CS 134:
Operating Systems
Security

CS 134:
Operating Systems
Security

Overview

Overview

Introduction to Security
Classification

Cryptography

Attacks

A Note on Time-Wasting

Security offers many stories

Most are entertaining

Relatively few are edifying

Please don’t waste class time with unsolicited stories

A Note on Time-Wasting
Security offers many stories
Most are entertaining
Relatively few are edifying
Please don't waste class time with unsolicited stories

Understanding Security

Many OSes need security

Security is different from reliability:

Reliability is robustness in the face of *failures*

Security is robustness in the face of *attackers*

To understand security, we must understand

- Threats
- Breaches
- Defenses

Understanding Security

Many OSes need security

Security is different from reliability:
Reliability is robustness in the face of failures
Security is robustness in the face of attackers

To understand security, we must understand
- Threats
- Breaches
- Defenses

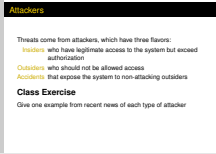
Attackers

Threats come from attackers, which have three flavors:

- **Insiders** who have legitimate access to the system but exceed authorization
- **Outsiders** who should not be allowed access
- **Accidents** that expose the system to non-attacking outsiders

Class Exercise

Give one example from recent news of each type of attacker



Attackers

Threats come from attackers, which have three flavors:

- **Insiders** who have legitimate access to the system but exceed authorization
- **Outsiders** who should not be allowed access
- **Accidents** that expose the system to non-attacking outsiders

Class Exercise

Give one example from recent news of each type of attacker

In practice, there is a fourth threat: outside→inside

Attackers

Threats come from attackers, which have three flavors:
Insiders who have legitimate access to the system but exceed authorization
Outsiders who should not be allowed access
Accidents that expose the system to non-attacking outsiders

Class Exercise

Give one example from recent news of each type of attacker

In practice, there is a fourth threat: outside→inside

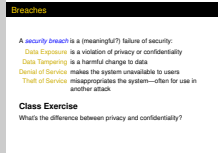
Breaches

A *security breach* is a (meaningful?) failure of security:

Data Exposure	is a violation of privacy or confidentiality
Data Tampering	is a harmful change to data
Denial of Service	makes the system unavailable to users
Theft of Service	misappropriates the system—often for use in another attack

Class Exercise

What's the difference between privacy and confidentiality?



Motivations

Attackers can have many motivations:

- **Exploration** for curiosity or as a prelude to another attack
- **Financial Gain** by direct theft of funds, theft of salable information, classroom cheating, financial manipulation, etc.
- **Vandalism** or simple maliciousness
- **Political Gain** often in the international arena
- **Revenge** of many kinds
- **Investigation** either by the state or by private entities

Motivations

Attackers can have many motivations:

Exploration for curiosity or as a prelude to another attack
Financial Gain by direct theft of funds, theft of salable information, classroom cheating, financial manipulation, etc.
Vandalism or simple maliciousness
Political Gain often in the international arena
Revenge of many kinds
Investigation either by the state or by private entities

Defenses

Defenses are much more varied than the attacks:

Obscurity	a la *The Purloined Letter*
Physical Security	to prevent outsider access
Authorization	to validate access
Enforcement	of access rules
Encryption	to hide secrets or aid validation
Monitoring	of access to detect attacks
Destruction	of unneeded sensitive data
Limitation	of unnecessary capabilities
etc.	

Defenses
Defenses are much more varied than the attacks:
Obscurity a la *The Purloined Letter*
Physical Security to prevent outsider access
Authorization to validate access
Enforcement of access rules
Encryption to hide secrets or aid validation
Monitoring of access to detect attacks
Destruction of unneeded sensitive data
Limitation of unnecessary capabilities
etc.

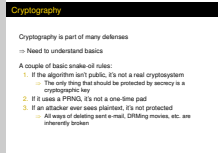
Cryptography

Cryptography is part of many defenses

⇒ Need to understand basics

A couple of basic snake-oil rules:

1. If the algorithm isn't public, it's not a real cryptosystem
   ⇒ The only thing that should be protected by secrecy is a cryptographic key
2. If it uses a PRNG, it's not a one-time pad
3. If an attacker ever sees plaintext, it's not protected
   ⇒ All ways of deleting sent e-mail, DRMing movies, etc. are inherently broken



Key Types

Two types of encryption/decryption keys:

