Anleitung 1

Elevation of Privilege Anleitung

1. Zeichnen Sie ein Diagramm des Systems, für das Sie ein Bedrohungsmodell erstellen möchten.
2. Teilen Sie die Karten an 3 – 6 Spieler aus.
3. Das Spiel beginnt mit der Karte "Tampering 3".

Es wird im Uhrzeigersinn gespielt. Wer an der Reihe ist, versucht die Suite (z.B. Tampering) zu bedienen, mit der die Runde eingeleitet wurde. Wer nicht bedienen kann, spielt eine Karte einer anderen Suite. Wer mit der höchstwertigen Karte bedienen kann, gewinnt die Runde, außer eine "Elevation of Privilege" Karte wird gespielt, die dann gewinnt. Um eine Karte zu spielen, lesen Sie die Bedrohung auf der Karte laut vor. Können Sie die Bedrohung nicht auf Ihr System anwenden, geht das Spiel weiter. Wer die Runde gewinnt, wählt Karte (und Suite), mit der die nächste Runde begonnen wird. Machen Sie nach jeder Runde eine kurze Pause, um über die Bedrohungen nachzudenken.

Punkte:

1 wenn Sie die Bedrohung auf Ihr System anwenden können,
+1 für den Rundengewinn

Anleitung

Anleitung 2

Elevation of Privilege Varianten

Optionale Varianten:

a) Sie können Karten, die Sie nicht auf das System anwenden können, nach der 3. Runde abgeben. Vielleicht kann es jemand anders .

b) Verdoppeln Sie die Punktezahlen und geben Sie 1 Punkt für Bedrohungen auf Karten anderer Spieler.

c) Spieler die nicht an der Reihe sind können nach Spielen einer Karte binnen 1 Minute Varianten einer Bedrohung einwerfen und erläutern, die auf das System anwendbar sind. Geben Sie hierfür 1 Extrapunkt. Markieren Sie im Diagramm, wo die Bedrohung statt findet.

Thanks to Laurie Williams for inspiration.

Context

- Regeln erklären
- Karten austeilen & Spiel beginnen
- Spielen
- Wrap up

Play

- Wer Tampering 3 hat, beginnt

Wer Tampering 3 hat:

- Karte spielen
- Bedrohung dokumentieren
- Hat jeder in dieser Runde gespielt?

falls nein:
nächster Spieler
im Uhrzeigersinn
spielt

falls ja:
nächste Runde
wird vom Gewinner
der vorherigen Runde
eingeleitet

(3 Min. Nachdenkzeit
nach jeder Runde
kann den Spielfluss
verbessern)

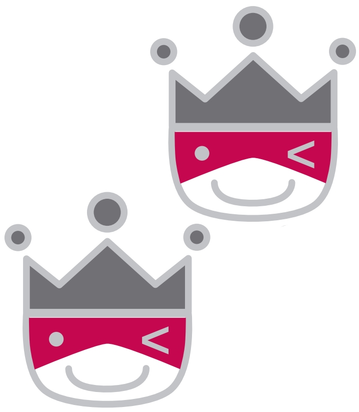
ja

- Ergebnisse notieren
- Zeit für eine weitere Runde?
- Wrap up

2

Spoofing

Ein Angreifer "sitzt" (lauscht) auf einem zufälligen Port oder Socket, den der Server üblicherweise nutzte.

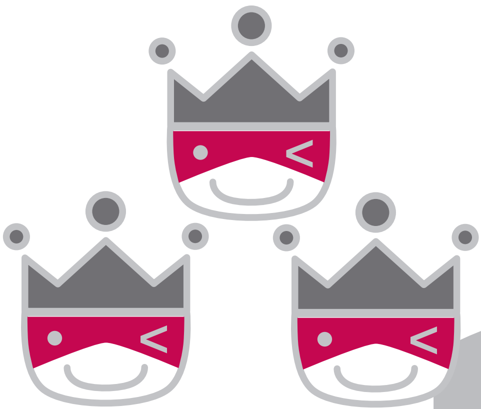


2

3

Spoofing

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

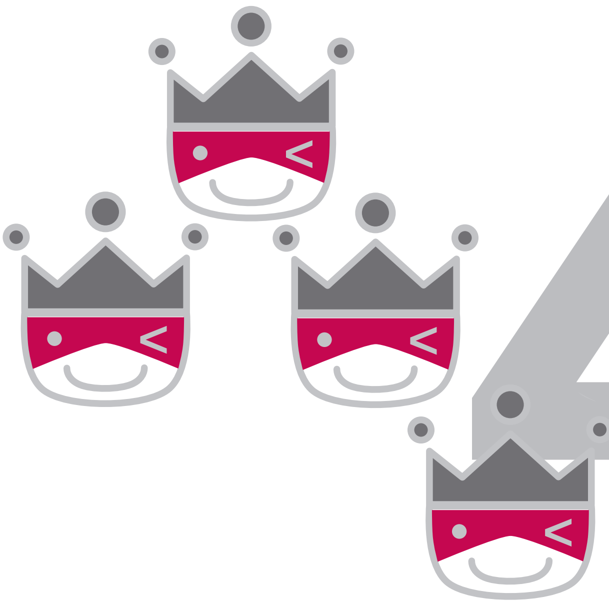


3

4

Spoofing

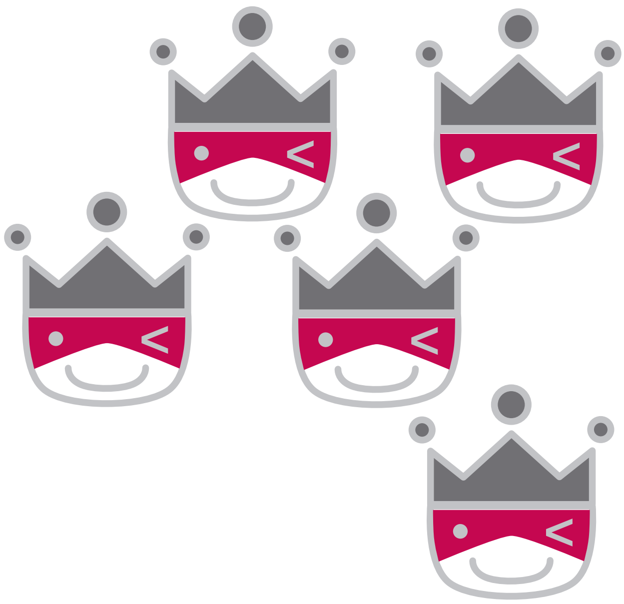
An attacker can anonymously connect, because we expect authentication to be done at a higher level.



5

Spoofing

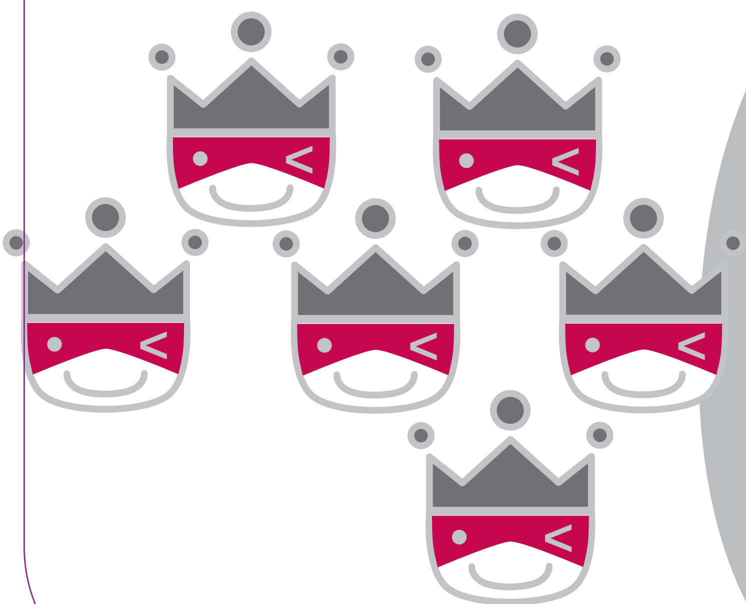
An attacker can confuse a client because there are too many ways to identify a server.



5

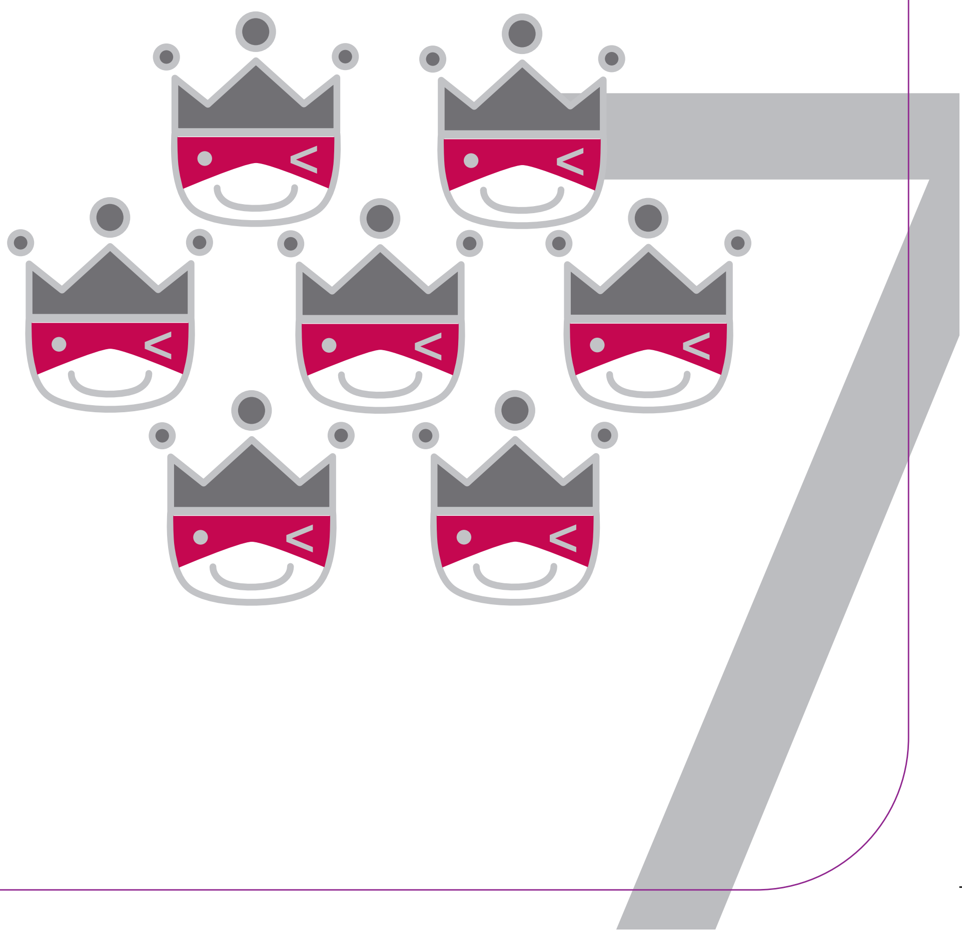
6 Spoofing

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).



