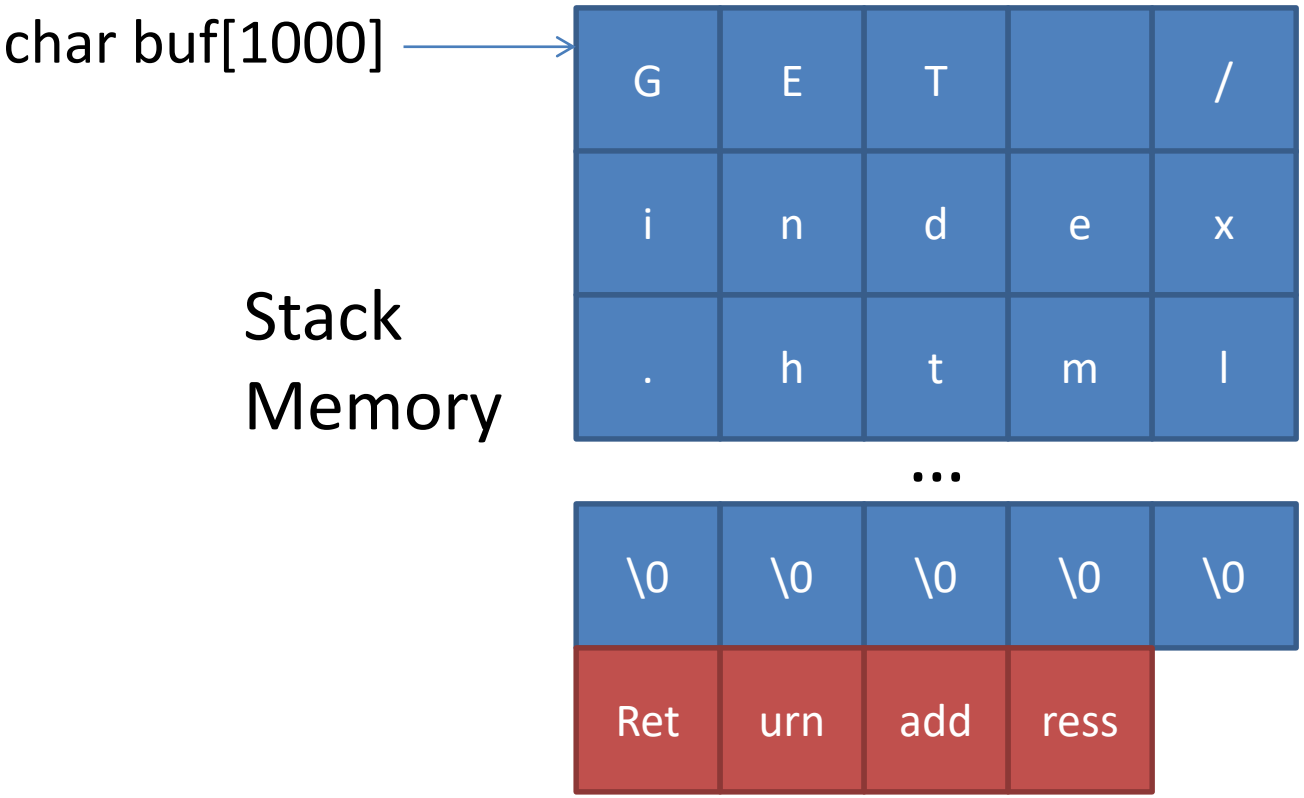
A well intentioned program…

- Suppose we have a protocol that does recv() until it finds \r\n\r\n.