1. Secret (“symmetric”) key shared between sender and receiver
   - Typically very fast
   - Prohibits many useful applications
   - Problem with getting key to both parties
   - Special case of one-time pad is only provably secure algorithm
     - All others (including public keys) boil down to “well, we haven’t cracked it *yet*”
2. Public/private key pair: one is published, the other secret
   - Usually symmetric in that either key can decrypt what the other encrypted
   - Typically quite slow
   - Clever applications

Key Types

Two types of encryption/decryption keys:

1. Secret (“symmetric”) key shared between sender and receiver
   - Typically very fast
   - Prohibits many useful applications
   - Problem with getting key to both parties
   - Special case of one-time pad is only provably secure algorithm
     - All others (including public keys) boil down to “well, we haven’t cracked it *yet*”
2. Public/private key pair: one is published, the other secret
   - Usually symmetric in that either key can decrypt what the other encrypted
   - Typically quite slow
   - Clever applications

Simple Public-Key Applications

Anybody can send me a secret message:

1. Look up my public key (it's on my Web page)
2. Encrypt with that key
3. Only I have the private key to decrypt it
4. If I have your public key, I can reply

I can broadcast an authenticated message:

1. Encrypt with my private key
2. Anybody can decrypt with public one
3. . . . but only I could have sent it

Simple Public-Key Applications
Anybody can send me a secret message:
1. Look up my public key (it's on my Web page)
2. Encrypt with that key
3. Only I have the private key to decrypt it
4. If I have your public key, I can reply
I can broadcast an authenticated message:
1. Encrypt with my private key
2. Anybody can decrypt with public one
3. . . . but only I could have sent it

Maybe your public key was even included in the message—but subject to MITM attacks

Cryptographic Hashes

Public keys are slow ⇒ so is sending large authenticated message

Solution: authenticate only a *hash* or *checksum* of message; if somebody tampers, hash will reveal that fact

Problem: suppose somebody tampers in such a way that the hash doesn't change?

This is called *hash collision* and destroys authenticity

Solution: *cryptographic hash* that makes it (effectively) impossible to find such a collision

Examples:

- MD4 (long since broken)
- MD5 (now broken!)
- SHA-1 (might be broken in next five years?)
- SHA-256, -512, etc.

Cryptographic Hashes
Public keys are slow ⇒ so is sending large authenticated message
Solution: authenticate only a hash or checksum of message; if somebody tampers, hash will reveal that fact
Problem: suppose somebody tampers in such a way that the hash doesn't change?
This is called hash collision and destroys authenticity
Solution: cryptographic hash that makes it (effectively) impossible to find such a collision
Examples:
- MD4 (long since broken)
- MD5 (now broken!)
- SHA-1 (might be broken in next five years?)
- SHA-256, -512, etc.

Diffie-Hellman Key Exchange

Problem: Not everybody has usable public key

- Amazon can publish theirs, but don't know yours
- No standard solution for people to publish PKs
- Besides, PKs are too slow

Solution: agree on a secret key on the fly

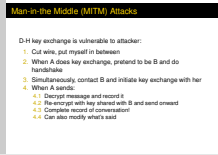
- Party A picks random numbers x and y, sends $x, x^y \mod m$ to B for some m
- Party B picks z, sends $x^z \mod m$ to A
- A can calculate $x^{yz} \mod m = (x^z)^y \mod m$ to use as key
- B can calculate $x^{yz} \mod m = (x^y)^z \mod m$
- Note that y and z never went over wire, so snooper can't do same calculation

Diffie-Hellman Key Exchange
Problem: Not everybody has usable public key
- Amazon can publish theirs, but don't know yours
- No standard solution for people to publish PKs
- Besides, PKs are too slow
Solution: agree on a secret key on the fly
- Party A picks random numbers x and y, sends $x, x^y \mod m$ to B for some m
- Party B picks z, sends $x^z \mod m$ to A
- A can calculate $x^{yz} \mod m = (x^z)^y \mod m$ to use as key
- B can calculate $x^{yz} \mod m = (x^y)^z \mod m$
- Note that y and z never went over wire, so snooper can't do same calculation

Man-in-the Middle (MITM) Attacks

D-H key exchange is vulnerable to attacker:

1. Cut wire, put myself in between
2. When A does key exchange, pretend to be B and do handshake
3. Simultaneously, contact B and initiate key exchange with her
4. When A sends:
   - 4.1 Decrypt message and record it
   - 4.2 Re-encrypt with key shared with B and send onward
   - 4.3 Complete record of conversation!
   - 4.4 Can also modify what's said



The Key-Exchange Problem

All cryptographic schemes suffer from a fundamental, insoluble problem: a local party cannot know whether the desired partner or an intruder is on the other end of a wire

Solution: have partner provide secret authentication token that only they know; public-key encryption can provide such a token

There seems to be a chicken-and-egg problem here...

