

# Instructions 1

## Elevation of Privilege Instructions

Draw a diagram of the system you want to threat model before you deal the cards.

Deal the deck to 3-6 players. Play starts with the 3 of Tampering. Play clockwise, and each player in turn follows in the suit if they have a card in the suit. If they don't have that suit, they can play another suit. The high card played takes the trick, with Elevation of Privilege taking precedence over the suit lead. Only Elevation of Privilege (EoP) or the lead suit can take a trick.

To play a card, read the card, announce your threat and record it. If the player can't link the threat to the system, play proceeds.

Take few minutes between hands to think about threats.

### Points:

1 for a threat on your card, +1 for taking the trick

# Instructions

## Elevation of Privilege Instructions

Threats should be articulated clearly, testable, and addressable. In the event that a threat leads to an argument, the threat should resolve by asking the question: "Would we take an actionable bug, feature request or design change for that?" If the answer is yes, it is a real threat. (This doesn't mean that threats outside of that aren't real, it's simply a way to focus discussion on actionable threats.) Questions that start with "There's a way" should be read as "There's a way...and here's how..." while questions that start with "Your code" should be read "The code we're collectively creating...and here's how."

The deck contains a number of special cards: trumps and open threats. EoP cards are trumps. They take the trick even if they are lower value than the suit that was led. The ace of each suit is an open threat card. When played, the player must identify a threat not listed on another card.

When all the cards have been played, whoever has the most points wins.

**Remember to have fun!**

# Instructions

# Instructions 2

## Elevation of Privilege Variants

### **Optional/variants:**

- You may pass cards after the third trick. This is helpful if you have cards that you can't tie to the system. Someone else may be able to.
- Double the number of points, and give one point for threats on other people's cards.
- Other players may "riff" on the threat and if they do, they get one point per additional threat.
- Limit riffing to no more than 60 seconds.
- Mark up the diagram with where the threat occurs.

**Questions are listed on the threat cards to help with the Aces.**

Thanks to Laurie Williams for inspiration.

Instructions



© 2009 Microsoft Corporation. This work is licensed under the Creative Commons Attribution 3.0 United States License. To view the full content of this license, visit <http://creativecommons.org/licenses/by/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

# Instructions

## Context



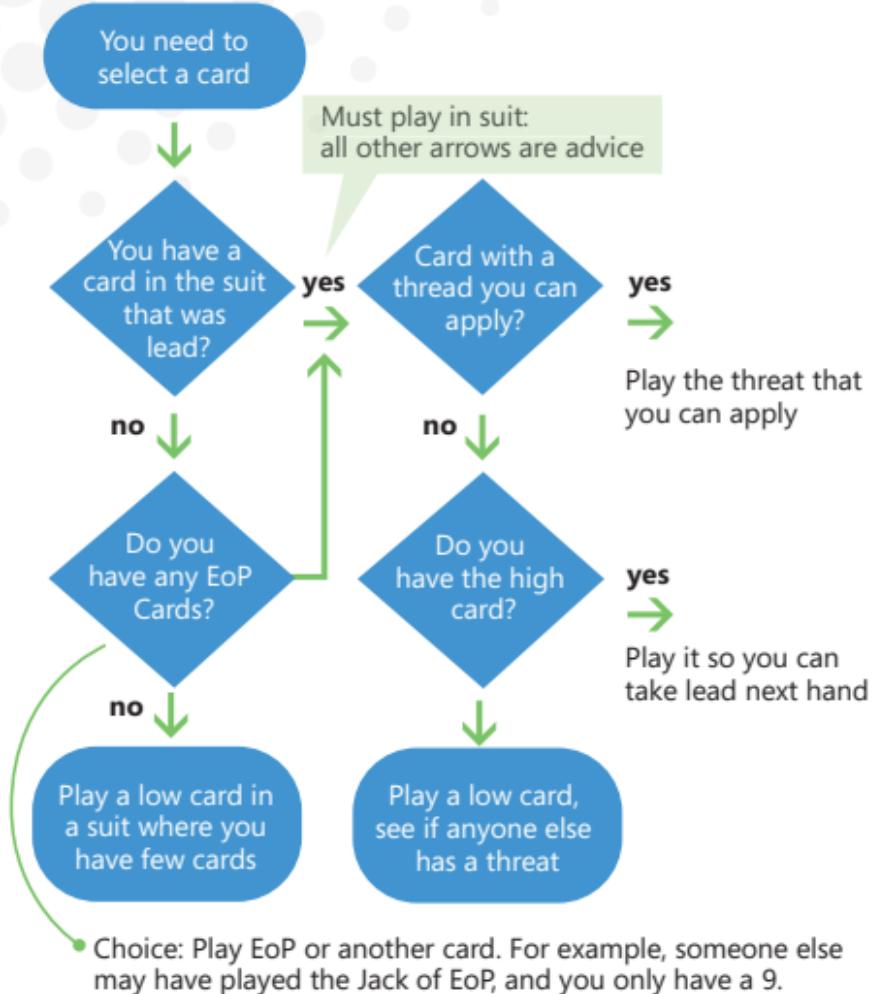
## Play

Whoever has 3 of Tampering starts

**Whoever has 3 of Tampering:**



# Strategy



**Microsoft®**

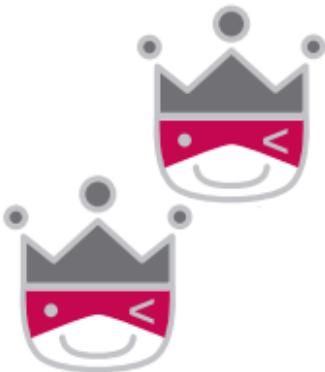
# elevation of privilege



# 2

## Spoofing

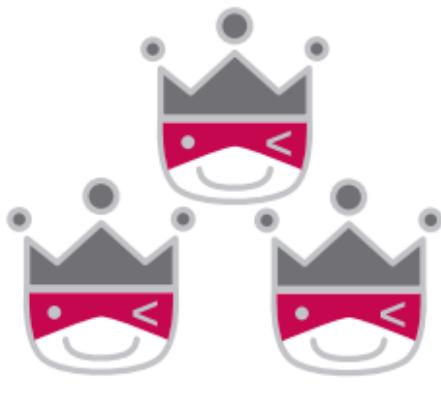
An attacker could squat on the random port or socket that the server normally uses



# 3

## Spoofing

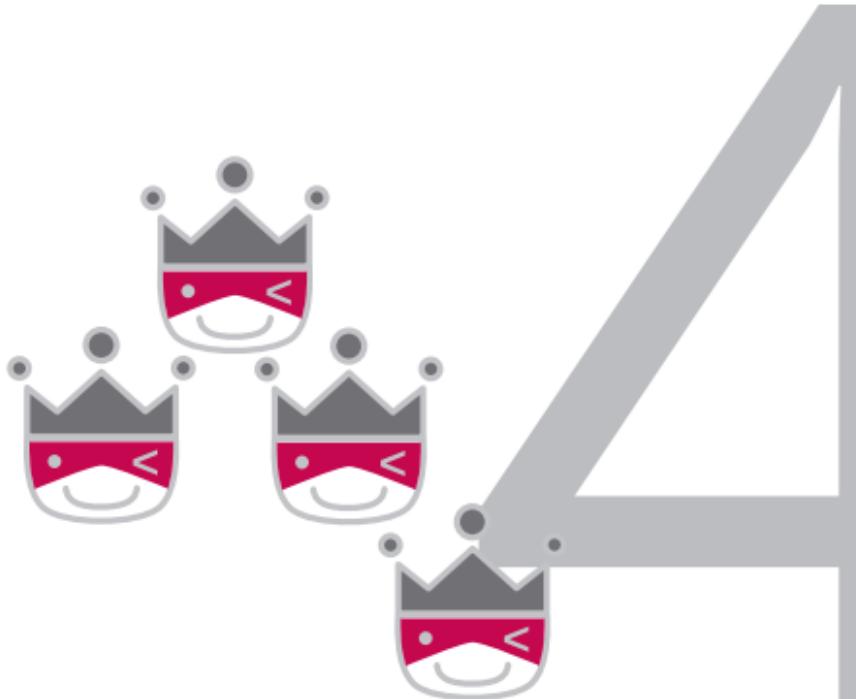
An attacker could try one credential after another and there's nothing to slow them down (online or offline)



# 4

# Spoofing

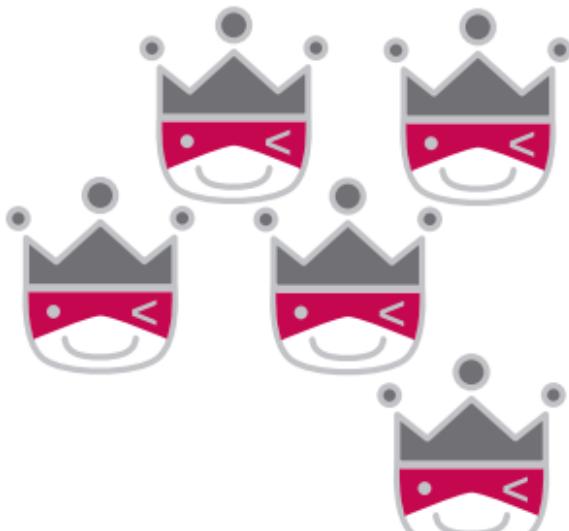
An attacker can anonymously connect, because we expect authentication to be done at a higher level