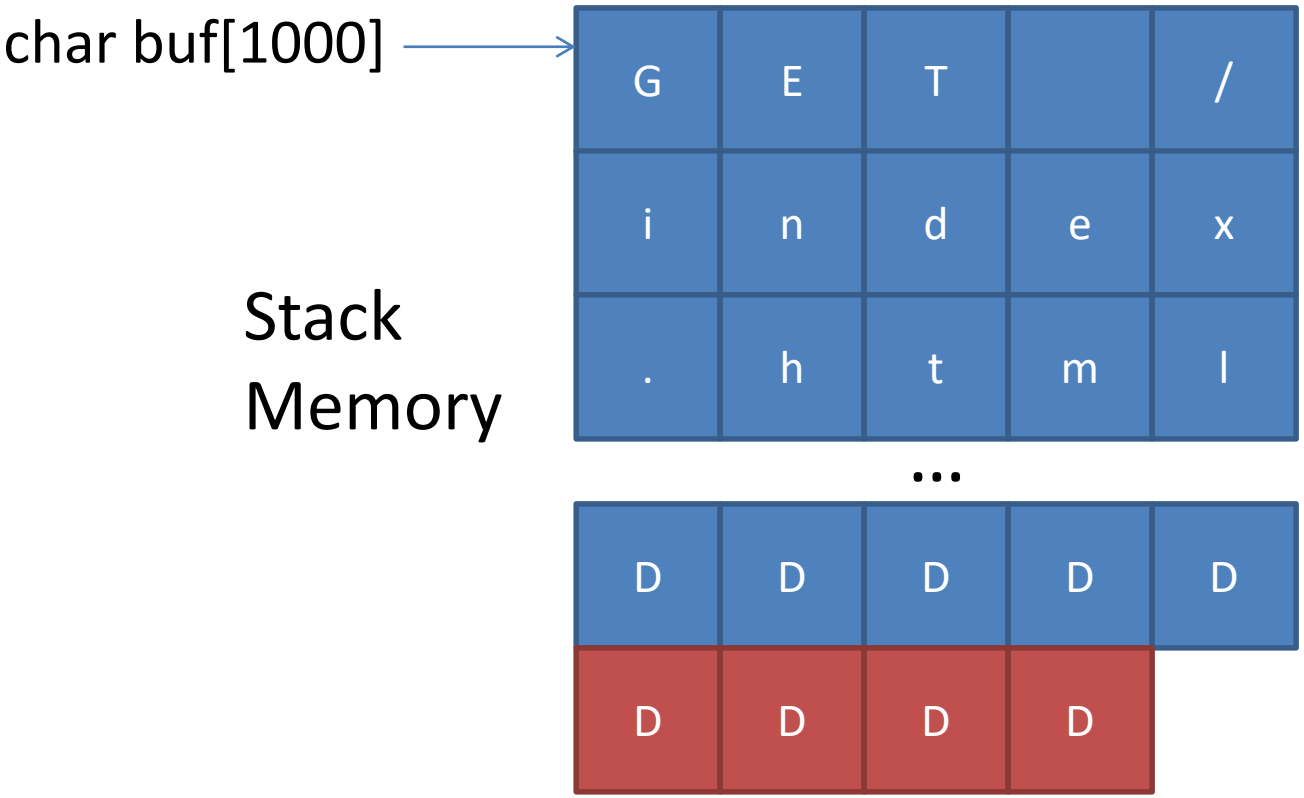
A well intentioned program…

- Suppose we have a protocol that does recv() until it finds \r\n\r\n.