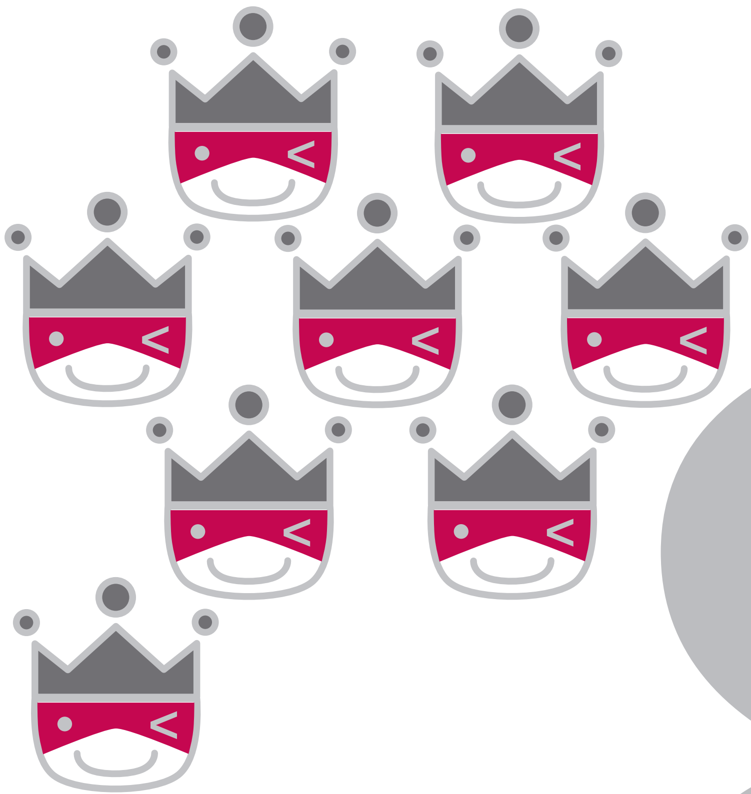
7 Spoofing

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).

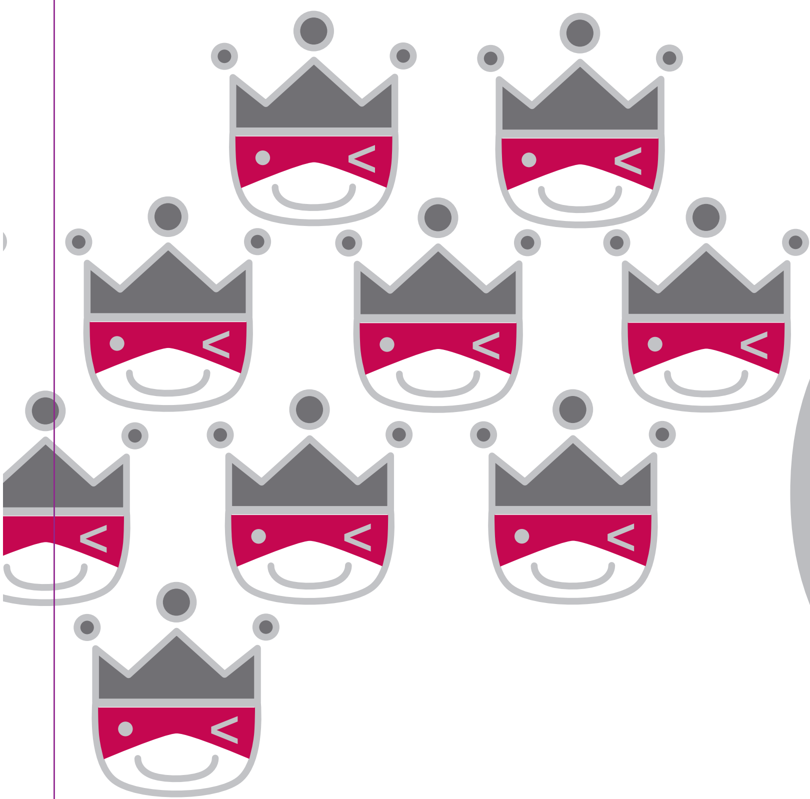


8

9

Spoofing

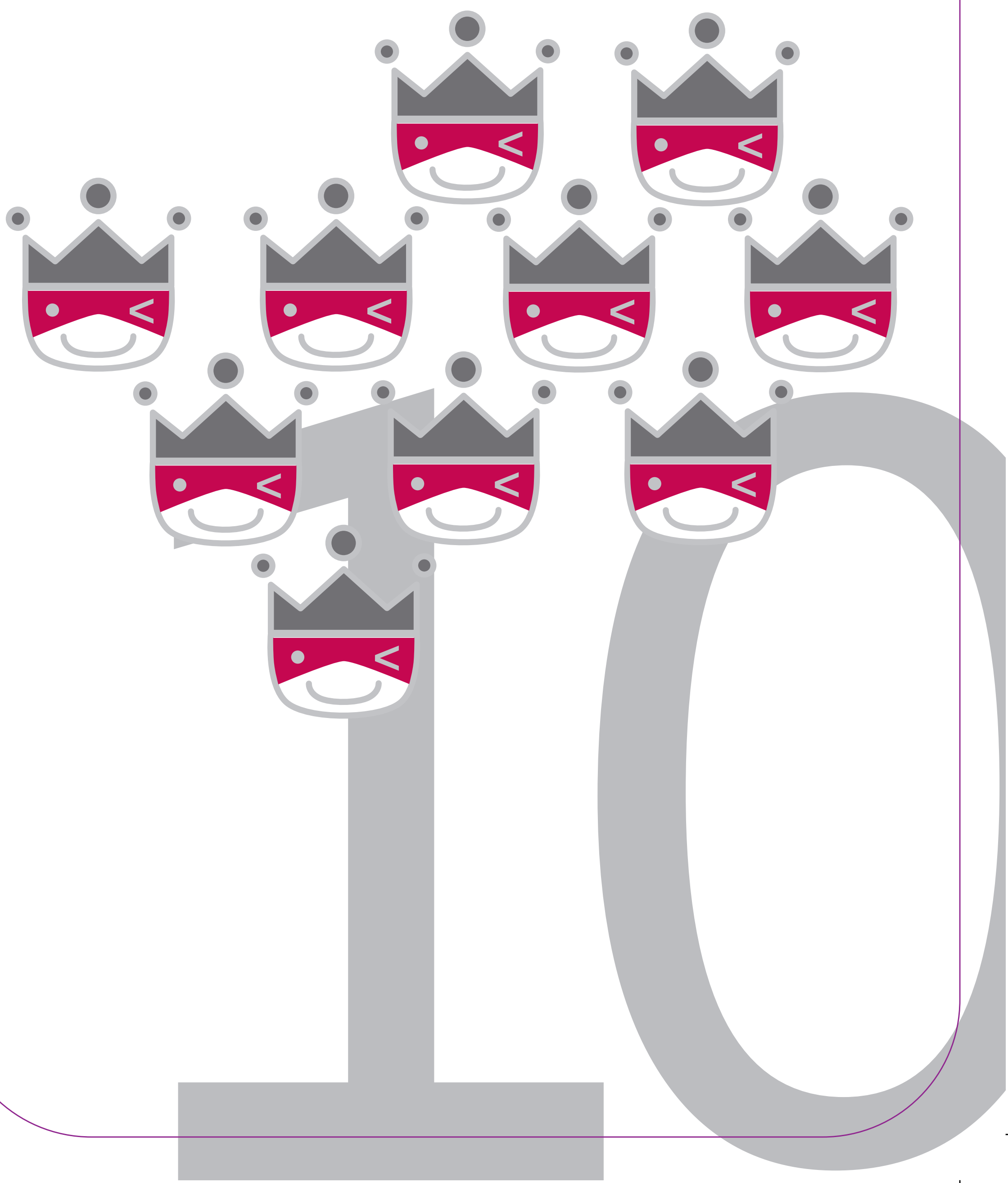
An attacker who gets a password can reuse it (use stronger authenticators).