This slide has animations.

The Key-Exchange Problem

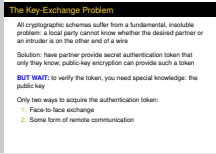
All cryptographic schemes suffer from a fundamental, insoluble problem: a local party cannot know whether the desired partner or an intruder is on the other end of a wire

Solution: have partner provide secret authentication token that only they know; public-key encryption can provide such a token

BUT WAIT: to verify the token, you need special knowledge: the public key

Only two ways to acquire the authentication token:

1. Face-to-face exchange
2. Some form of remote communication



There seems to be a chicken-and-egg problem here...

This slide has animations.

The Key-Exchange Problem

All cryptographic schemes suffer from a fundamental, insoluble problem: a local party cannot know whether the desired partner or an intruder is on the other end of a wire

Solution: have partner provide secret authentication token that only they know; public-key encryption can provide such a token

BUT WAIT: to verify the token, you need special knowledge: the public key

Only two ways to acquire the authentication token:

1. Face-to-face exchange
2. Some form of remote communication

BUT WAIT: the remote communication must *itself* be authenticated or an MITM attack can substitute a fake token!

The Key-Exchange Problem
All cryptographic schemes suffer from a fundamental, insoluble problem: a local party cannot know whether the desired partner or an intruder is on the other end of a wire
Solution: have partner provide secret authentication token that only they know; public-key encryption can provide such a token
BUT WAIT: to verify the token, you need special knowledge: the public key
Only two ways to acquire the authentication token:
1. Face-to-face exchange
2. Some form of remote communication
BUT WAIT: the remote communication must itself be authenticated or an MITM attack can substitute a fake token!

There seems to be a chicken-and-egg problem here. . .

This slide has animations.

Public-Key Infrastructure (PKI)

It's not practical to visit Amazon, Netflix, Newegg, etc. to get copies of their public keys

Failed solution: have them publish public keys on their Web pages

Class Exercise

Why doesn't that work?

Public-Key Infrastructure (PKI)
It's not practical to visit Amazon, Netflix, Newegg, etc. to get copies of their public keys
Failed solution: have them publish public keys on their Web pages
Class Exercise
Why doesn't that work?

There are two problems here:

1. You have to trust that the key list issued with the browser is correct. Unless you personally visited (and trust) the browser manufacturer, a third party could have introduced a substitute.
2. The PKI is as secure as its weakest link. If a repository's private key is leaked, anybody can pretend to be Amazon. This has happened.

A partial solution is *key revocation*. That's hard, and we'll ignore that issue here.

Public-Key Infrastructure (PKI)

It's not practical to visit Amazon, Netflix, Newegg, etc. to get copies of their public keys

Failed solution: have them publish public keys on their Web pages

Class Exercise

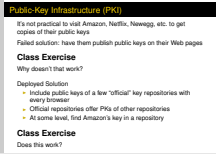
Why doesn't that work?

Deployed Solution

- Include public keys of a few "official" key repositories with every browser
- Official repositories offer PKs of other repositories
- At some level, find Amazon's key in a repository

Class Exercise

Does this work?



There are two problems here:

1. You have to trust that the key list issued with the browser is correct. Unless you personally visited (and trust) the browser manufacturer, a third party could have introduced a substitute.
2. The PKI is as secure as its weakest link. If a repository's private key is leaked, anybody can pretend to be Amazon. This has happened.

A partial solution is *key revocation*. That's hard, and we'll ignore that issue here.

Key Management

Suppose your homework is encrypted with a passphrase of “CS 134 is cool”. You decrypt your homework to work on it.

- Is the key still in memory? It could be in:
  - Terminal buffers
  - Local memory of decryption program (stack, heap)
  - Swap space
  - Memory of intermediate programs like X server and `screen`
- Have you ever accidentally typed into the wrong window?
- Are you *certain* there’s no key logger on your machine?
- Are you *certain* you can trust your OS?
- Are you *certain* you can trust your CPU chip?

Key Management

Suppose your homework is encrypted with a passphrase of “CS 134 is cool”. You decrypt your homework to work on it.

- Is the key still in memory? It could be in:
  - Terminal buffers
  - Local memory of decryption program (stack, heap)
  - Swap space
  - Memory of intermediate programs like X server and screen
- Have you ever accidentally typed into the wrong window?
- Are you certain there’s no key logger on your machine?
- Are you certain you can trust your OS?
- Are you certain you can trust your CPU chip?