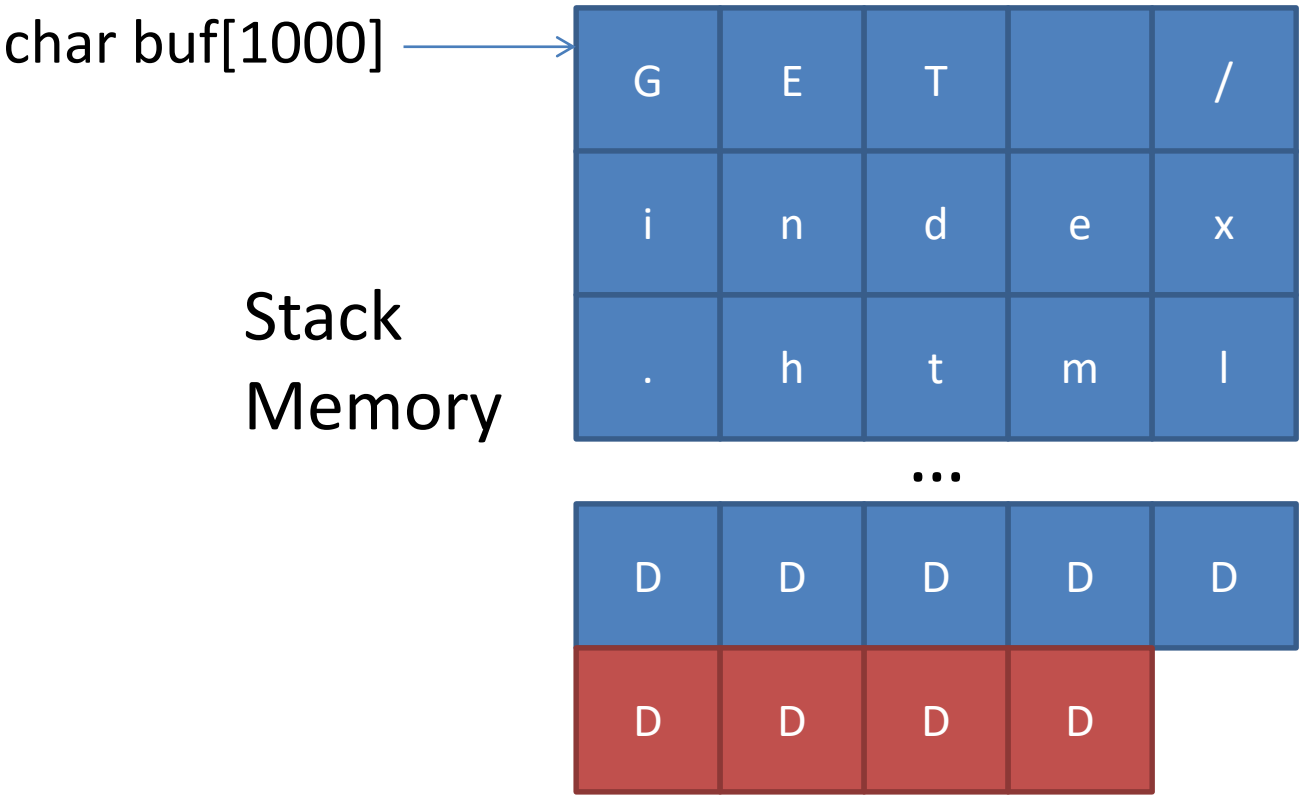
A well intentioned program…

- What happens if we're sent more than 1000 bytes before we see \r\n\r\n? Keep writing…



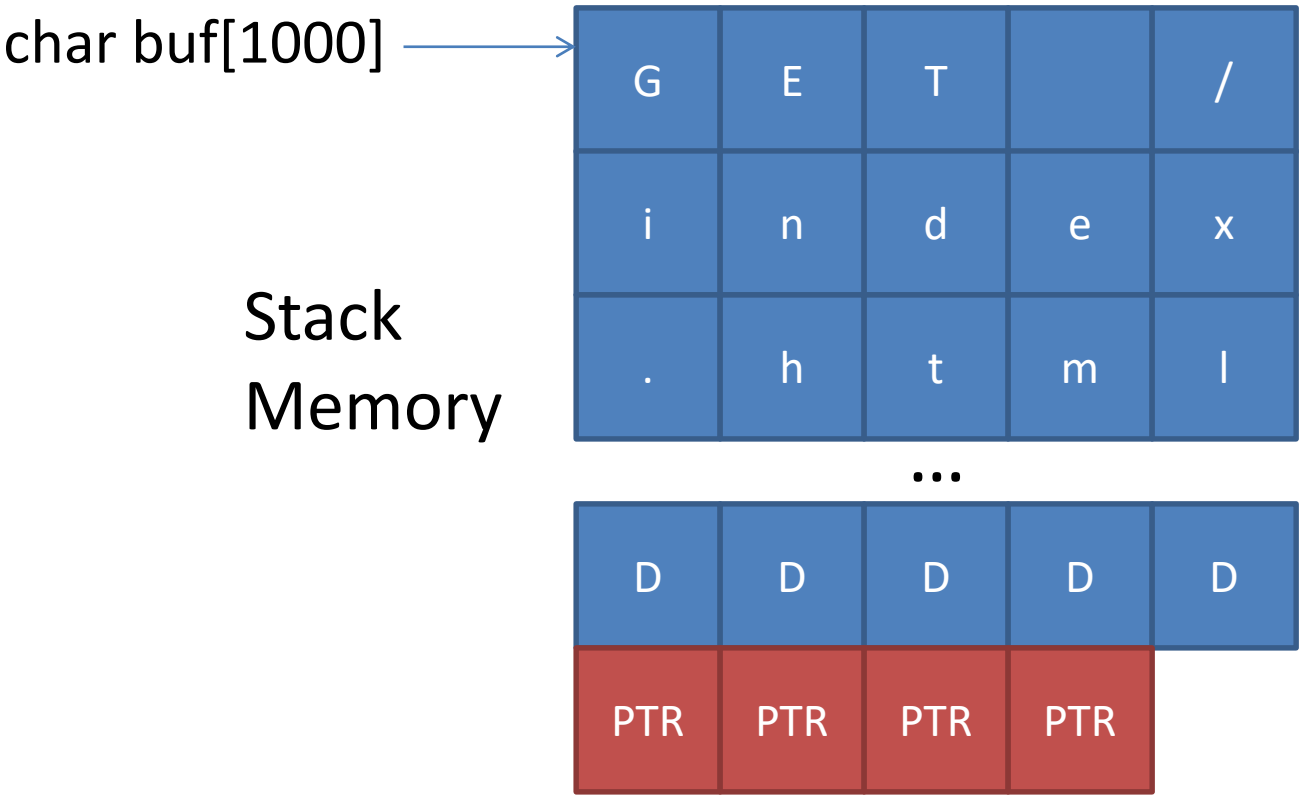
A well intentioned program…

- Uh, if we can overwrite the return address…
- We can control execution on return.