9

10 Spoofing

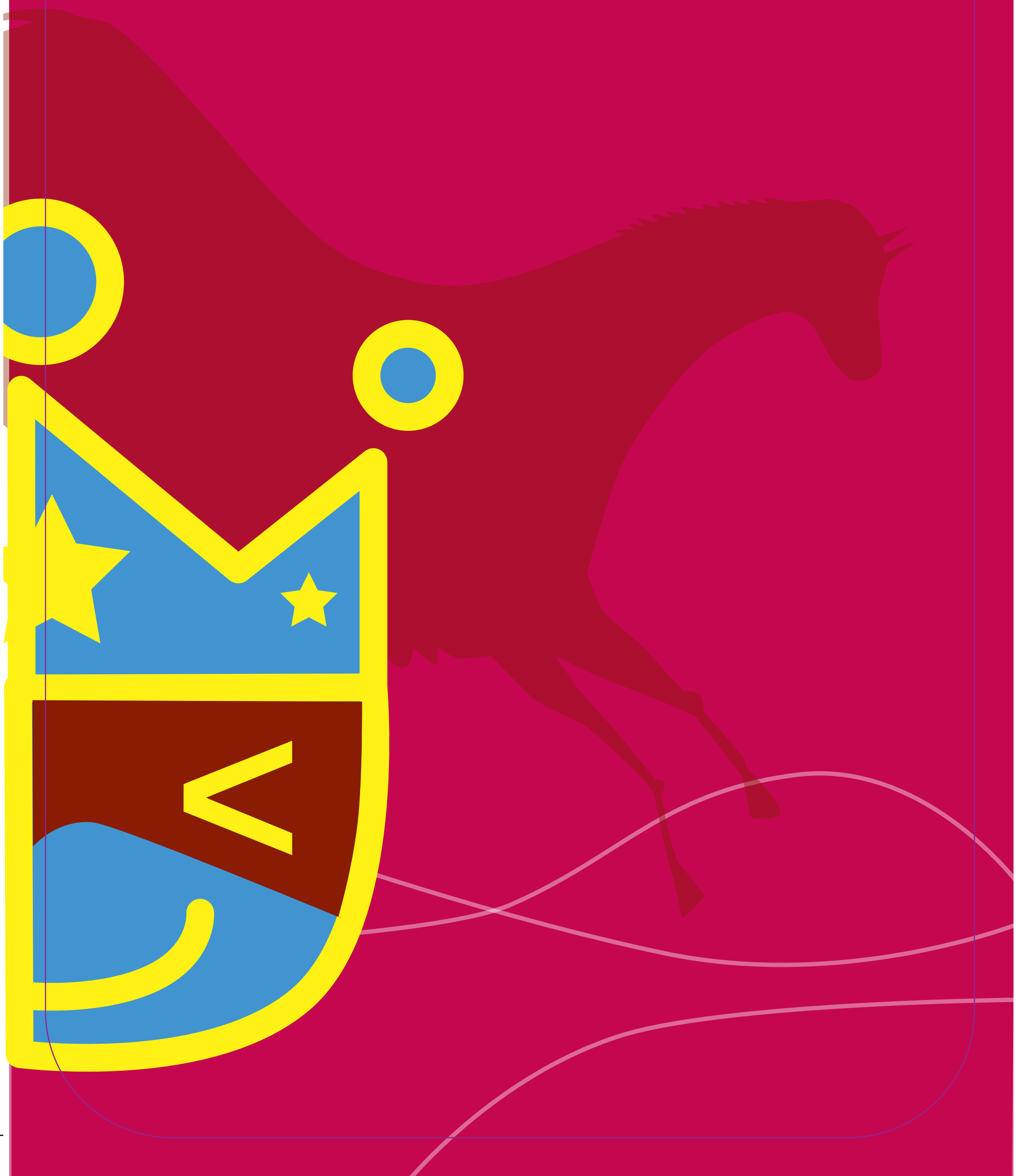
An attacker can choose to use weaker or no authentication.



J

Spoofing

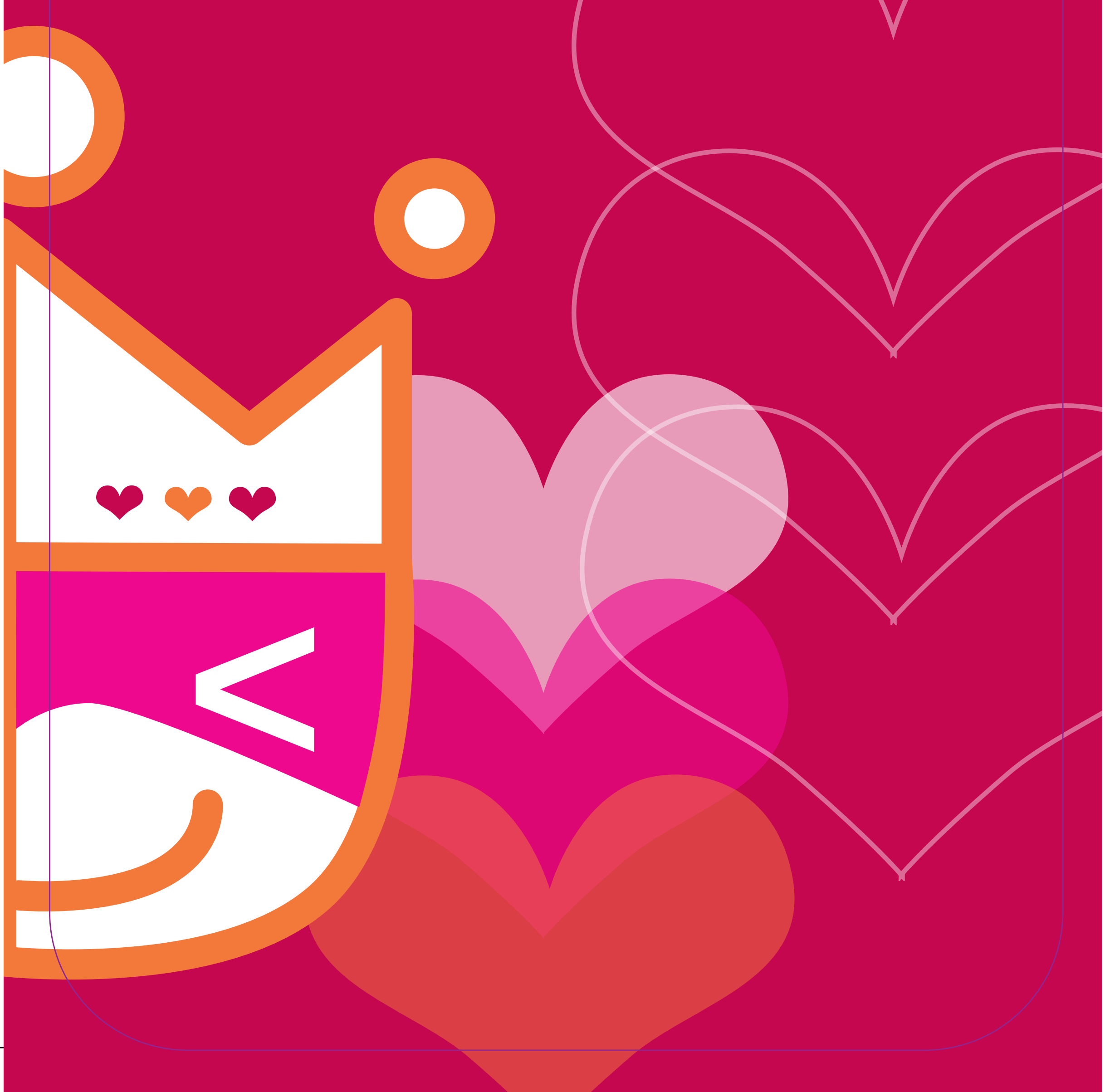
An attacker could steal credentials stored on the client and reuse them.



Q

Spoofing

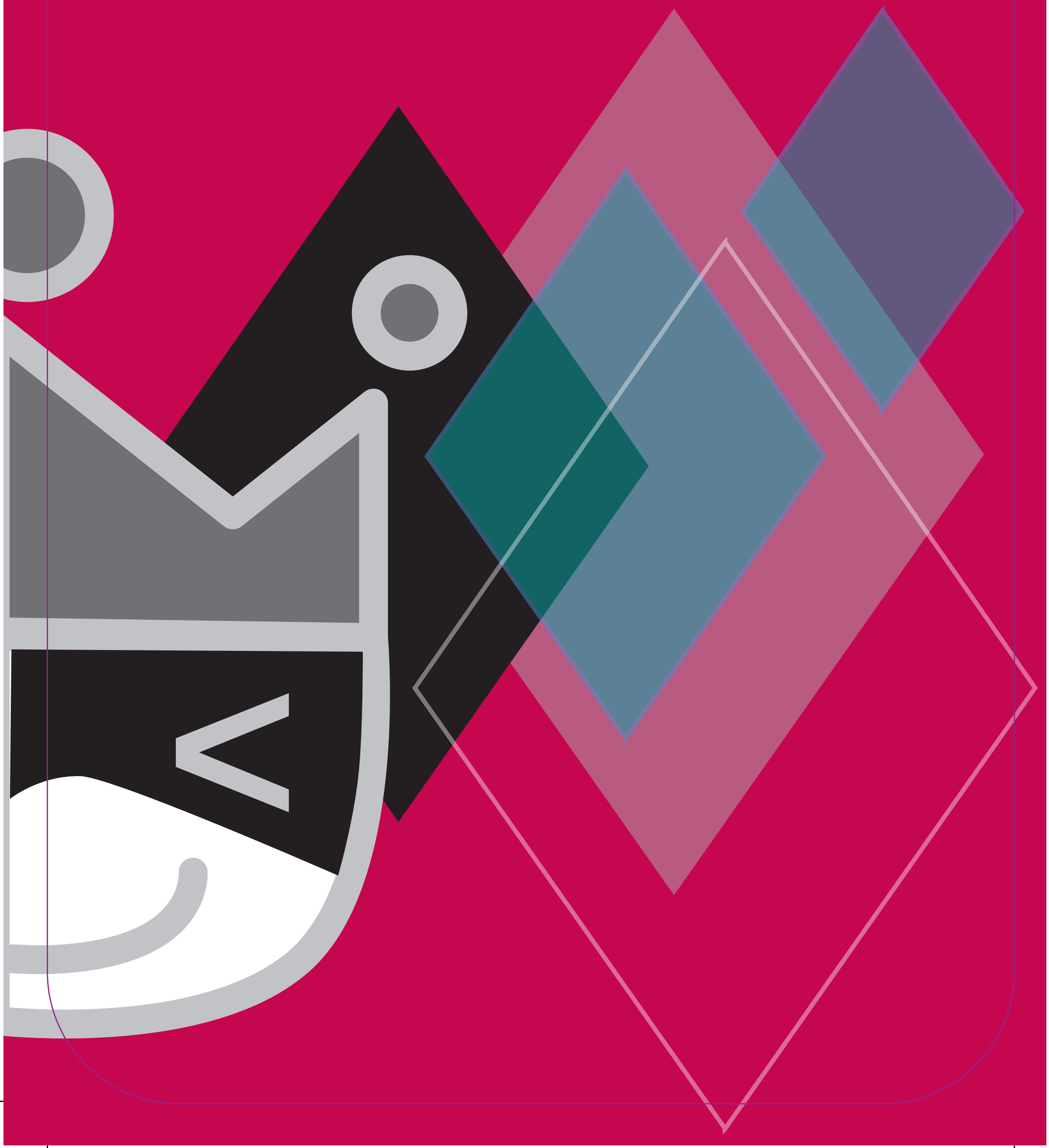
An attacker could go after the way credentials are updated or recovered (account recovery doesn't require disclosing the old password).