Defense Rule #1

Rule #1 of defending against bad guys is the same regardless of whether you're doing computer security, neighborhood crime patrols, or interstellar warfare: think like the enemy.

This means you need to develop a nasty attitude. When you walk out of here today, look for all the ways an off-campus thief could (try to) get rich. Could they succeed?

Defense Rule #1

Rule #1 of defending against bad guys is the same regardless of whether you're doing computer security, neighborhood crime patrols, or interstellar warfare: think like the enemy.

This means you need to develop a nasty attitude. When you walk out of here today, look for all the ways an off-campus thief could (try to) get rich. Could they succeed?

HMC depends a lot on a combination of the honor code and the fact that we have good mechanisms for keeping outsiders off campus.

Common Attacks

We've already seen MITM. Other common attacks include:

- Logic bombs
- Back doors
- Random probes
- Password guessing
- Privilege escalation
- Buffer overflows (oh my!)
- Trojan horses
- Viruses
- Worms
- Social engineering

But that's not all. . .

Common Attacks
We've already seen MITM. Other common attacks include:
- Logic bombs
- Back doors
- Random probes
- Password guessing
- Privilege escalation
- Buffer overflows (oh my!)
- Trojan horses
- Viruses
- Worms
- Social engineering

But that's not all. . .

The point here is that no list of attacks is comprehensive.

The Rounding Attack

This really happened:

- Banks have to round interest to nearest penny
- Programmer rewrote rounding code:
  1. If < 0.5, round down normally
  2. If ≥ 0.5, *still* round down

  ... But that leaves bank out of balance, so credit leftover penny to own account and everything works out just fine!
- Every month, hits 50% of customers on average
- Even small bank has thousands of customers. . . big one has hundreds of thousands or millions

So. . . how did he get caught? (Yes, he got caught.)

The Rounding Attack

This really happened:
- Banks have to round interest to nearest penny
- Programmer rewrote rounding code:
  1. If < 0.5, round down normally
  2. If ≥ 0.5, still round down

  ... But that leaves bank out of balance, so credit leftover penny to own account and everything works out just fine!
- Every month, hits 50% of customers on average
- Even small bank has thousands of customers. . . big one has hundreds of thousands or millions

So. . . how did he get caught? (Yes, he got caught.)

The point of talking about this attack is that it doesn't fall into the neat categories from the previous slide.
The bad guy was caught because a "little old lady" checked her statement carefully, and when it didn't make sense she went to the bank and asked for help.
But there's another enduring principle here: greed. The bad guy could have fled before he was uncovered, but the money was rolling in every month and so he kept wanting more.

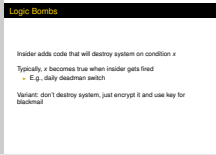
Logic Bombs

Insider adds code that will destroy system on condition x

Typically, x becomes true when insider gets fired

- E.g., daily deadman switch

Variant: don't destroy system, just encrypt it and use key for blackmail



Back Doors

Rewrite login program to accept hardwired account and password

Insider can now get root access even after being fired and having account deleted

For insidiously nasty variant, read “Reflections on Trusting Trust,” Ken Thompson’s Turing Award lecture

Scary thought: it can be done in hardware, and neither we nor Intel have a way to find out if it has been

Back Doors

Rewrite login program to accept hardwired account and password

Insider can now get root access even after being fired and having account deleted

For insidiously nasty variant, read “Reflections on Trusting Trust,” Ken Thompson’s Turing Award lecture

Scary thought: it can be done in hardware, and neither we nor Intel have a way to find out if it has been

Random Probes

Myth: “Sure, they attack Google all the time. But nobody knows my machine even exists.”

Reality: Bad guys don’t need to know your name or where you are. They just have to try all possible IP addresses. (Even in IPv6, this can be done.)

⇒ *Assume intruders will find you and probe you, unless a firewall protects you*

Random Probes

Myth: “Sure, they attack Google all the time. But nobody knows my machine even exists.”

Reality: Bad guys don’t need to know your name or where you are. They just have to try all possible IP addresses. (Even in IPv6, this can be done.)

⇒ *Assume intruders will find you and probe you, unless a firewall protects you*

Password Guessing

Having probed, log into an account:

- User `guest`, password `guest`
- `admin`/`admin`
- `root`/<null> (really!)