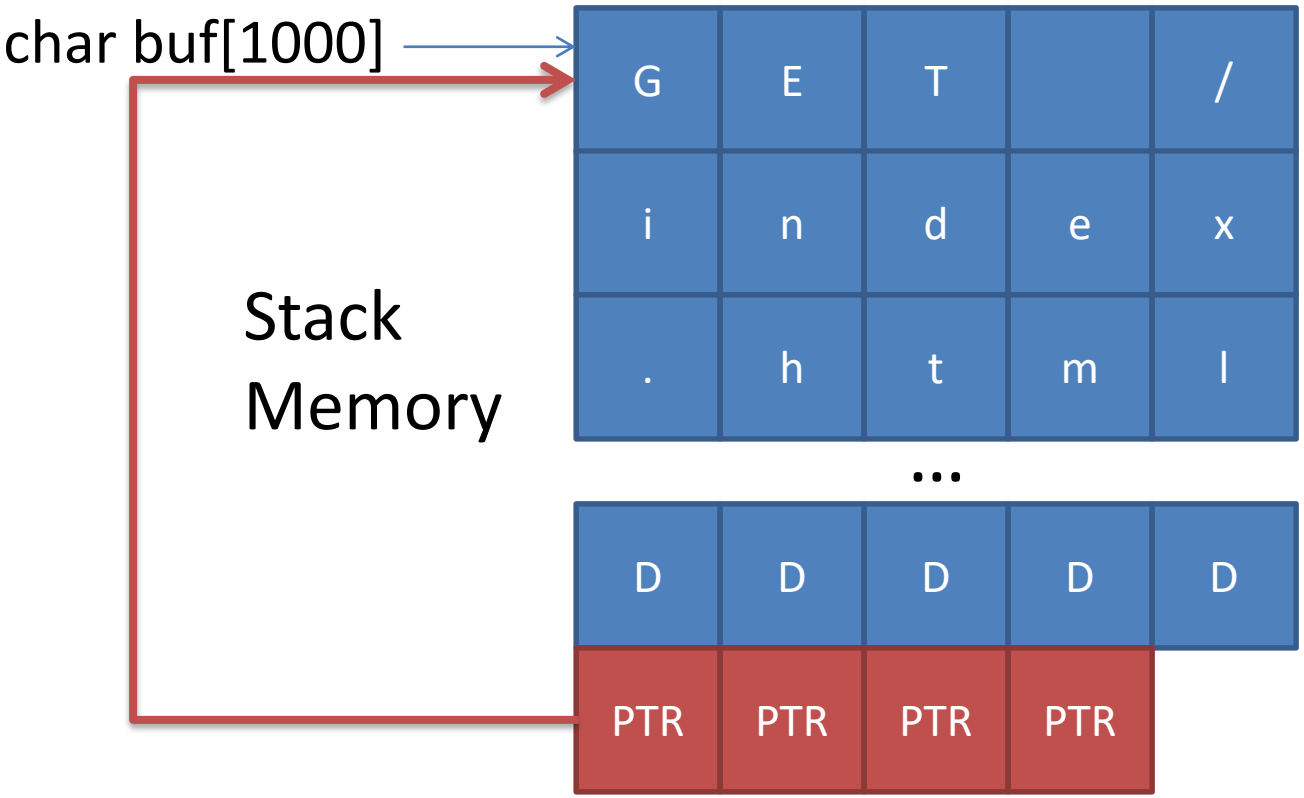
A well intentioned program…

- Let's send malicious data that contains a ptr.