K

Spoofing

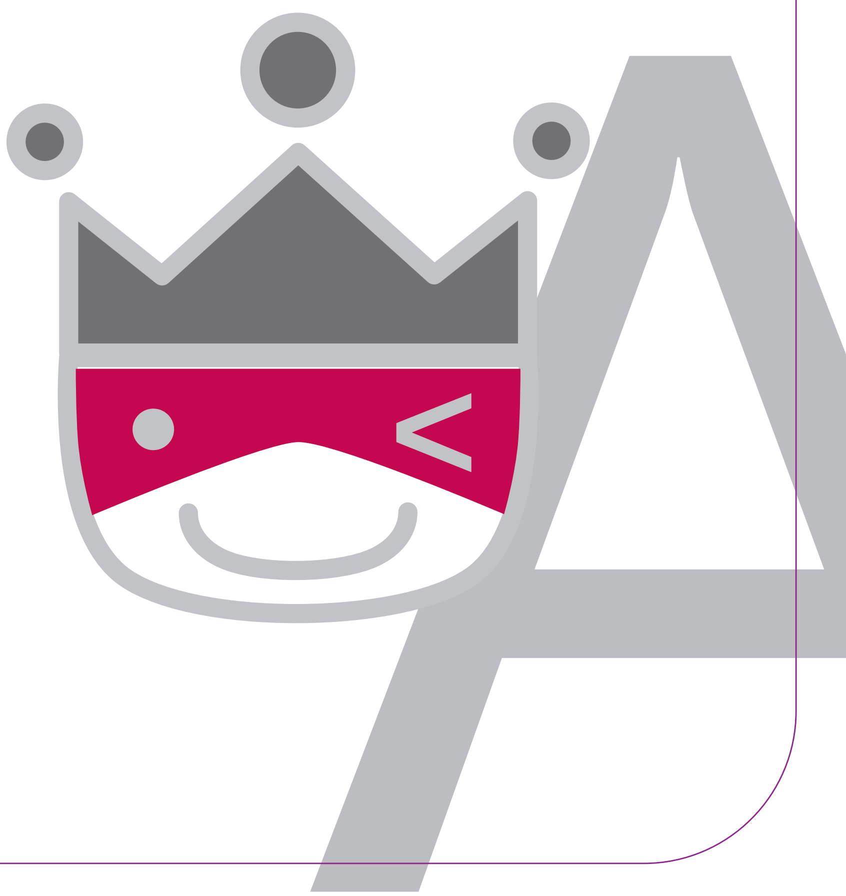
Your system ships with a default admin password and doesn't force a change.



A

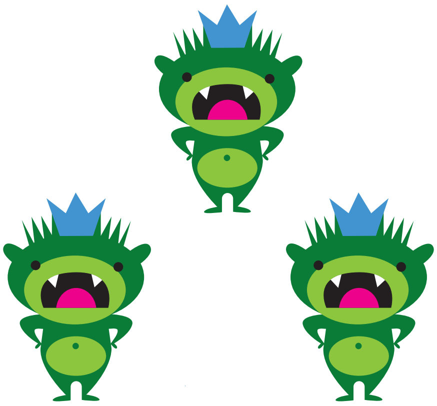
Spoofing

You’ve invented a new Spoofing attack.



3 Tampering

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

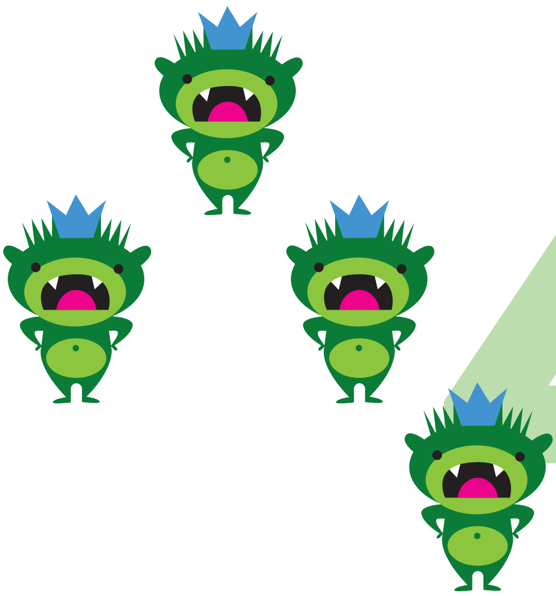


3

4

Tampering

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

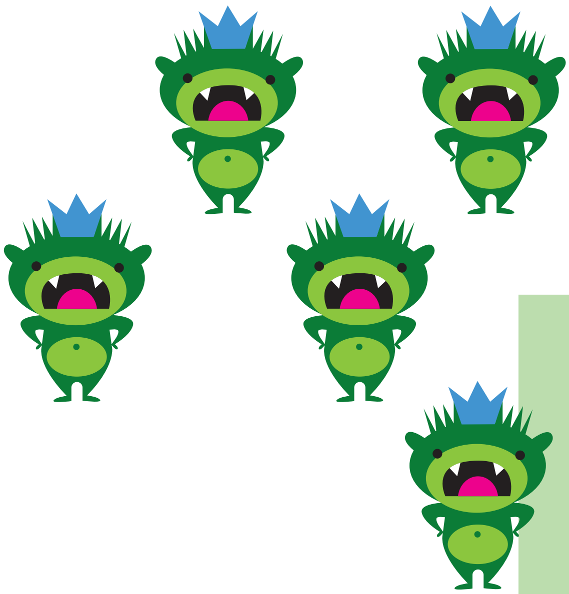


4

5

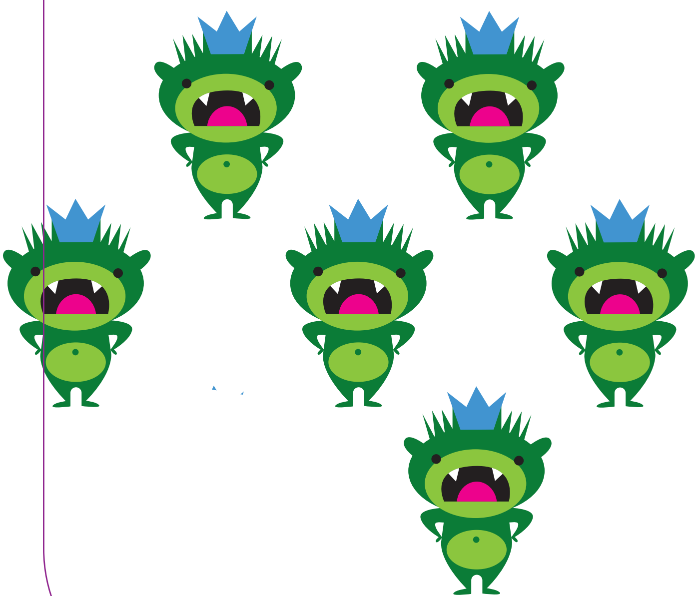
Tampering

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



6 Tampering

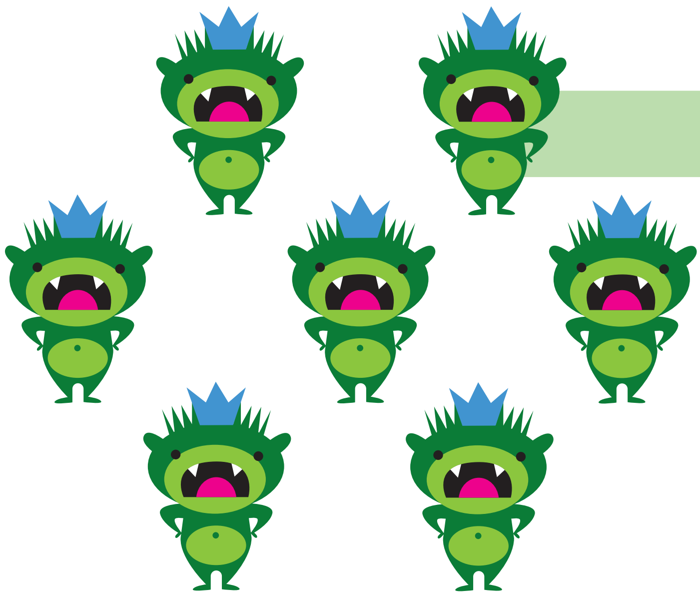
An attacker can write to a data store your code relies on.