# 5

# Spoofing

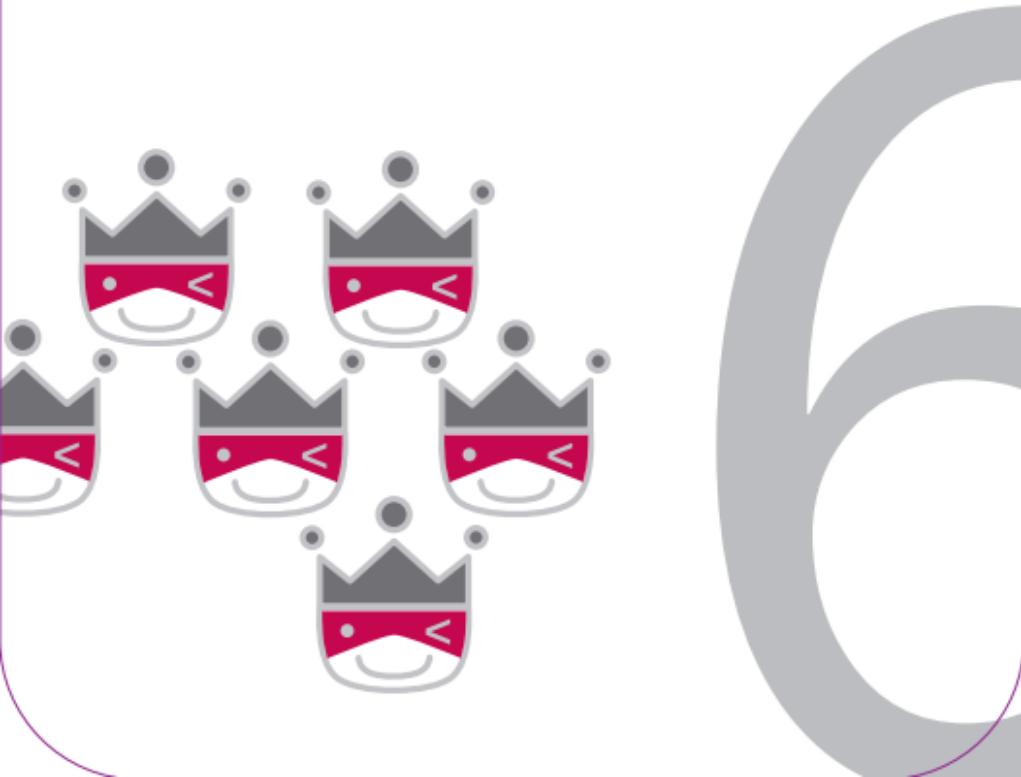
An attacker can confuse a client because there are too many ways to identify a server



# 6

## Spoofing

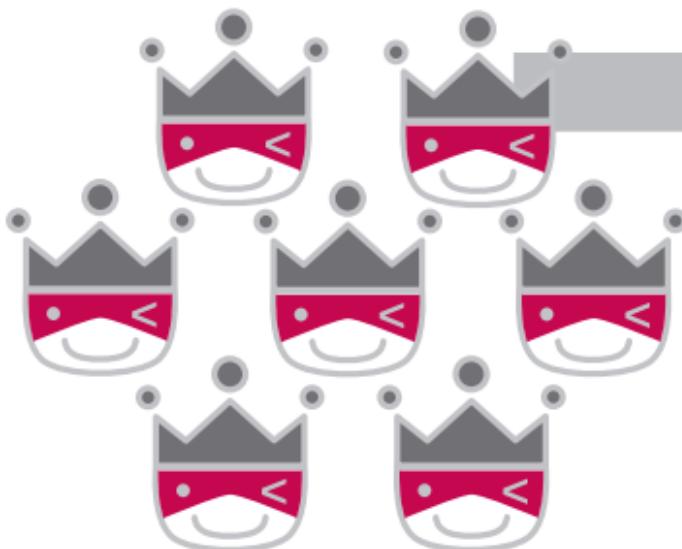
An attacker can spoof a server because identifiers aren't stored on the client and checked for consistency on re-connection (that is, there's no key persistence)



# 7

# Spoofing

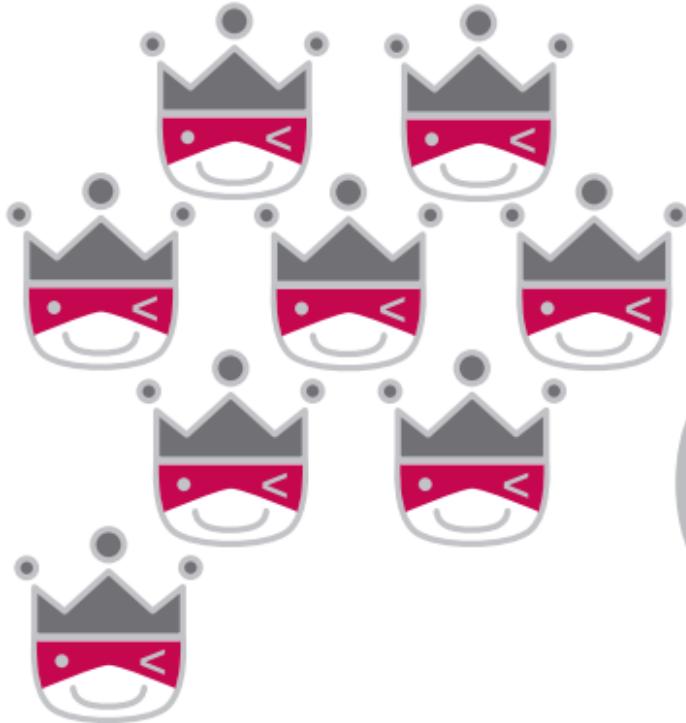
An attacker can connect to a server or peer over a link that isn't authenticated (and encrypted)



# 8

## Spoofing

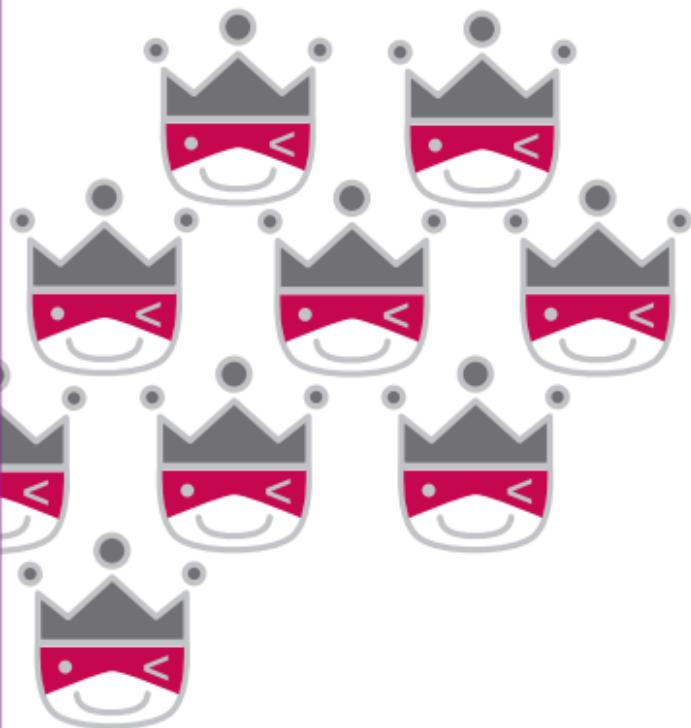
An attacker could steal credentials stored on the server and reuse them  
(for example, a key is stored in a world readable file)



# 9

# Spoofing

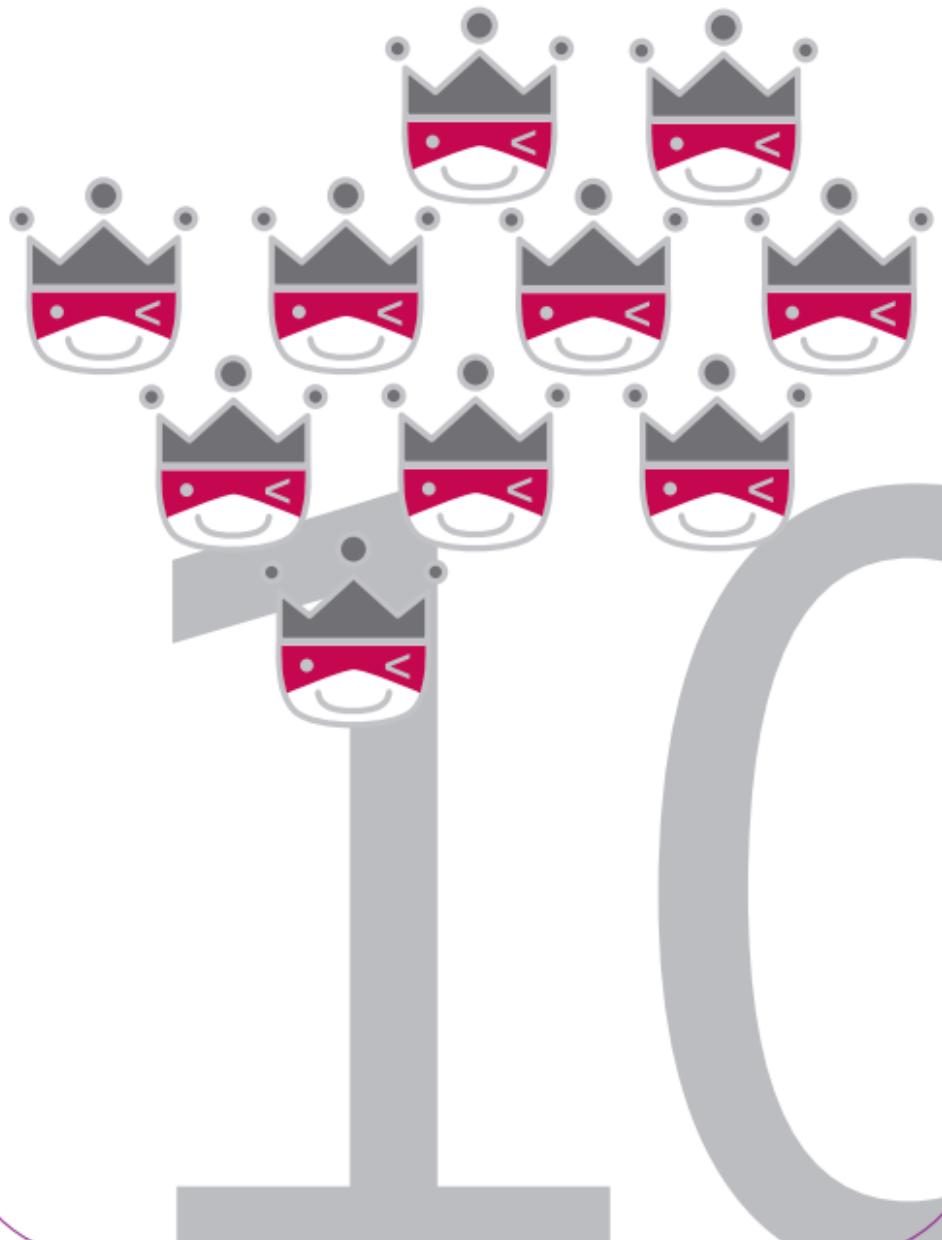
An attacker who gets a password can reuse it  
(Use stronger authenticators)