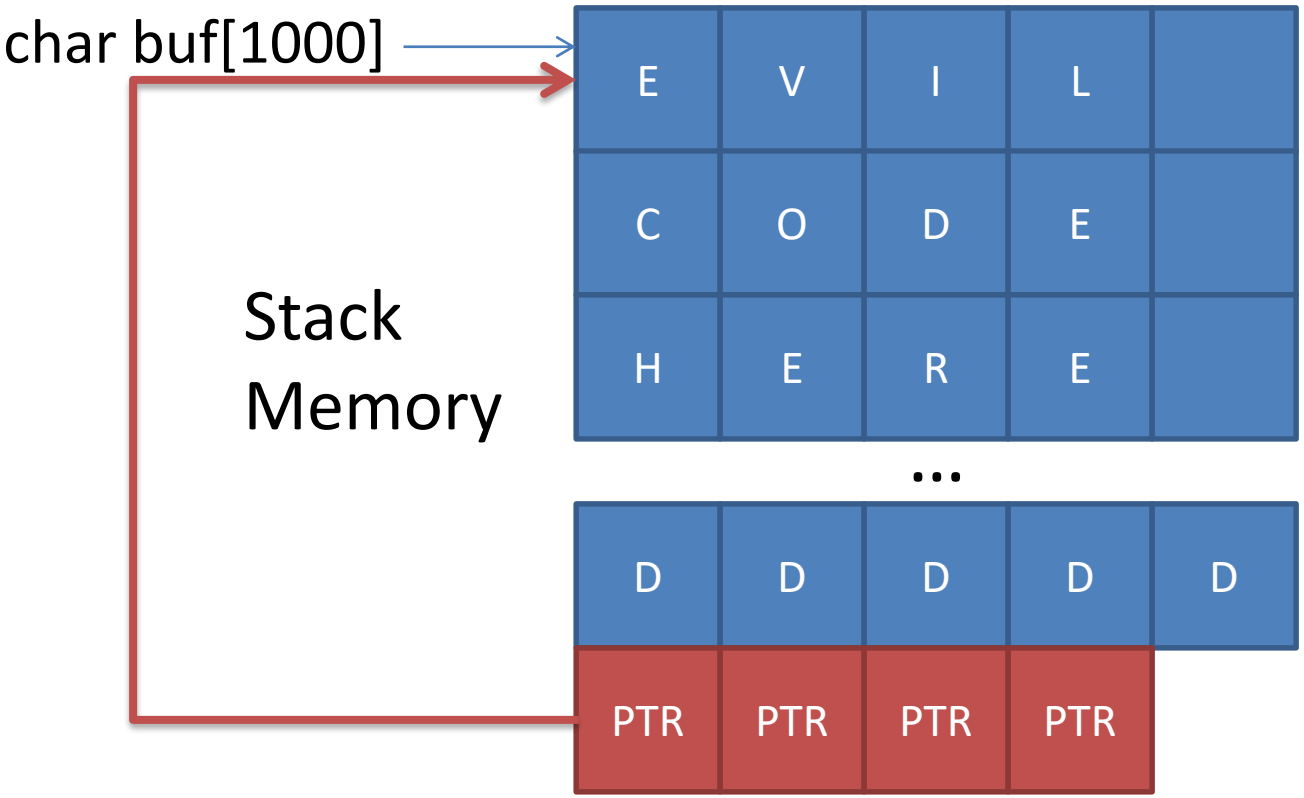
A well intentioned program…

- Let's send malicious data that contains a ptr.