7

Tampering

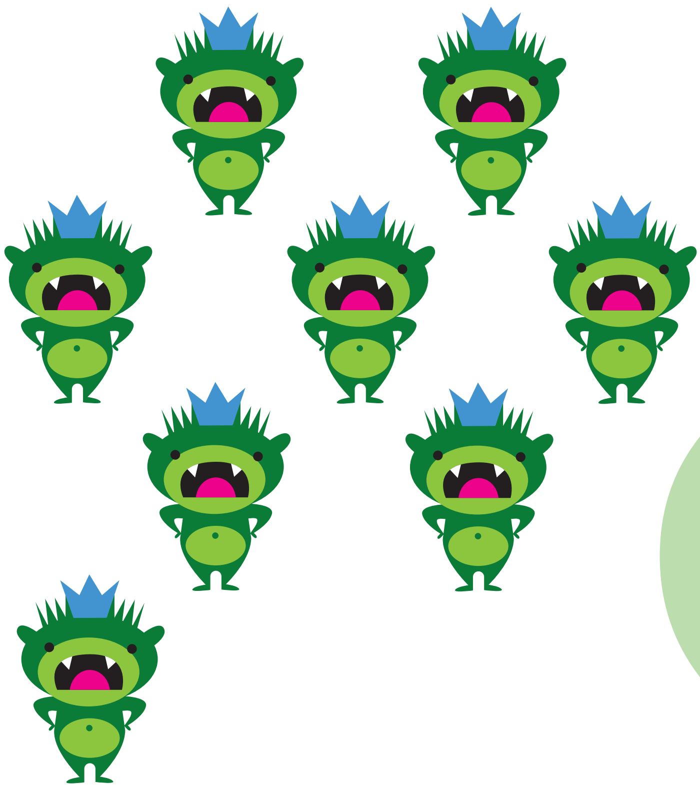
An attacker can bypass permissions because you don’t make names canonical before checking access permissions.



8

Tampering

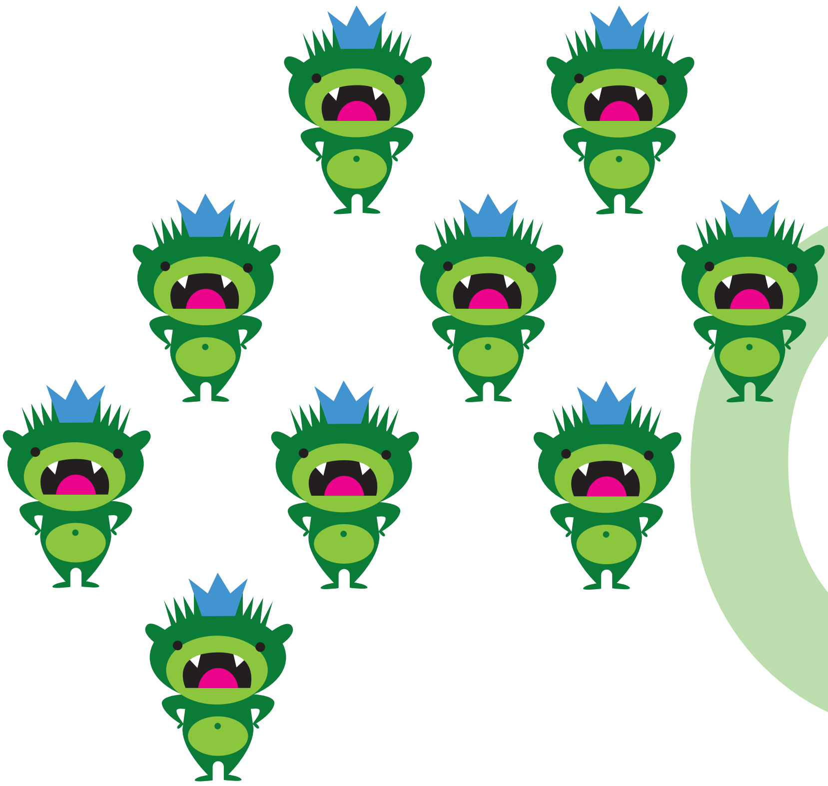
An attacker can manipulate data because there’s no integrity protection for data on the network.



9

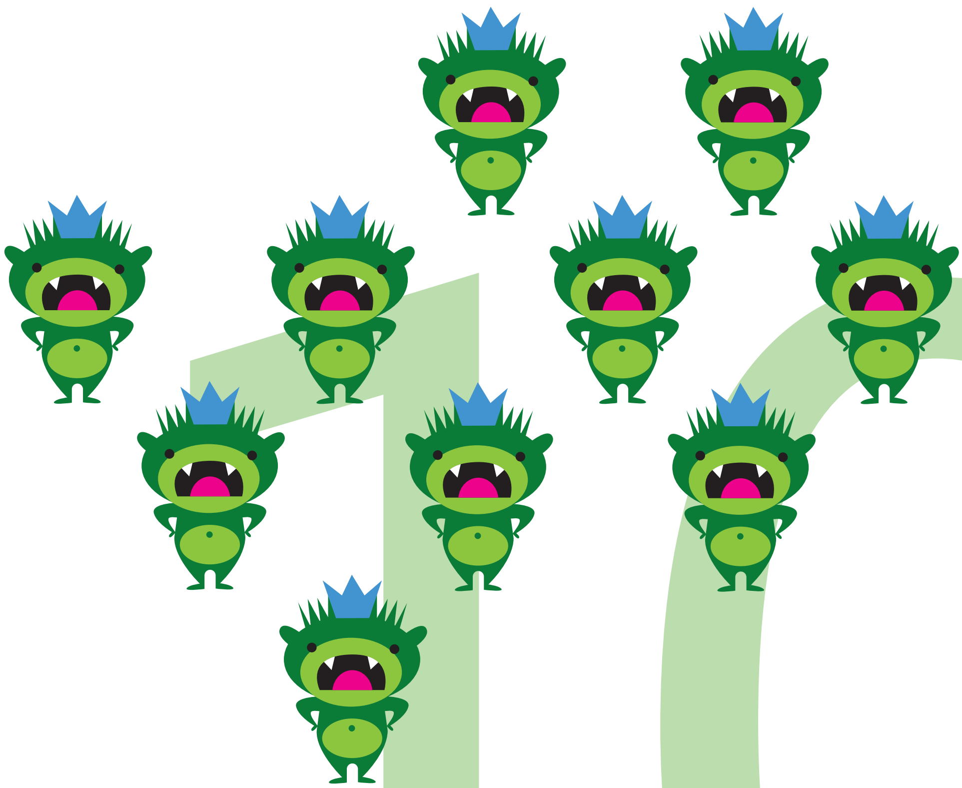
Tampering

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

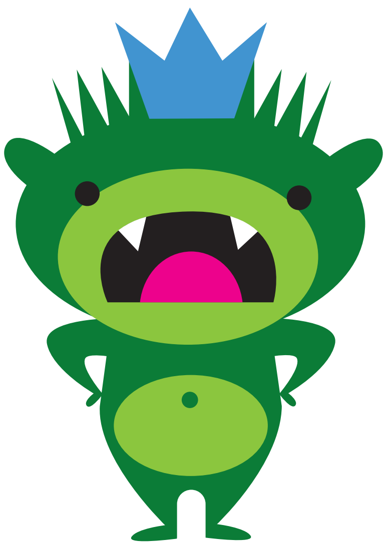
Tampering

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.

2

3

Repudiation

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

4

Repudiation

An attacker can alter files or messages 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.

5

6

Repudiation

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

7

Repudiation

An attacker can make the logs wrap around and lose data.

8

Repudiation

An attacker can make a log lose or confuse security information.

9

Repudiation

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

10

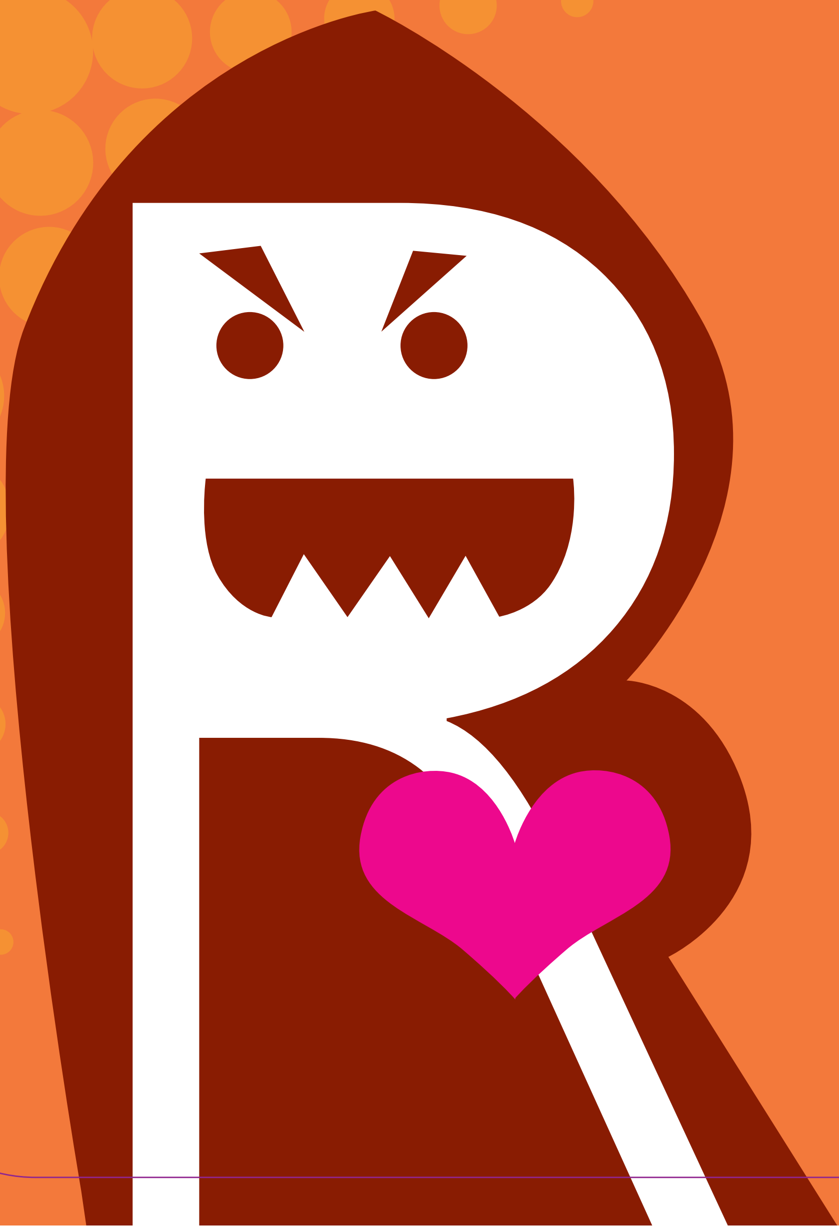
Repudiation

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 would have no way to prove them wrong.