# 10

## Spoofing

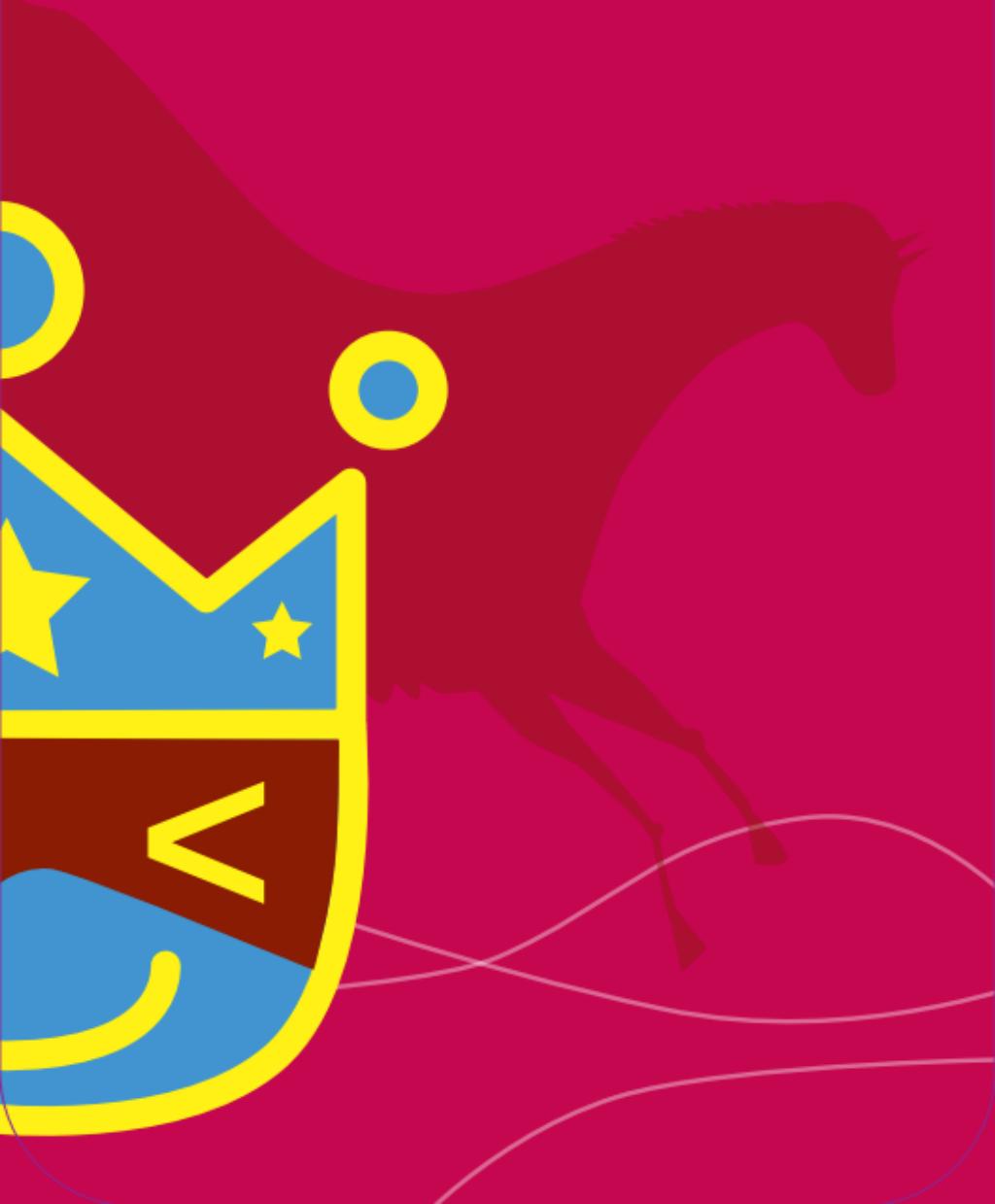
An attacker can choose to use weaker or no authentication



# J

## Spoofing

An attacker could steal credentials stored on the client and reuse them



# Q

## Spoofing

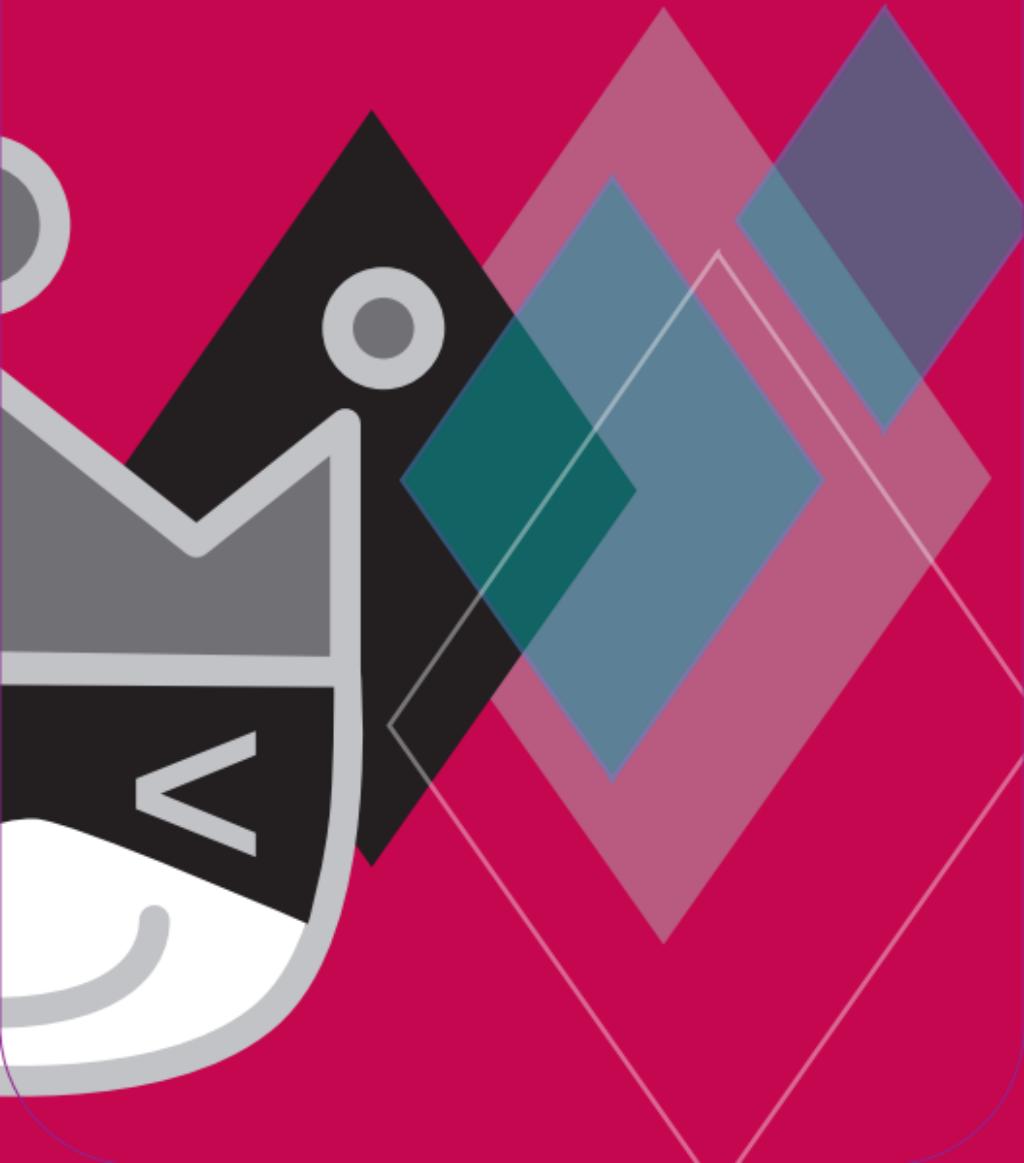
An attacker could go after the way credentials are updated or recovered (account recovery doesn't require disclosing the old password)



# K

# Spoofing

Your system ships with a default admin password, and doesn't force a change



# A

# Spoofing

You've invented a new  
Spoofing attack

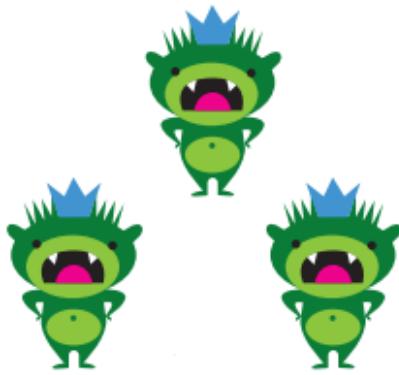


# A

# 3

## Tampering

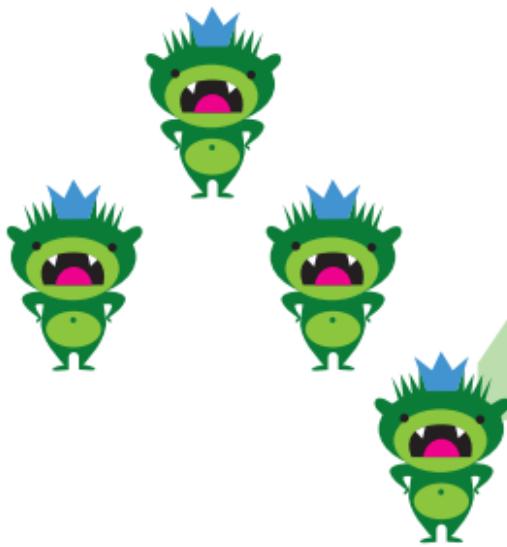
An attacker can take advantage of your custom key exchange or integrity control which you built instead of using standard crypto



# 4

## Tampering

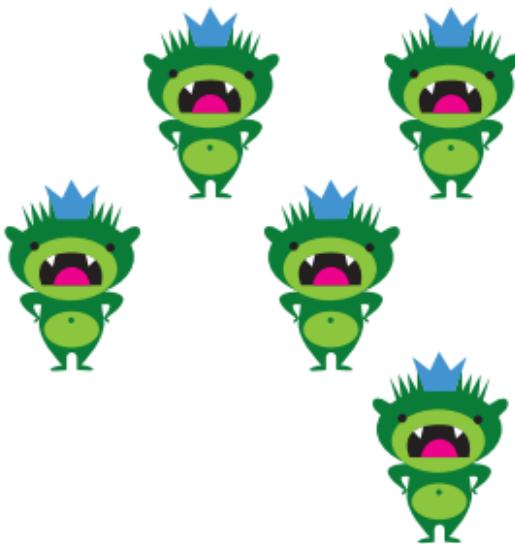
Your code makes access control decisions all over the place, rather than with a security kernel