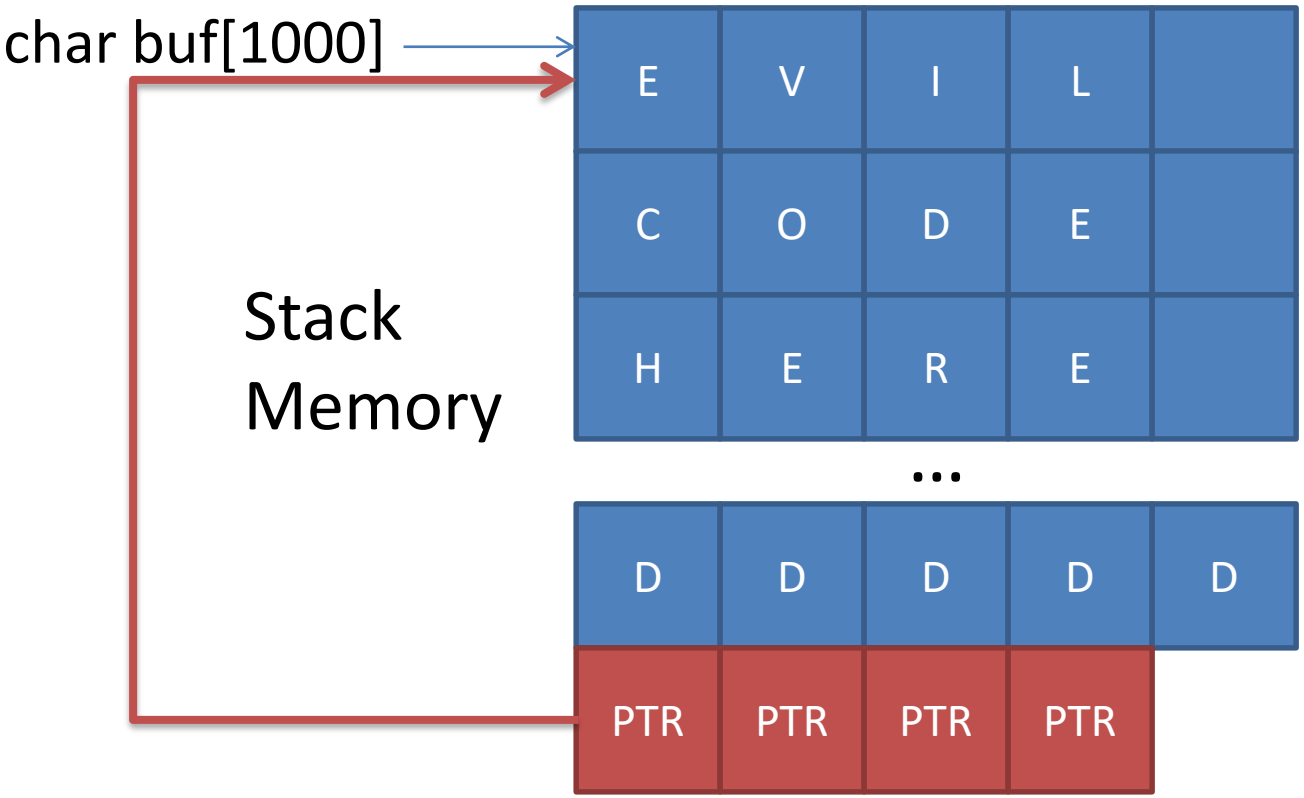
A well intentioned program…

- Oh, and also some commands up here…



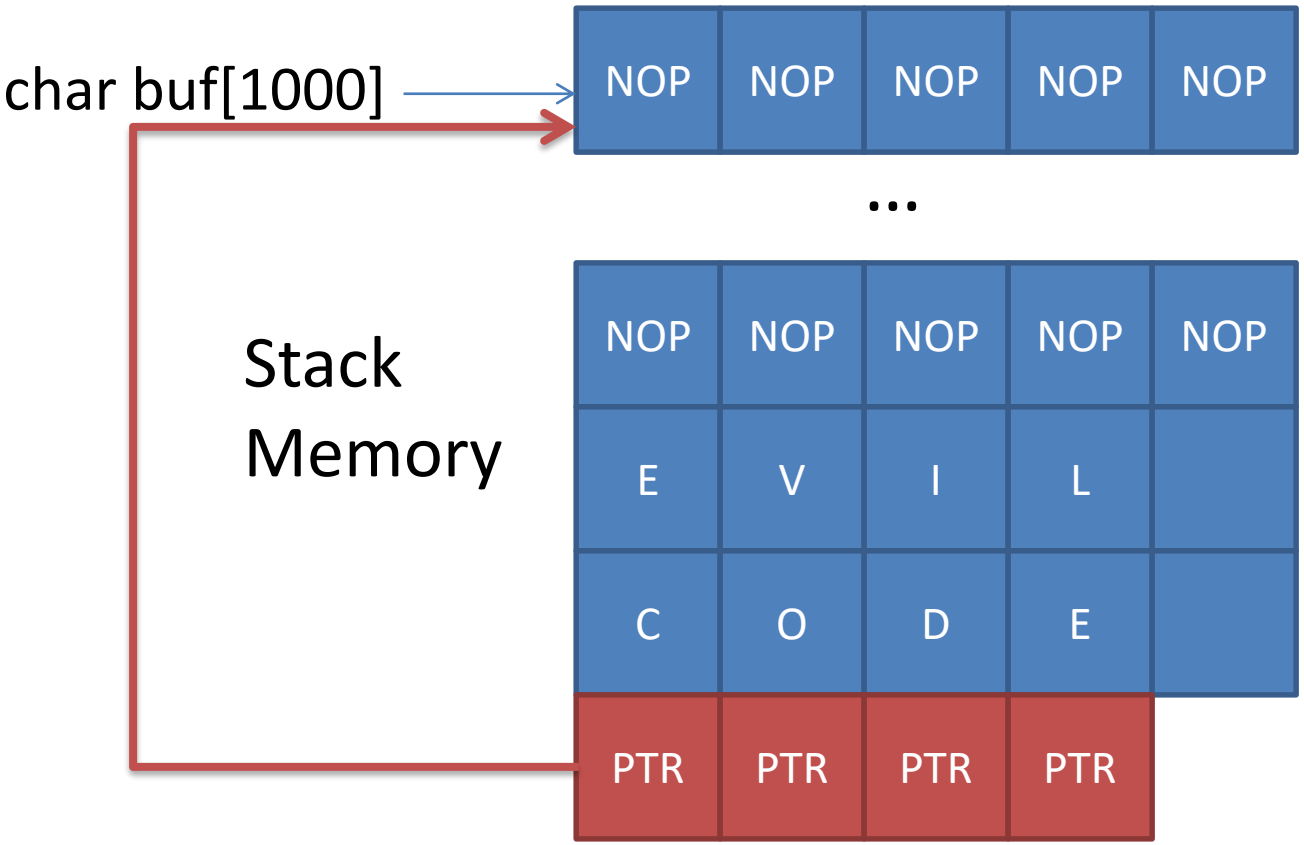