K

Repudiation

The system has no logs.

logs = 0

A

Repudiation

You've invented a new Repudiation attack.

2

Information Disclosure

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

2

3

Information Disclosure

An attacker can see error messages with security-sensitive content.

3

4

Information Disclosure

An attacker can read content because messages (for example, 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.

6

7

Information Disclosure

An attacker can act as a “man in the middle” because you don’t authenticate endpoints of a network connection.

7

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.

Q

Information Disclosure

An attacker can read the entire channel because the channel (for example, HTTP or SMTP) isn’t encrypted.

K

Information Disclosure

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

A

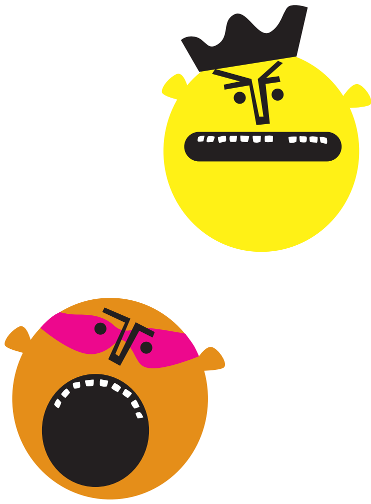
Information Disclosure

You've invented a new Information Disclosure attack.

2

Denial of Service

An attacker can make your authentication system unusable or unavailable.

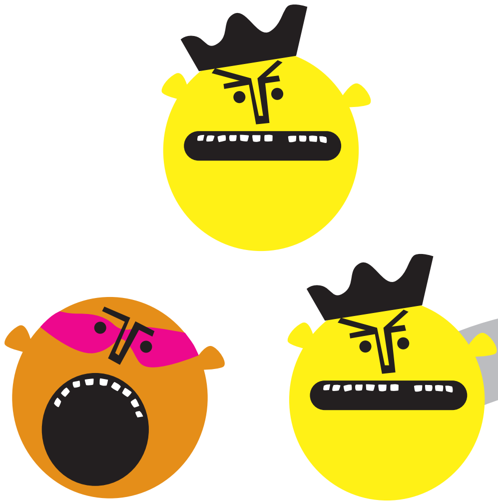


2

3

Denial of Service

An attacker can make a client unavailable or unusable but the problem goes away when the attacker stops **(client, authenticated, temporary)**.

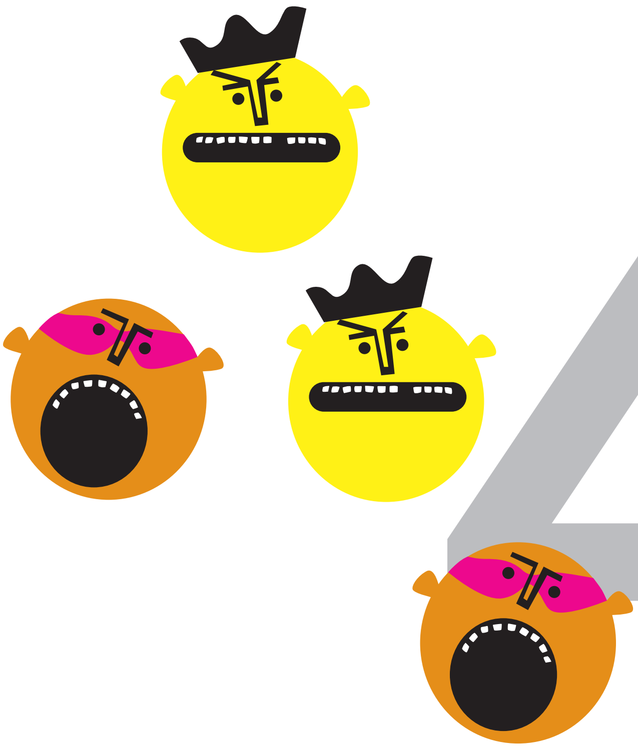


3

4

Denial of Service

An attacker can make a server unavailable or unusable but the problem goes away when the attacker stops **(server, authenticated, temporary)**.

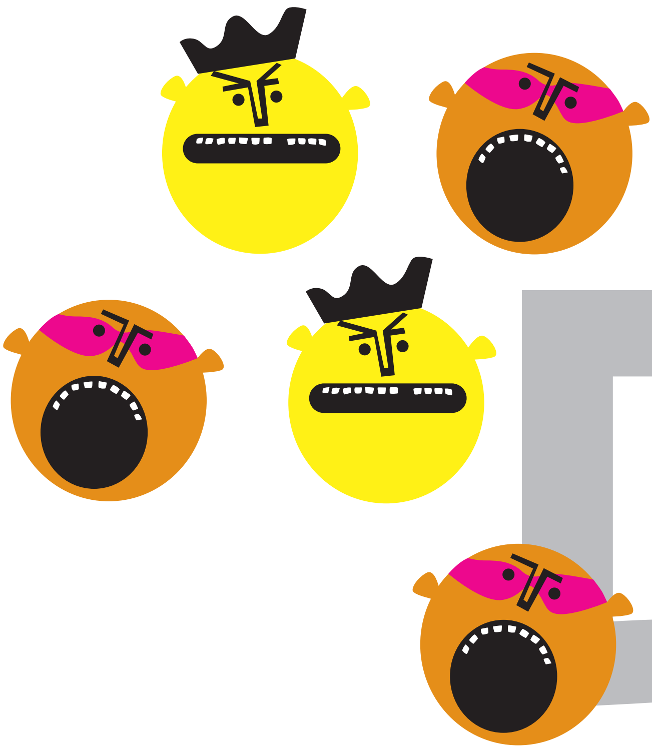


4

5

Denial of Service

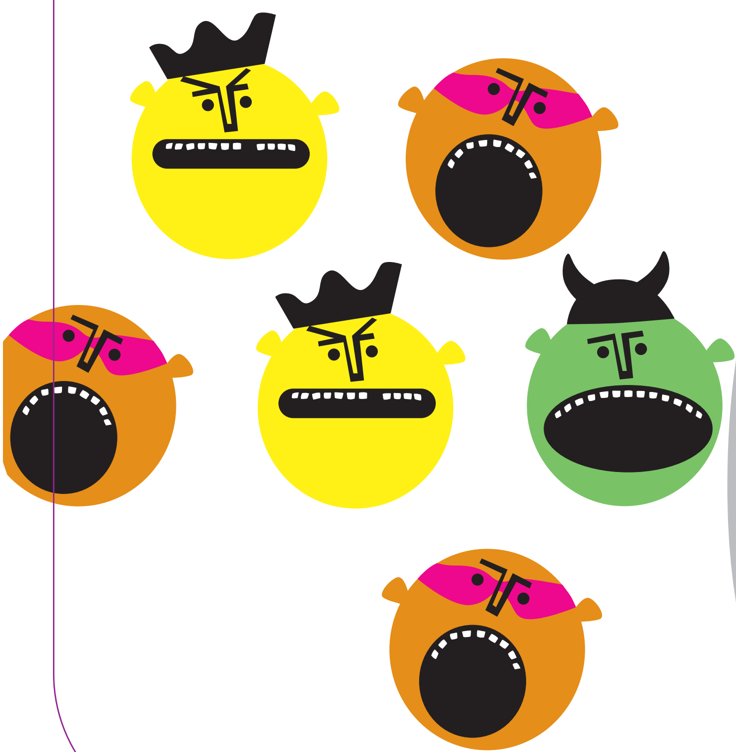
An attacker can make a client unavailable or unusable without ever authenticating, but the problem goes away when the attacker stops **(client, anonymous, temporary)**.



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 **(server, anonymous, temporary)**.