# 5

## Tampering

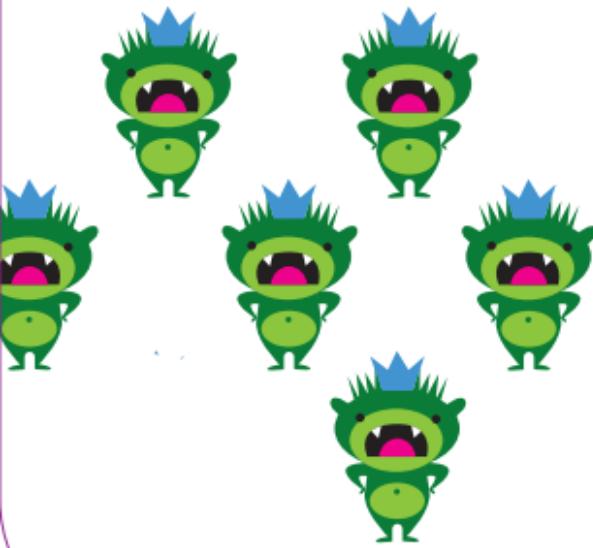
An attacker can replay data without detection because your code doesn't provide timestamps or sequence numbers



# 6

## Tampering

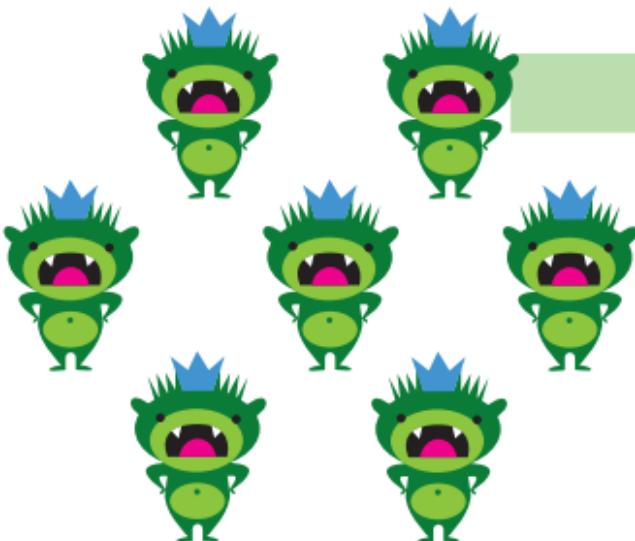
An attacker can write to a data store your code relies on



# 7

## Tampering

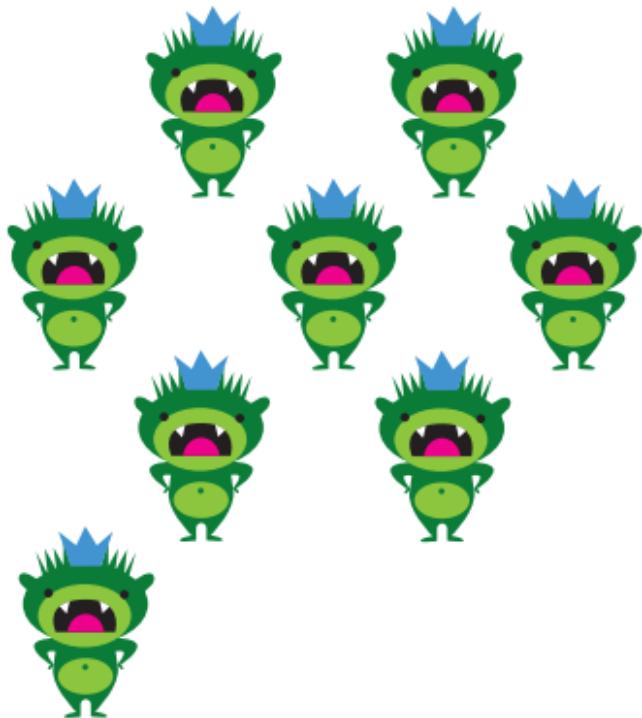
An attacker can bypass permissions because you don't make names canonical before checking access permissions



# 8

## Tampering

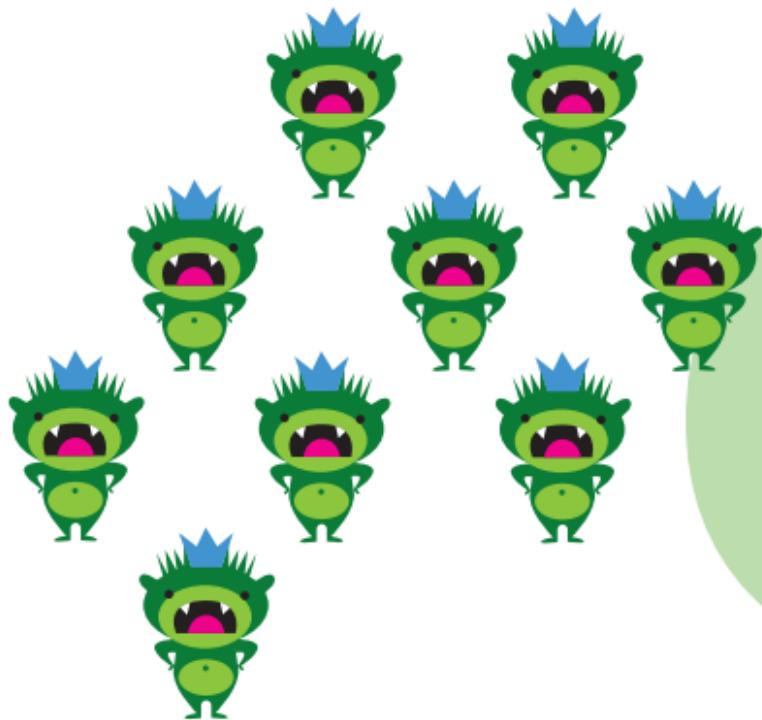
An attacker can manipulate data because there's no integrity protection for data on the network



# 9

# Tampering

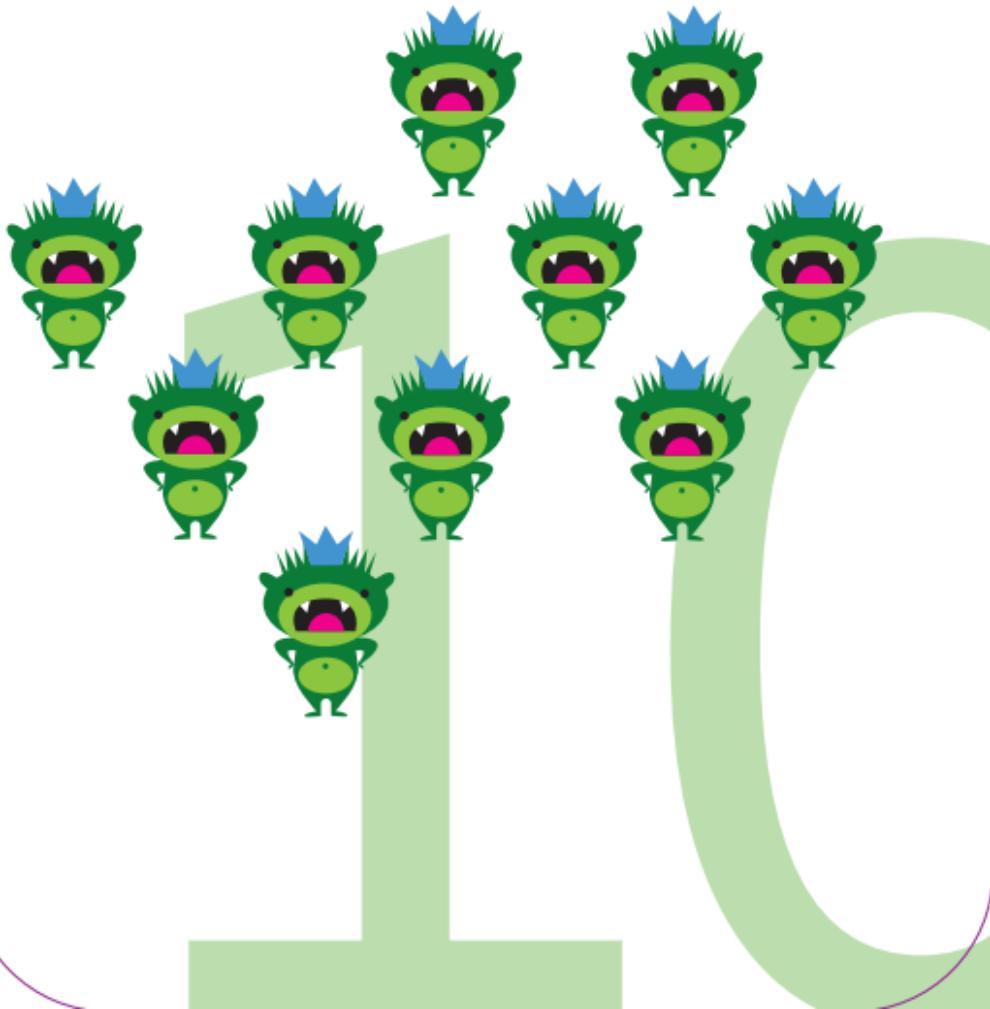
An attacker can provide or control state information



# 10

## Tampering

An attacker can alter information in a data store because it has weak ACLs or includes a group which is equivalent to everyone ("all Live ID holders")



# J

## Tampering

An attacker can write to some resource because permissions are granted to the world or there are no ACLs



# Q

## Tampering

An attacker can change parameters over a trust boundary and after validation (for example, important parameters in a hidden field in HTML, or passing a pointer to critical memory)



# K

## Tampering

An attacker can load code inside your process via an extension point



# A

## Tampering

You've invented a new  
Tampering attack



# 2

## Repudiation

An attacker can pass data through the log to attack a log reader, and there's no documentation of what sorts of validation are done

R  
R



# 3

# Repudiation

A low privilege attacker can  
read interesting security  
information in the logs

R  
R R

3

# 4

## Repudiation

An attacker can alter digital signatures because the digital signature system you're implementing is weak, or uses MACs where it should use a signature



# 5

## Repudiation

An attacker can alter log messages on a network because they lack strong integrity controls

R R  
R R R  
R



# 6

## Repudiation

An attacker can create a log entry without a timestamp  
(or no log entry is timestamped)

R R R  
R R R  
R



# 7

# Repudiation

An attacker can make the logs wrap around and lose data

R R  
R R R  
R R

# 8

## Repudiation

An attacker can make a log lose or confuse security information

R R  
R R R  
R R  
R

8

# 9

# Repudiation

An attacker can use a shared key to authenticate as different principals, confusing the information in the logs