A well intentioned program…

- Function returns, executes evil code.



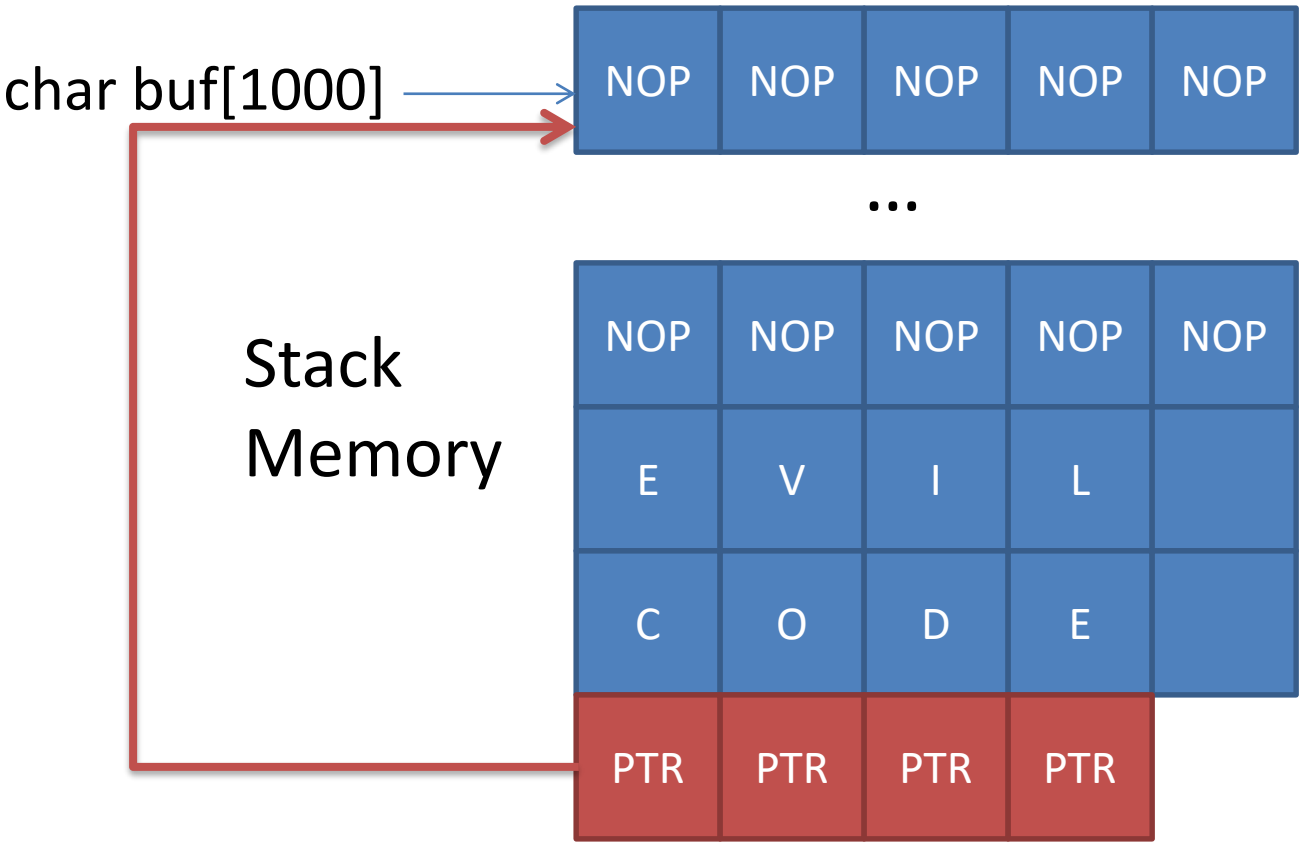
A well intentioned program…

- Improve chances: “NO OP sled”