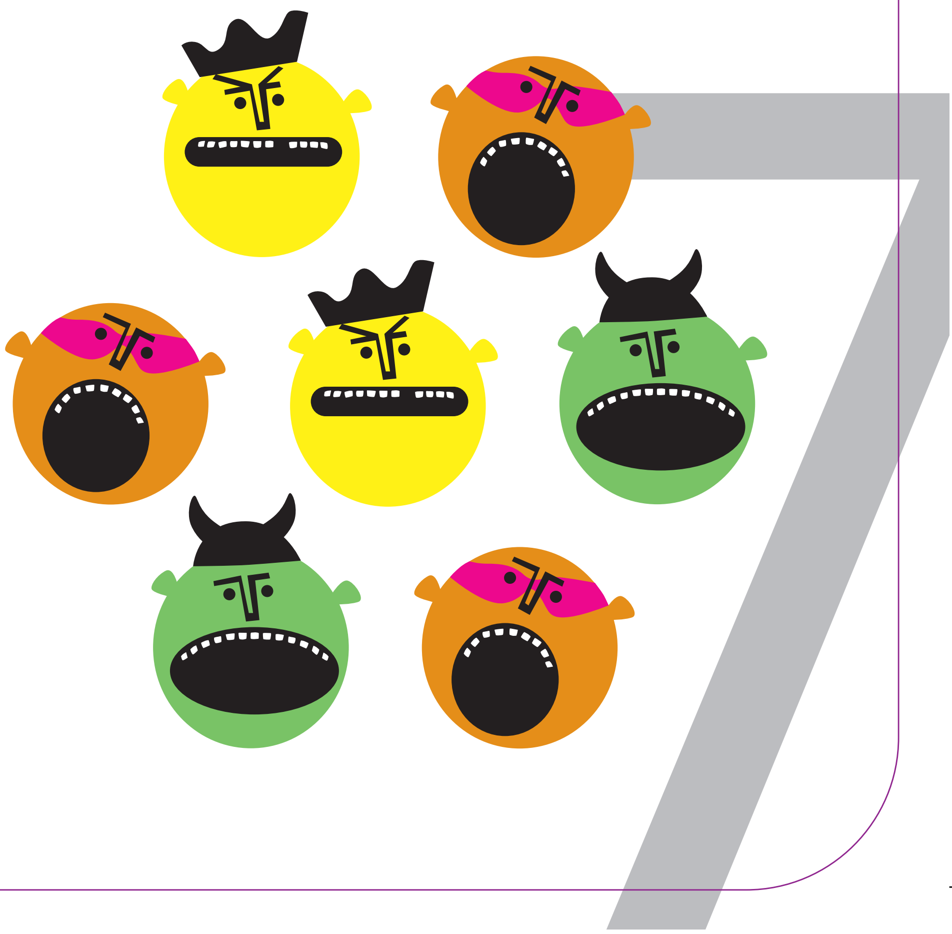


6

7

Denial of Service

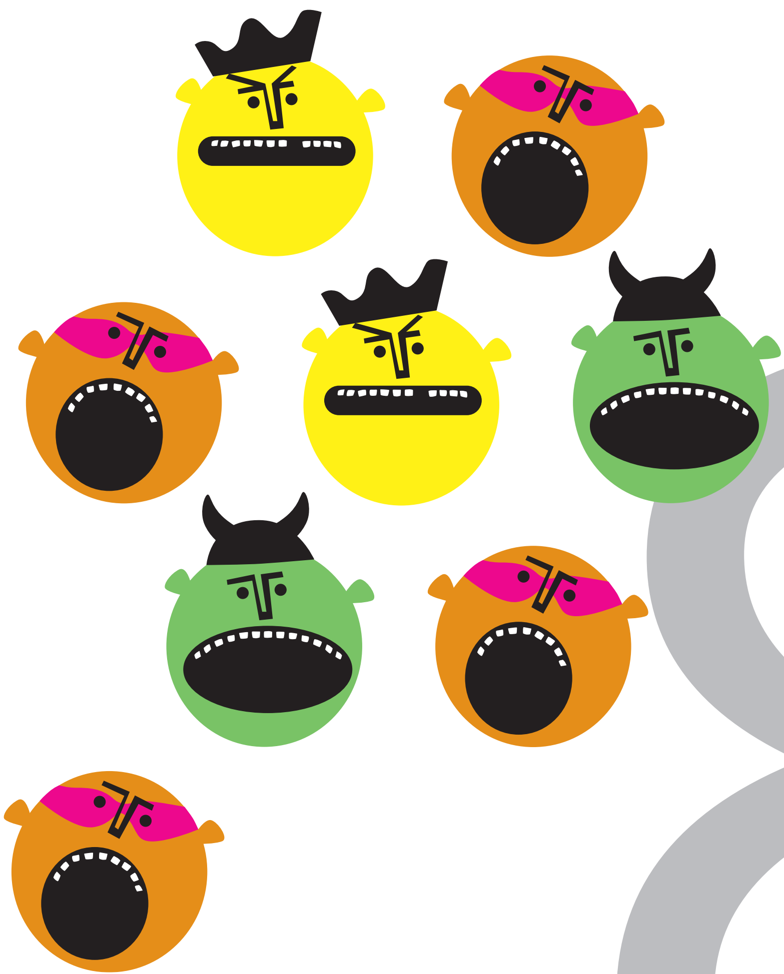
An attacker can make a client unavailable or unusable and the problem persists after the attacker goes away **(client, authenticated, persistent)**.



8

Denial of Service

An attacker can make a server unavailable or unusable and the problem persists after the attacker goes away **(server, authenticated, persistent)**.

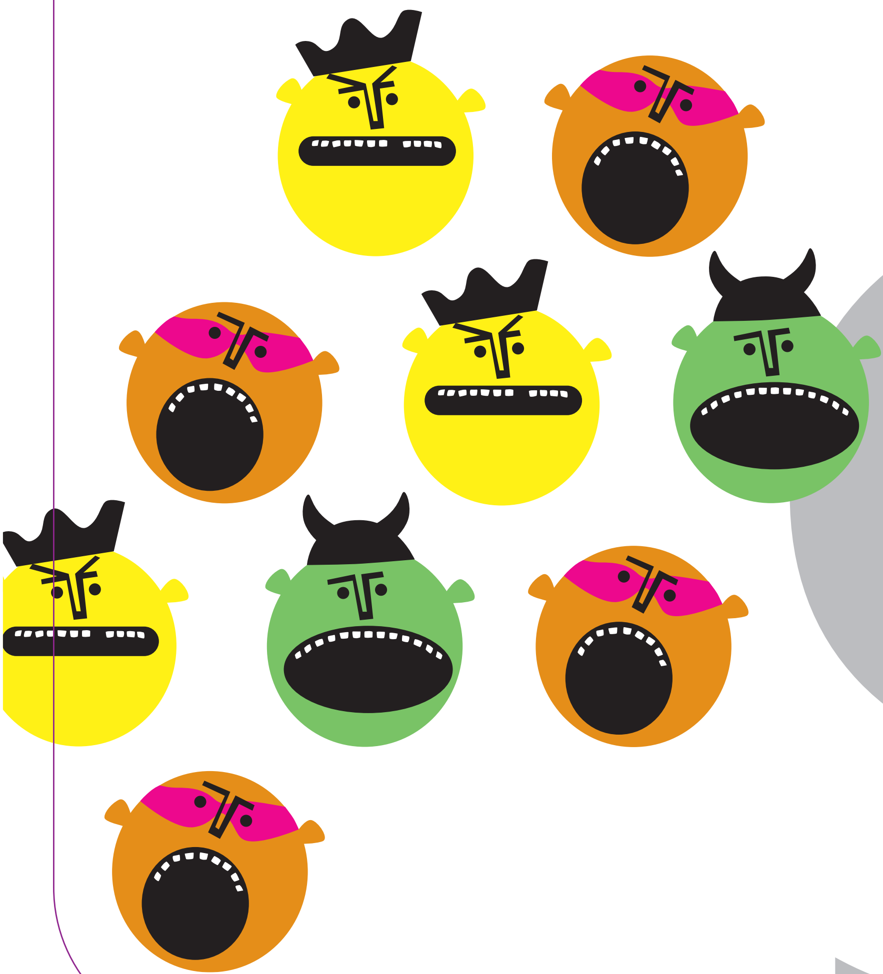


8

9

Denial of Service

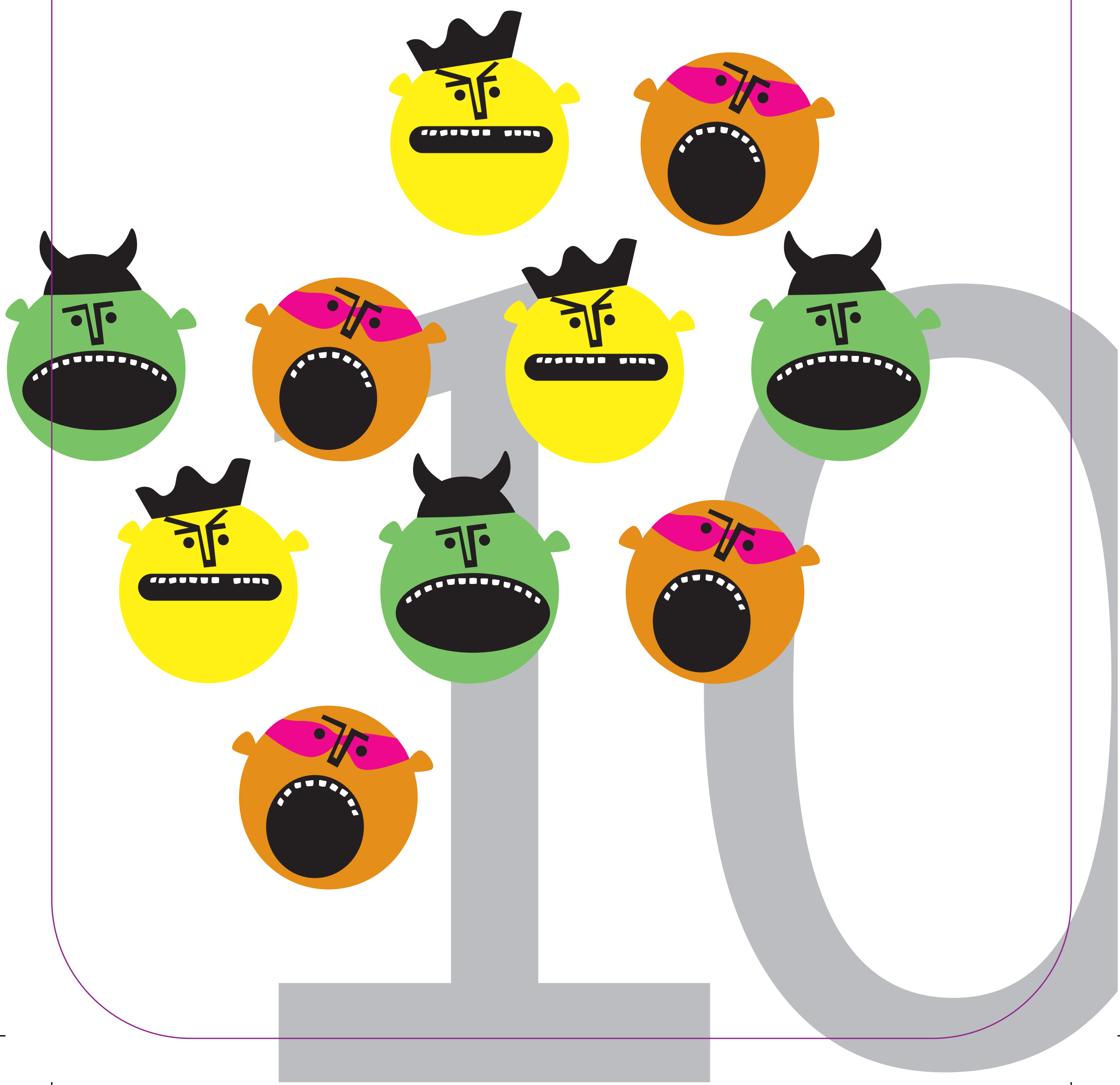
An attacker can make a client unavailable or unusable without ever authenticating, and the problem persists after the attacker goes away **(client, anonymous, persistent)**.