Bad guys have huge lists of common accounts (e.g., `phpadmin`, `cisco`, `help`) and passwords

Variation: acquire encrypted passwords and rather than decrypting, run common passwords through one-way encryption algorithm to search for hits (“dictionary attack”)

Password Guessing

Having probed, log into an account:
- User guest, password guest
- admin/admin
- root/<null> (really!)

Bad guys have huge lists of common accounts (e.g., phpadmin, cisco, help) and passwords

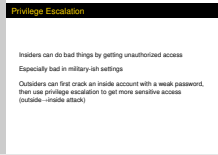
Variation: acquire encrypted passwords and rather than decrypting, run common passwords through one-way encryption algorithm to search for hits (“dictionary attack”)

Privilege Escalation

Insiders can do bad things by getting unauthorized access

Especially bad in military-ish settings

Outsiders can first crack an inside account with a weak password, then use privilege escalation to get more sensitive access (outside→inside attack)



Buffer Overflows

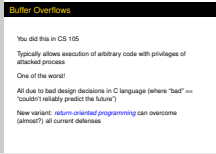
You did this in CS 105

Typically allows execution of arbitrary code with privileges of attacked process

One of the worst!

All due to bad design decisions in C language (where “bad” == “couldn’t reliably predict the future”)

New variant: *return-oriented programming* can overcome (almost?) all current defenses



ROP is a bit like level 0 of the buffer bomb, where you just called a preexisting function. The only defense I can see is to arrange that every time a new stack frame is created, the VM tables are adjusted such that nothing above the local variables is writable. I don’t think that’s practical, for several reasons.

Trojan Horses

Pretend to be what you're not

Canonical example:

```
clear_screen();
printf("Login: ");
gets(login_name);
printf("Password: ");
gets(password);
    /* record the stolen information */
printf("Login failed\n");
execv("/bin/login", NULL);
```

User reveals password, thinks she just mistyped it

Note that phishing is a variant on the Trojan horse

Trojan Horses
Pretend to be what you're not
Canonical example:
clear_screen();
printf("Login: ");
gets(login_name);
printf("Password: ");
gets(password);
/* record the stolen information */
printf("Login failed\n");
execv("/bin/login", NULL);
User reveals password, thinks she just mistyped it
Note that phishing is a variant on the Trojan horse

Viruses and Worms

Both are self-propagating programs: make new copies in places that will let them spread further

Virus: attaches itself to a legitimate program; when real program is run, spreads

Worm: standalone program that tries to infect other systems, either via network or “sneakernet” (e.g., USB drive)

Worm propagation is often via weaknesses such as phishing or buffer overflows

Virus-ness or worm-ness is secondary; they’re just carriers for malware

Viruses and Worms

Both are self-propagating programs: make new copies in places that will let them spread further

Virus: attaches itself to a legitimate program; when real program is run, spreads

Worm: standalone program that tries to infect other systems, either via network or “sneakernet” (e.g., USB drive)

Worm propagation is often via weaknesses such as phishing or buffer overflows

Virus-ness or worm-ness is secondary; they’re just carriers for malware

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

Social Engineering
Basic idea: trick humans into doing what you want
Usually depends on fact that people are either (a) helpful or (b) venal:

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick's book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

Social Engineering
Basic idea: trick humans into doing what you want
Usually depends on fact that people are either (a) helpful or (b) venal:
“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick’s book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

Social Engineering
Basic idea: trick humans into doing what you want
Usually depends on fact that people are either (a) helpful or (b) venal:
“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”
“Click here for naked pictures of Dustin Hoffman”

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick’s book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick’s book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

. . . or just drop a USB drive in a parking lot.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

. . . or just drop a USB drive in a parking lot.

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick’s book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

. . . or just drop a USB drive in a parking lot.

Note that phishing is a special case of social engineering

Social Engineering

Basic idea: trick humans into doing what you want

Usually depends on fact that people are either (a) helpful or (b) venal:

“Hey, I’m at a customer site and I must have forgotten the stupid root password. What is it again?”

“Click here for naked pictures of Dustin Hoffman”

“My name is Mrs. Abdullah Suckergrabber and in the name of Jesus I needs your help to transfer $40 MILLION dollars that my late husband stoel from the impoverished person of Afrika.”

. . . or just drop a USB drive in a parking lot.

Note that phishing is a special case of social engineering

Note that the USB-drive attack only works because Windows will stupidly auto-run software from an unknown source.

Kevin Mitnick’s book has a wealth of social-engineering attacks.

Basic principle: carry a clipboard and people will trust you.