R R  
R R R  
R R R  
R

# 10

## Repudiation

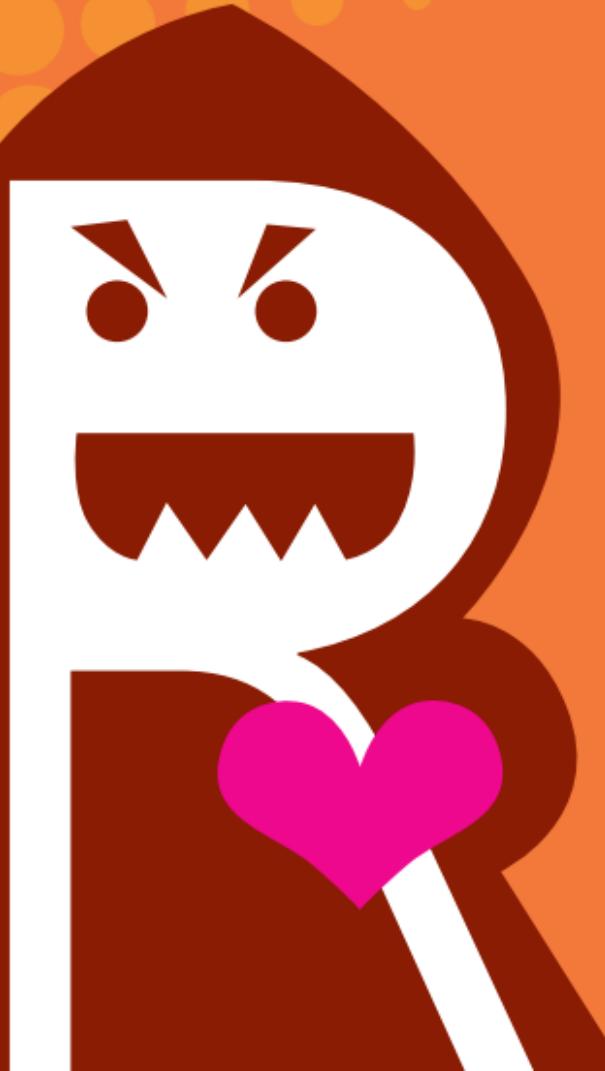
An attacker can get arbitrary data into logs from unauthenticated (or weakly authenticated) outsiders without validation



# J

## Repudiation

An attacker can edit logs and there's no way to tell (perhaps because there's no heartbeat option for the logging system)



# Q

## Repudiation

An attacker can say "I didn't do that," and you'd have no way to prove them wrong



I didn't  
do that.

# K

## Repudiation

The system has no logs

**logs = 0**

# A

# Repudiation

You've invented a new  
Repudiation attack

# R

# A

# 2

## Information Disclosure

An attacker can brute-force file encryption because there's no defense in place (example defense: password stretching)



# 3

## Information Disclosure

An attacker can see error messages with security sensitive content



# 4

## Information Disclosure

An attacker can read content because messages (say, an email or HTTP cookie) aren't encrypted even if the channel is encrypted



# 5

## Information Disclosure

An attacker may be able to read a document or data because it's encrypted with a non-standard algorithm



# 6

## Information Disclosure

An attacker can read data because it's hidden or occluded (for undo or change tracking) and the user might forget that it's there



# 7

# Information Disclosure

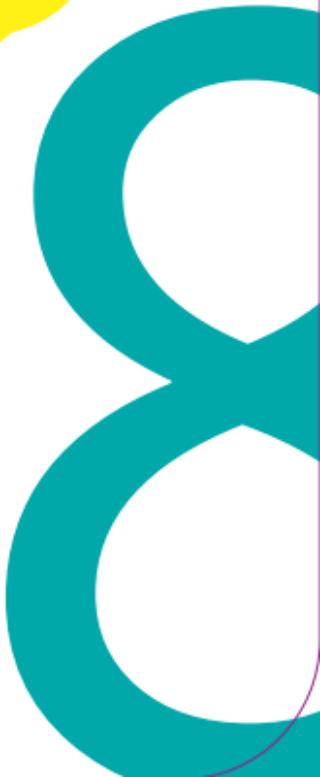
An attacker can act as a 'man in the middle' because you don't authenticate endpoints of a network connection



# 8

## Information Disclosure

An attacker can access information through a search indexer, logger, or other such mechanism



# 9

# Information Disclosure

An attacker can read sensitive information in a file with bad ACLs



# 10

## Information Disclosure

An attacker can read information in files with no ACLs



# J

## Information Disclosure

An attacker can discover the fixed key being used to encrypt



Found it!



# Information Disclosure

An attacker can read the entire channel because the channel (say, HTTP or SMTP) isn't encrypted

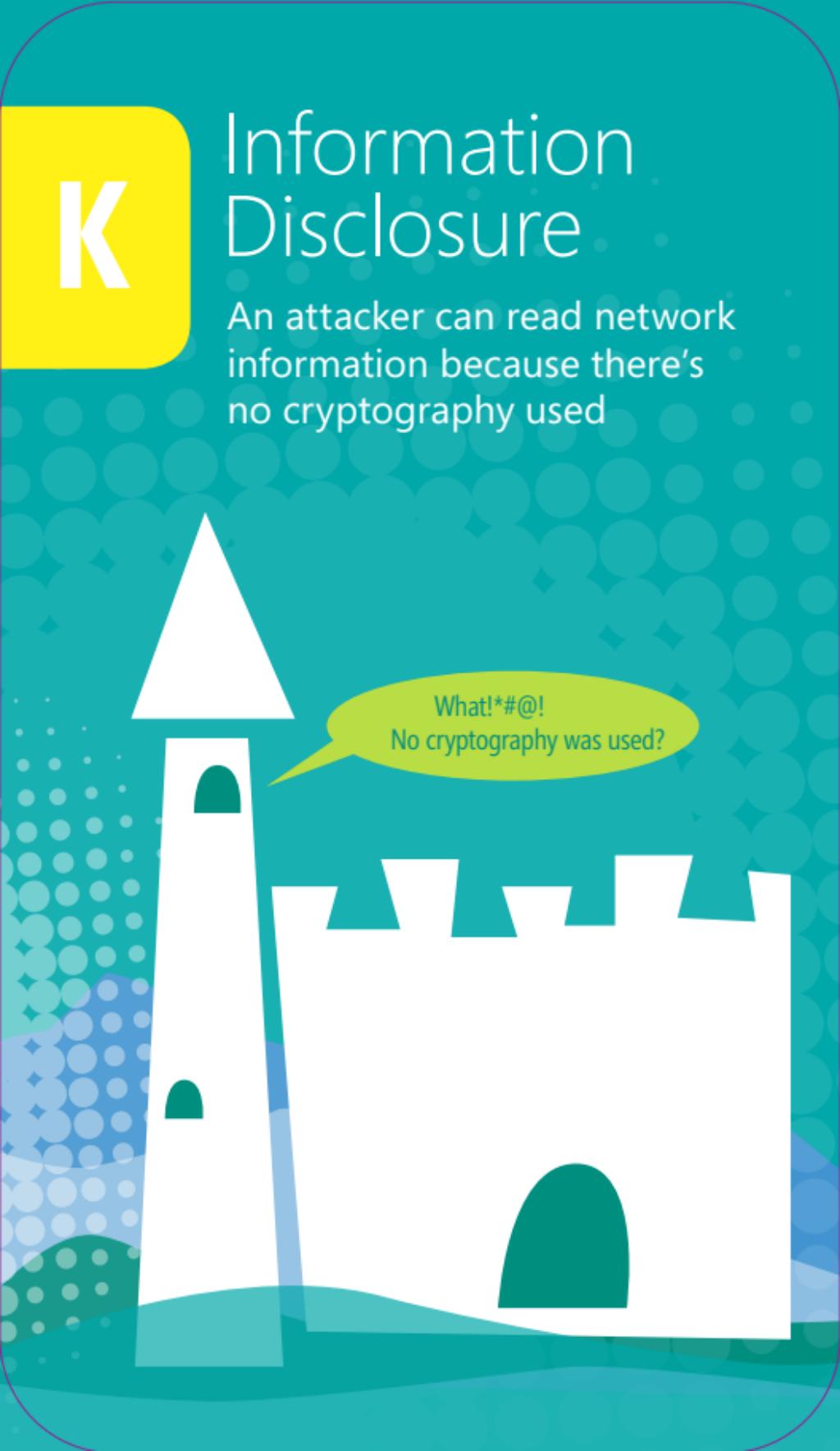
Don't tell anyone, but...



# K

## Information Disclosure

An attacker can read network information because there's no cryptography used



What!\*#@!  
No cryptography was used?

# A

# Information Disclosure

You've invented a new  
Information Disclosure attack



# 2

## Denial of Service

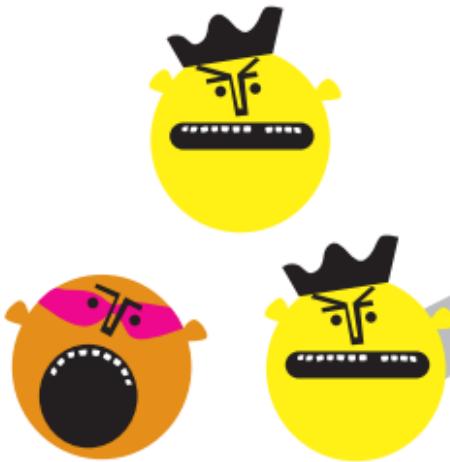
An attacker can make your authentication system unusable or unavailable



# 3

## Denial of Service

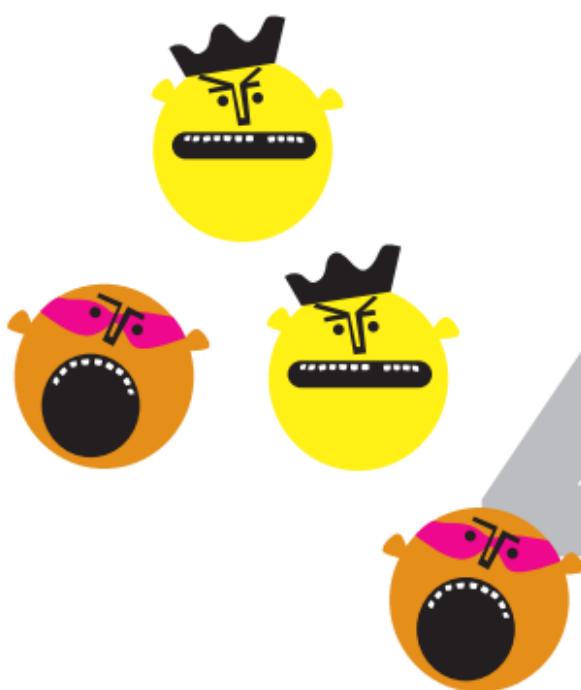
An attacker can make a client unavailable or unusable but the problem goes away when the attacker stops