A well intentioned program…

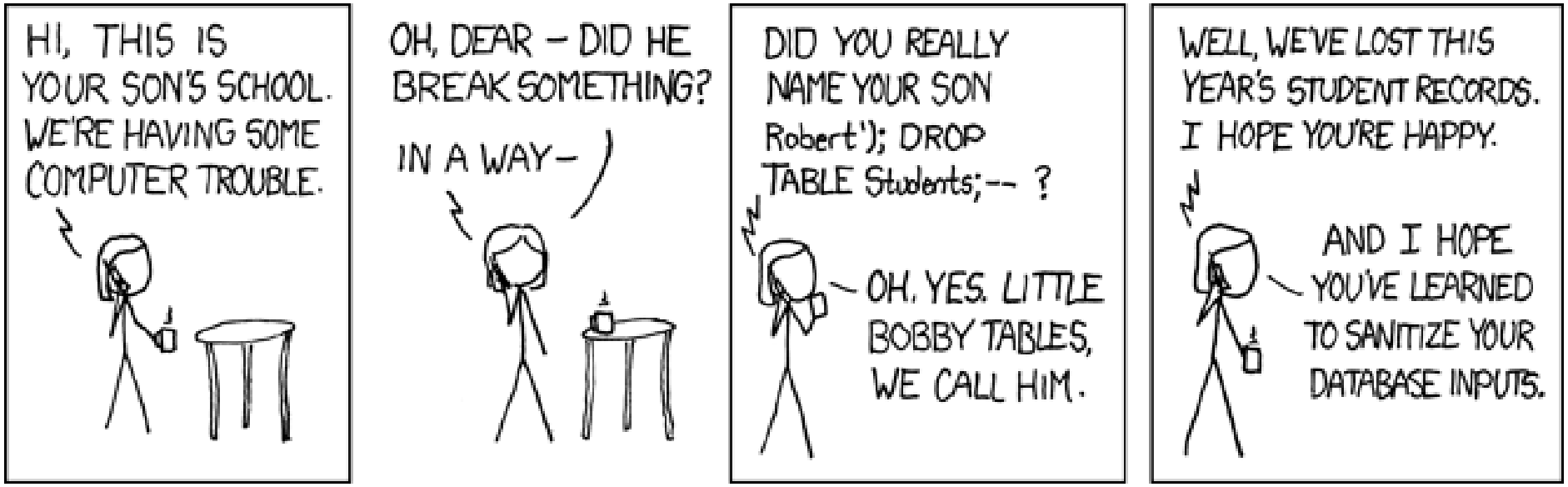
- See: “Smashing the Stack for Fun and Profit”



Input from Network

- Programs that receive user input are susceptible to buffer overflow (& more) attacks.
- Potentially much more problematic to receive input from the Internet!
- If attackers can take over program's control flow, they can execute *anything.*

Relevant XKCD #327



Alt text: Her daughter is named Help I'm trapped in a driver's license factory.

Bottom line: be careful about what you're accepting from the network!
Make sure the memory you're using is bounded and that the data is valid!

1988: The Morris Worm

- Cornell student Robert Morris
- Exploited buffer overflow in fingerd
  - It had a 512-byte buffer, he exploited it to execute /bin/sh, giving him shell access
- Told compromised host to download his worm code, it self-replicated by exploiting others
- Claimed “wanted to gauge size of Internet”

1988: The Morris Worm

- Worm did a check to see if it needed to replicate itself
  - If machine already compromised (process running) don't infect again.
- Worried about admins putting up fake process
  - Replicate anyway, at random, 1/7 times.
- This effectively shut down LOTS of machines.

1988: The Morris Worm

- Robert Morris:
  - First person convicted under Computer Fraud and Abuse Act
  - Sentenced to three years probation, 400 hours community service, $10,000
- Where is he now?

Exploits Today

- Worms
- Trojans (trick user)
- Browser exploits (drive-by downloads)

- Often used in BotNets

BotNets

- Having access to 1000's of machines is lucrative!

- Send Spam.
- Flood target with traffic (DDoS).
- Steal data (CC #'s, state secrets, etc.).
- Mine bitcoins.

Questions?

Final Exam

- Friday, December 15, at 9:00

- LOCATION: SCI 199