# 4

## Denial of Service

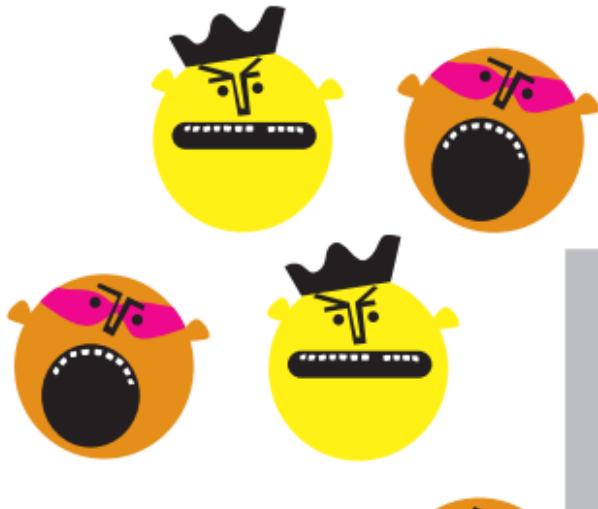
An attacker can make a server unavailable or unusable but the problem goes away when the attacker stops



# 5

# Denial of Service

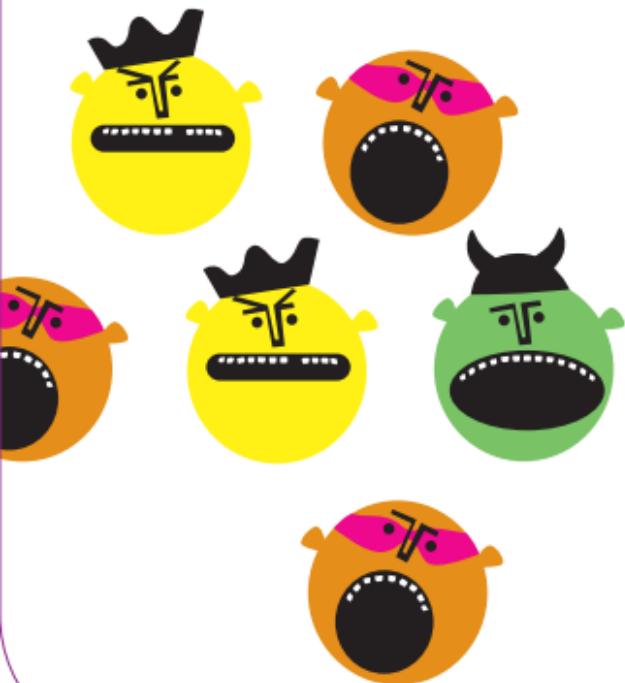
An attacker can make a client unavailable or unusable without ever authenticating but the problem goes away when the attacker stops



# 6

# Denial of Service

An attacker can make a server unavailable or unusable without ever authenticating but the problem goes away when the attacker stops



# 7

# Denial of Service

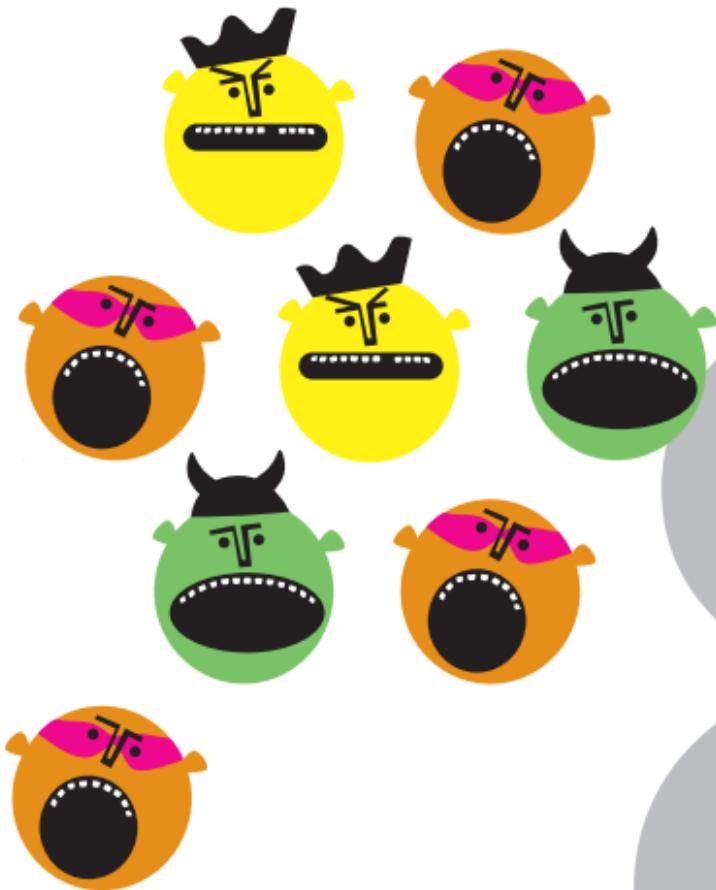
An attacker can make a client unavailable or unusable and the problem persists after the attacker goes away



# 8

# Denial of Service

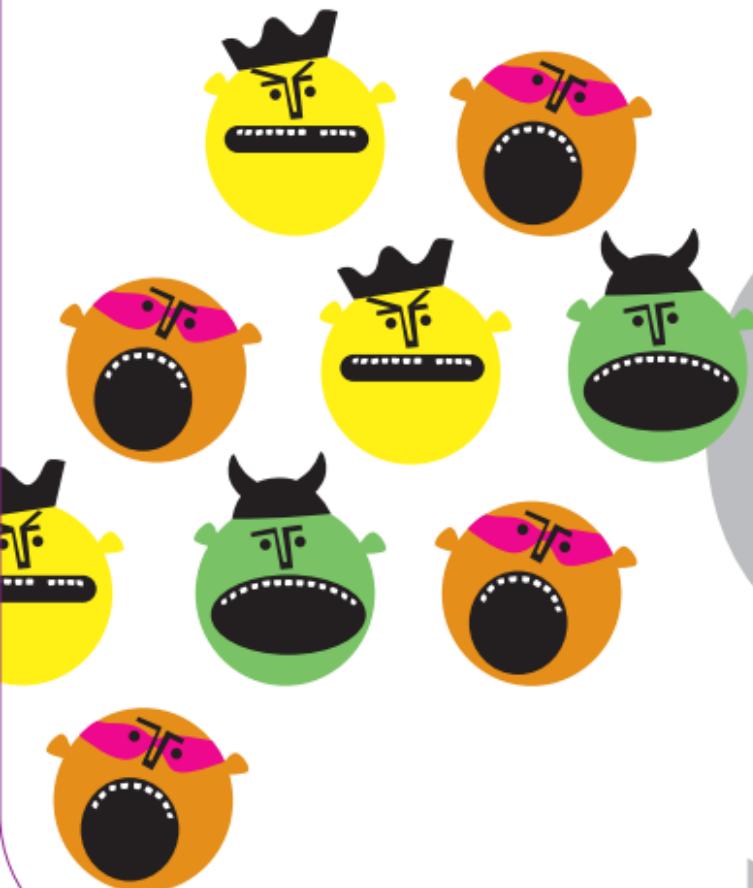
An attacker can make a server unavailable or unusable and the problem persists after the attacker goes away



# 9

# Denial of Service

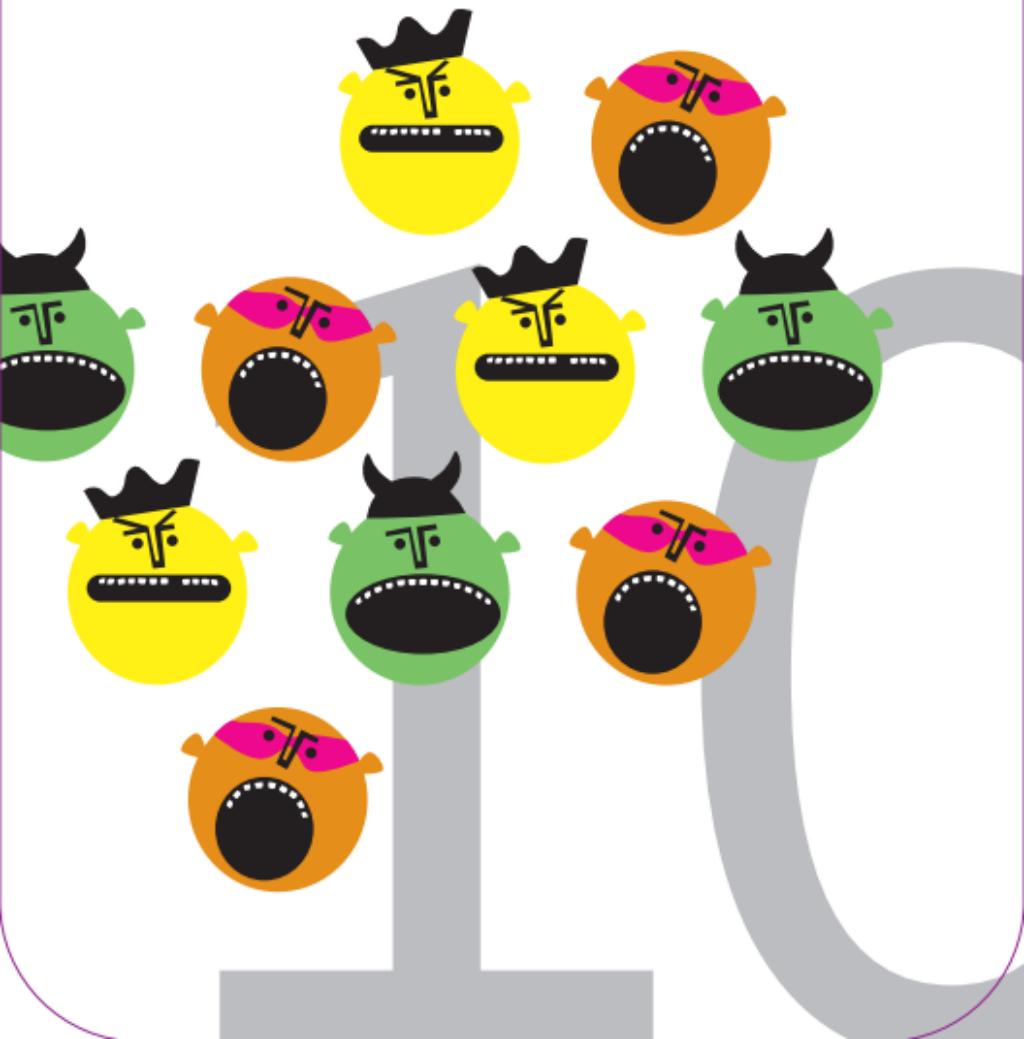
An attacker can make a client unavailable or unusable without ever authenticating and the problem persists after the attacker goes away



# 10

## Denial of Service

An attacker can make a server unavailable or unusable without ever authenticating and the problem persists after the attacker goes away



# J

# Denial of Service

An attacker can cause the logging subsystem to stop working



# Q

# Denial of Service

An attacker can amplify a Denial of Service attack through this component with amplification on the order of 10:1



# K

# Denial of Service

An attacker can amplify a Denial of Service attack through this component with amplification on the order of 100:1



# A

# Denial of Service

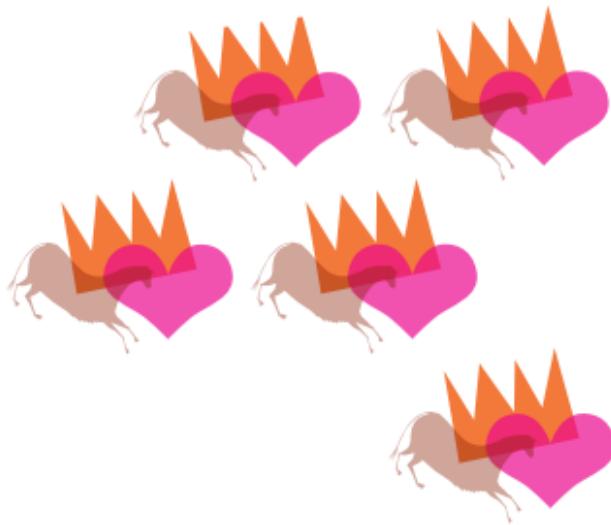
You've invented a new  
Denial of Service attack



# 5

# Elevation of Privilege

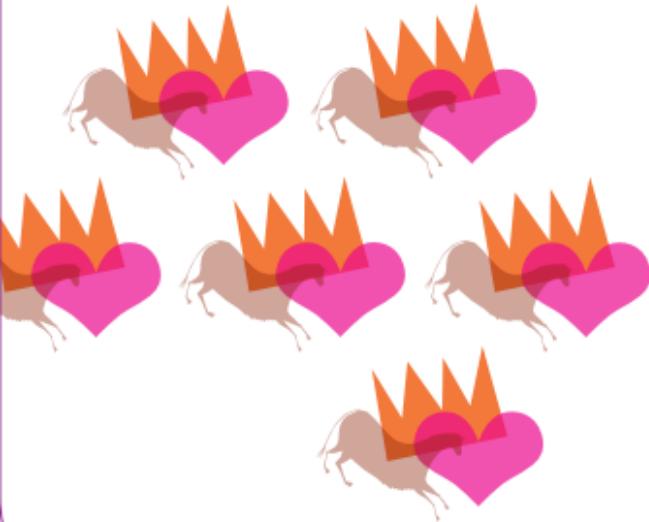
An attacker can force data through different validation paths which give different results



# 6

# Elevation of Privilege

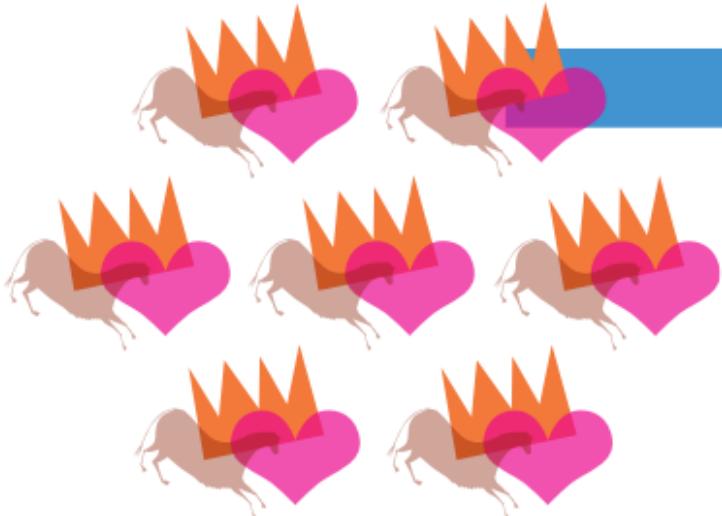
An attacker could take advantage of .NET permissions you ask for, but don't use



# 7

# Elevation of Privilege

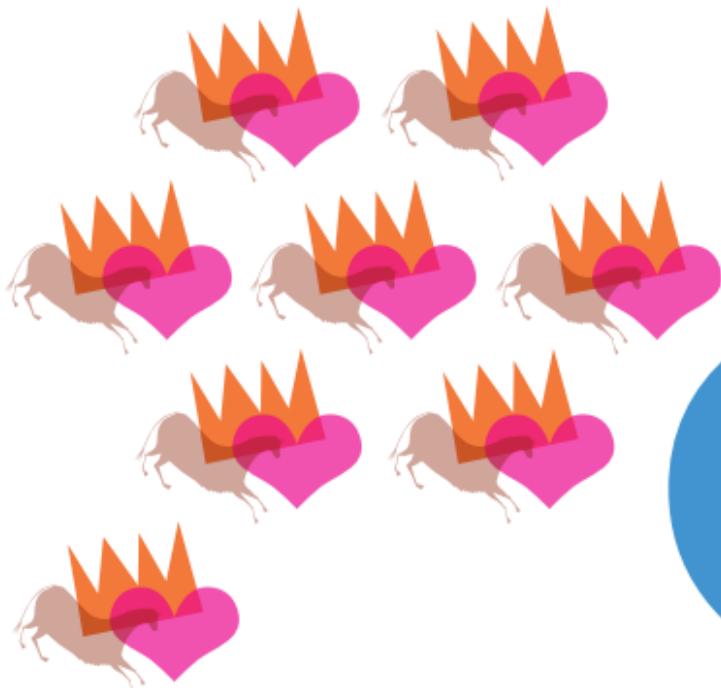
An attacker can provide a pointer across a trust boundary, rather than data which can be validated



# 8

# Elevation of Privilege

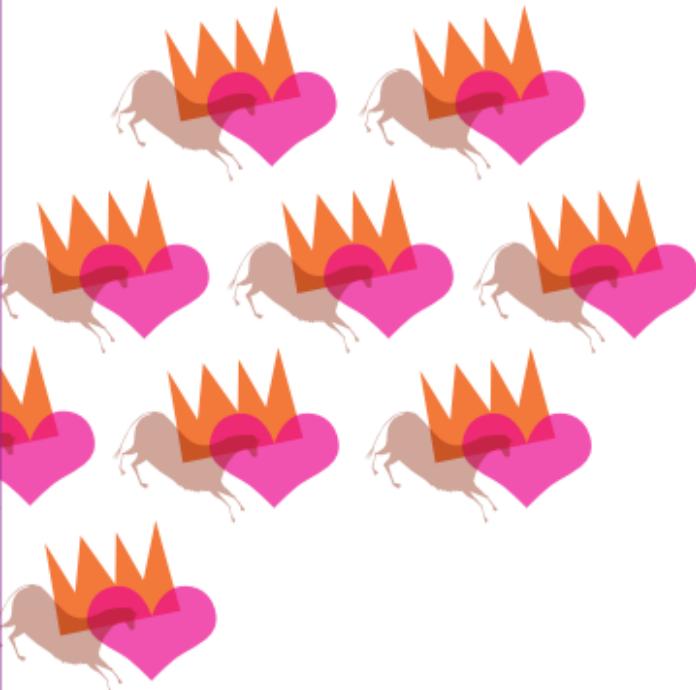
An attacker can enter data that is checked while still under their control and used later on the other side of a trust boundary



# 9

# Elevation of Privilege

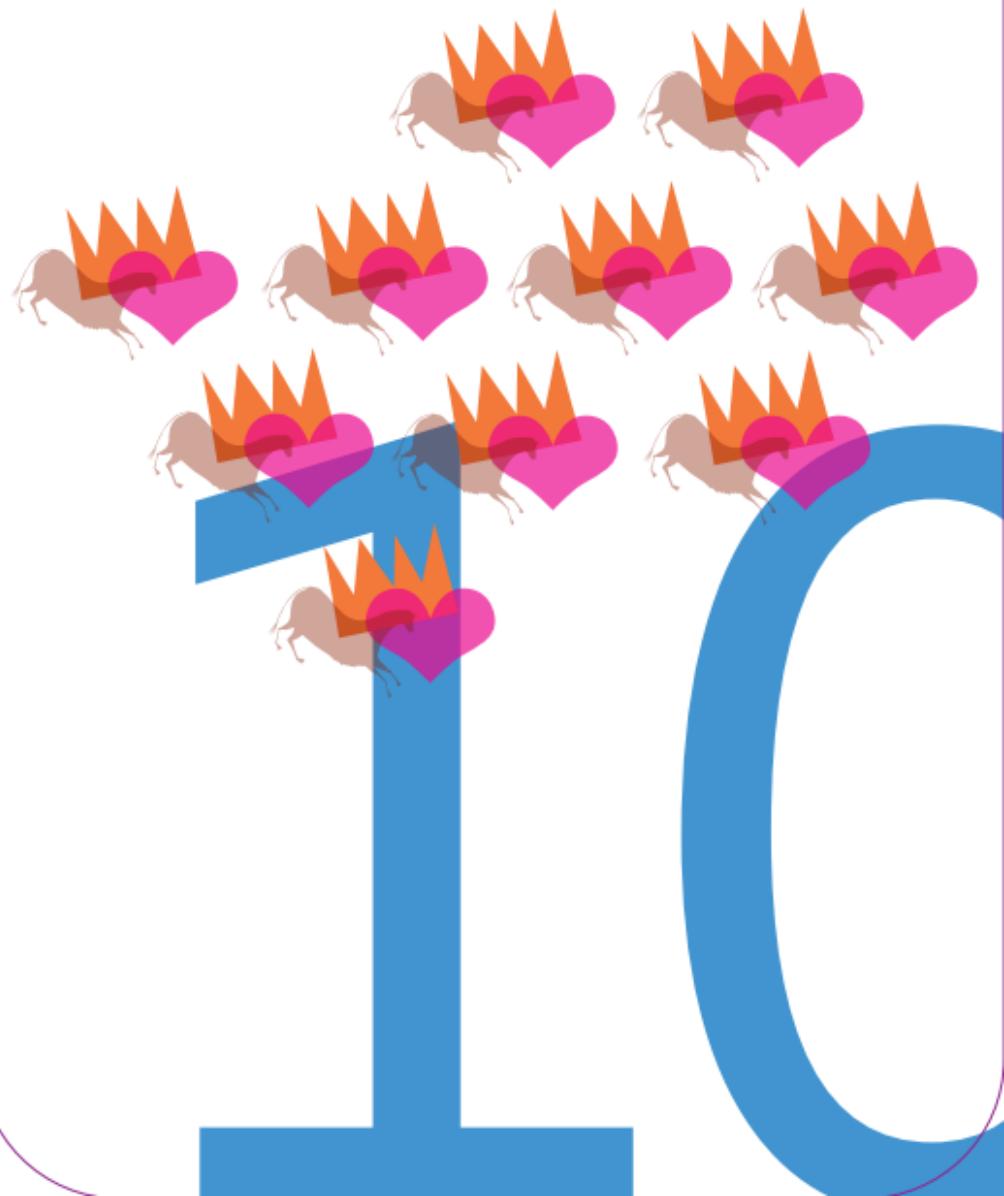
There's no reasonable way for a caller to figure out what validation of tainted data you perform before passing it to them



# 10

## Elevation of Privilege

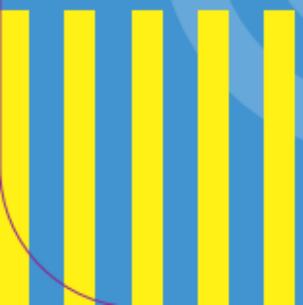
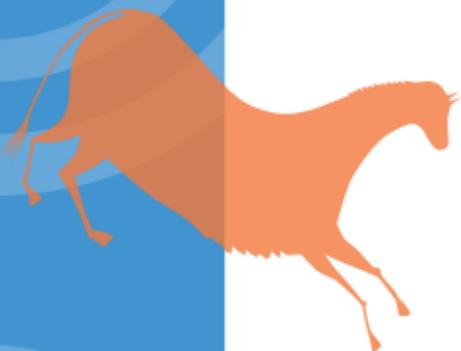
There's no reasonable way for a caller to figure out what security assumptions you make



J

# Elevation of Privilege

An attacker can reflect input back to a user, like cross site scripting



# Q

## Elevation of Privilege

You include user-generated content within your page, possibly including the content of random URLs



# K

# Elevation of Privilege

An attacker can inject a command that the system will run at a higher privilege level



# A

# Elevation of Privilege

You've invented a new  
Elevation of Privilege attack





## Spoofing

2. An attacker could squat on the random port or socket that the server normally uses
3. An attacker could try one credential after another and there's nothing to slow them down (online or offline)
4. An attacker can anonymously connect because we expect authentication to be done at a higher level
5. An attacker can confuse a client because there are too many ways to identify a server
6. An attacker can spoof a server because identifiers aren't stored on the client and checked for consistency on re-connection (that is, there's no key persistence)
7. An attacker can connect to a server or peer over a link that isn't authenticated (and encrypted)
8. An attacker could steal credentials stored on the server and reuse them (for example, a key is stored in a world readable file)
9. An attacker who gets a password can reuse it (Use stronger authenticators)
10. An attacker can choose to use weaker or no authentication

continued on back

# Spoofing



## Spoofing cont.

- J. An attacker could steal credentials stored on the client and reuse them
- Q. An attacker could go after the way credentials are updated or recovered (account recovery doesn't require disclosing the old password)
- K. Your system ships with a default admin password, and doesn't force a change
- A. You've invented a new Spoofing attack

# Spoofing



## Tampering

3. An attacker can take advantage of your custom key exchange or integrity control which you built instead of using standard crypto
  4. Your code makes access control decisions all over the place, rather than with a security kernel
  5. An attacker can replay data without detection because your code doesn't provide timestamps or sequence numbers
  6. An attacker can write to a data store your code relies on
  7. An attacker can bypass permissions because you don't make names canonical before checking access permissions
  8. An attacker can manipulate data because there's no integrity protection for data on the network
  9. An attacker can provide or control state information
  10. An attacker can alter information in a data store because it has weak ACLs or includes a group which is equivalent to everyone ("all Live ID holders")
  - J. An attacker can write to some resource because permissions are granted to the world or there are no ACLs
- continued on back

# Tampering



## Tampering cont.

Q. An attacker can change parameters over a trust boundary and after validation (for example, important parameters in a hidden field in HTML, or passing a pointer to critical memory)

K. An attacker can load code inside your process via an extension point

A. You've invented a new Tampering attack

# Tampering

# R

## Repudiation

2. An attacker can pass data through the log to attack a log reader, and there's no documentation of what sorts of validation are done
3. A low privilege attacker can read interesting security information in the logs
4. An attacker can alter digital signatures because the digital signature system you're implementing is weak, or uses MACs where it should use a signature
5. An attacker can alter log messages on a network because they lack strong integrity controls
6. An attacker can create a log entry without a timestamp (or no log entry is timestamped)
7. An attacker can make the logs wrap around and lose data
8. An attacker can make a log lose or confuse security information
9. An attacker can use a shared key to authenticate as different principals, confusing the information in the logs
10. An attacker can get arbitrary data into logs from unauthenticated (or weakly authenticated) outsiders without validation

continued on back

# Repudiation

# R

## Repudiation cont.

- 10. An attacker can get arbitrary data into logs from unauthenticated (or weakly authenticated) outsiders without validation
- J. An attacker can edit logs and there's no way to tell (perhaps because there's no heartbeat option for the logging system)
- Q. An attacker can say "I didn't do that," and you'd have no way to prove them wrong
- K. The system has no logs
- A. You've invented a new Repudiation attack

# Repudiation



## Information Disclosure

2. An attacker can brute-force file encryption because there's no defense in place (example defense: password stretching)
  3. An attacker can see error messages with security-sensitive content
  4. An attacker can read content because messages (say, an email or HTTP cookie) aren't encrypted even if the channel is encrypted
  5. An attacker may be able to read a document or data because it's encrypted with a non-standard algorithm
  6. An attacker can read data because it's hidden or occluded (for undo or change tracking) and the user might forget that it's there
  7. An attacker can act as a 'man in the middle' because you don't authenticate endpoints of a network connection
  8. An attacker can access information through a search indexer, logger, or other such mechanism
  9. An attacker can read sensitive information in a file with bad ACLs
  10. An attacker can read information in files with no ACLs
- continued on back

# Information Disclosure



## Information Disclosure cont.

- J. An attacker can discover the fixed key being used to encrypt
- Q. An attacker can read the entire channel because the channel (say, HTTP or SMTP) isn't encrypted
- K. An attacker can read network information because there's no cryptography used
- A. You've invented a new Information Disclosure attack

# Information Disclosure



## Denial of Service

2. An attacker can make your authentication system unusable or unavailable
3. An attacker can make a client unavailable or unusable but the problem goes away when the attacker stops  
**(client, authenticated, temporary)**
4. An attacker can make a server unavailable or unusable but the problem goes away when the attacker stops  
**(server, authenticated, temporary)**
5. An attacker can make a client unavailable or unusable without ever authenticating but the problem goes away when the attacker stops  
**(client, anonymous, temporary)**
6. An attacker can make a server unavailable or unusable without ever authenticating but the problem goes away when the attacker stops  
**(server, anonymous, temporary)**
7. An attacker can make a client unavailable or unusable and the problem persists after the attacker goes away  
**(client, authenticated, persistent)**
8. An attacker can make a server unavailable or unusable and the problem persists after the attacker goes away  
**(server, authenticated, persistent)**
9. An attacker can make a client unavailable or unusable without ever authenticating and the problem persists after the attacker goes away  
**(client, anonymous, persistent)**

continued on back

## Denial of Service



## Denial of Service cont.

10. An attacker can make a server unavailable or unusable without ever authenticating and the problem persists after the attacker goes away (**server, anonymous, persistent**)

J. An attacker can cause the logging subsystem to stop working

Q. An attacker can amplify a Denial of Service attack through this component with amplification on the order of 10:1

K. An attacker can amplify a Denial of Service attack through this component with amplification on the order of 100:1

A. You've invented a new Denial of Service attack

# Denial of Service



## Elevation of Privilege (EoP)

5. An attacker can force data through different validation paths which give different results
  6. An attacker could take advantage of .NET permissions you ask for, but don't use
  7. An attacker can provide a pointer across a trust boundary, rather than data which can be validated
  8. An attacker can enter data that is checked while still under their control and used later on the other side of a trust boundary
  9. There's no reasonable way for a caller to figure out what validation of tainted data you perform before passing it to them
  10. There's no reasonable way for a caller to figure out what security assumptions you make
- J. An attacker can reflect input back to a user, like cross site scripting
- Q. You include user-generated content within your page, possibly including the content of random URLs
- K. An attacker can inject a command that the system will run at a higher privilege level
- A. You've invented a new Elevation of Privilege attack

# Elevation of Privilege



# Elevation of Privilege

# About

## Threat Modeling

Elevation of Privilege is designed to be the easiest way to start looking at your design from a security perspective. It's one way to threat model, intended to be picked up and used by any development group. Because it uses the STRIDE threats, it gives you a framework for thinking, and specific actionable examples of those threats.

STRIDE stands for:

**Spoofing** Impersonating something or someone else.

**Tampering** Modifying data or code

**Repudiation** Claiming to have not performed an action

**Information Disclosure** Exposing information to someone not authorized to see it

**Denial of Service** Deny or degrade service to users

**Elevation of Privilege** Gain capabilities without proper authorization

At [www.microsoft.com/security/sdl/eop](http://www.microsoft.com/security/sdl/eop) we have resources for you including videos, score sheets and tips and tricks for playing.

About



# SDL

Elevation of Privilege is a fun and easy way to get started understanding the security of your systems by threat modeling. As you discover and correct design-level security problems, it's worth thinking about the other ways security issues can creep into your code. Microsoft has a large collection of free resources available to help you get started with the Security Development Lifecycle (SDL).

To learn more about threat modeling and the Microsoft Security Development Lifecycle, visit our website at [microsoft.com/sdl/](http://microsoft.com/sdl/)

**Microsoft®**

**Security Development Lifecycle**