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 **(server, anonymous, persistent)**.



J

Denial of Service

An attacker can cause the logging subsystem to stop working.

Q

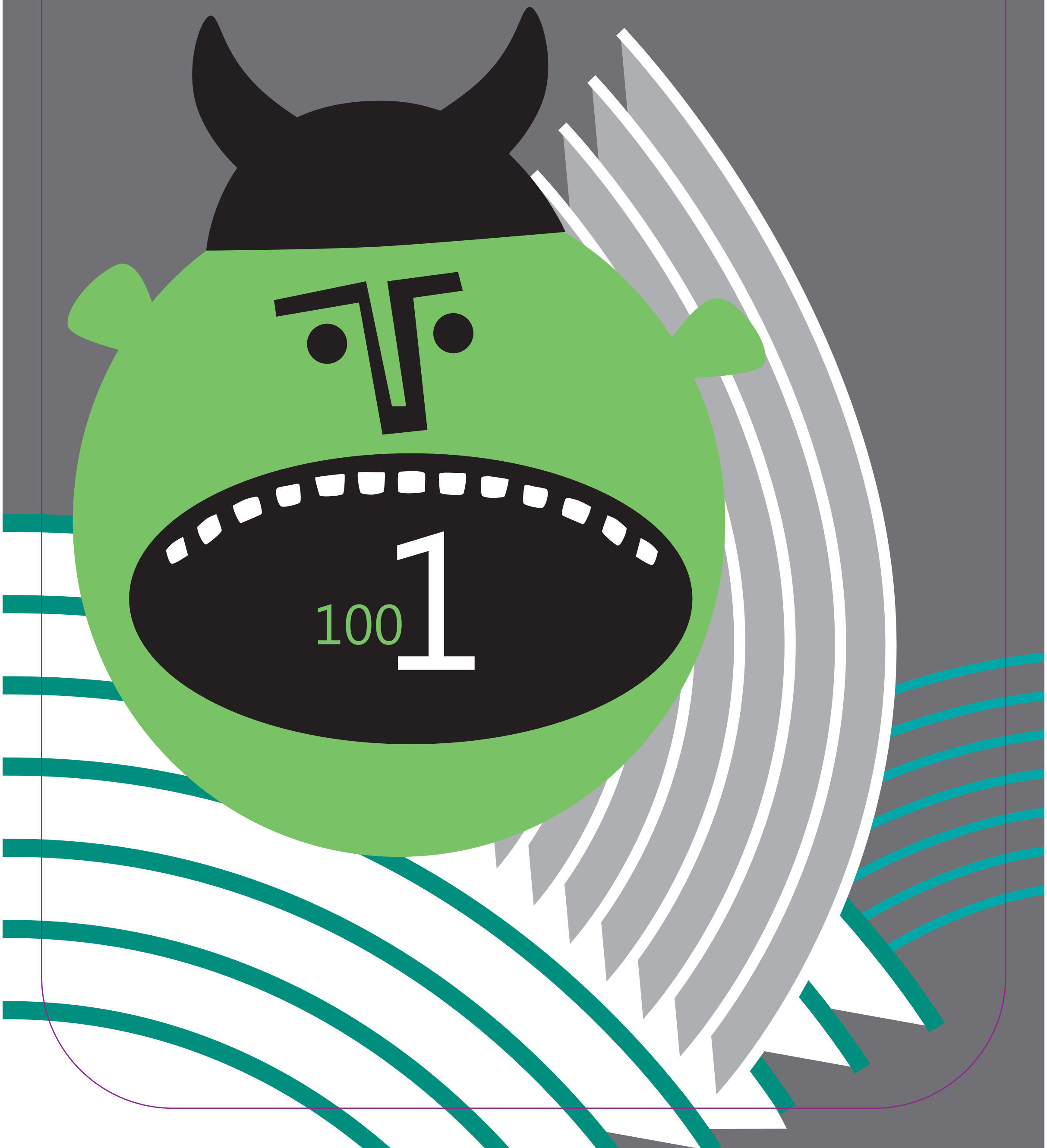
Denial of Service

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

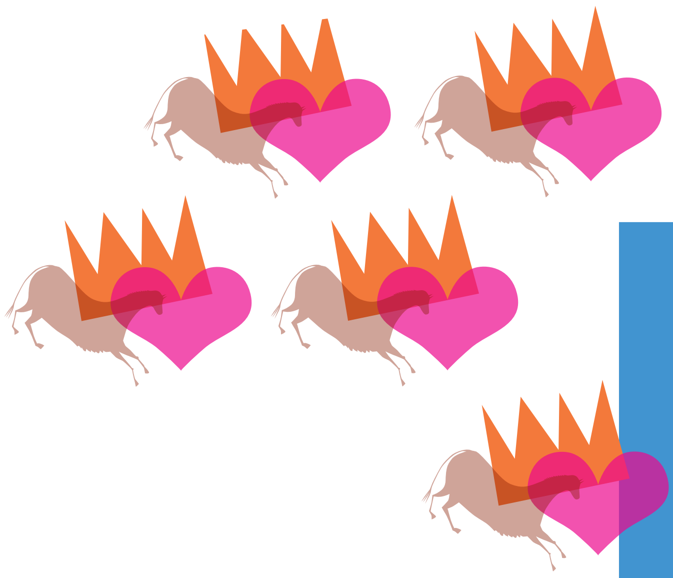
Denial of Service

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.

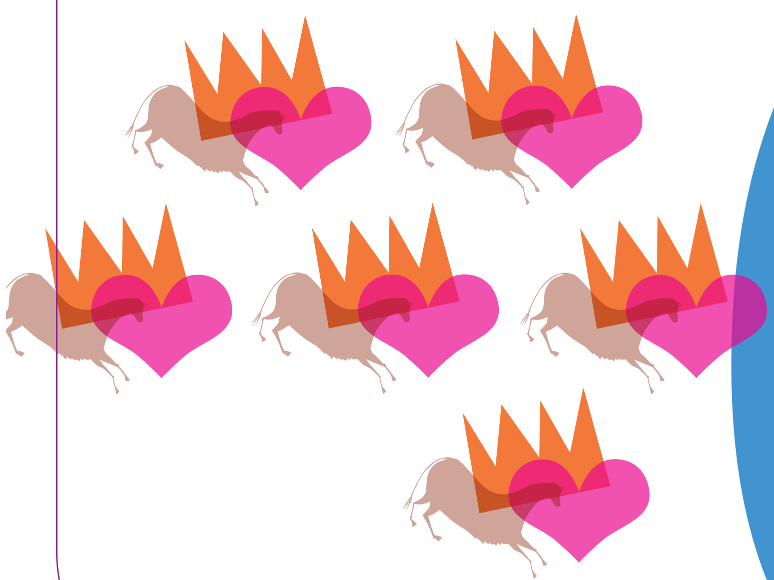


5

6

Elevation of Privilege

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

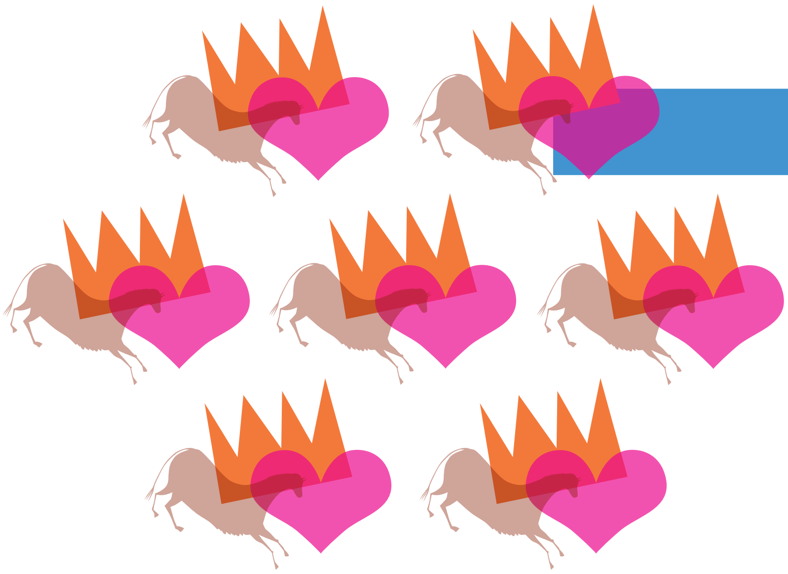


6

7

Elevation of Privilege

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

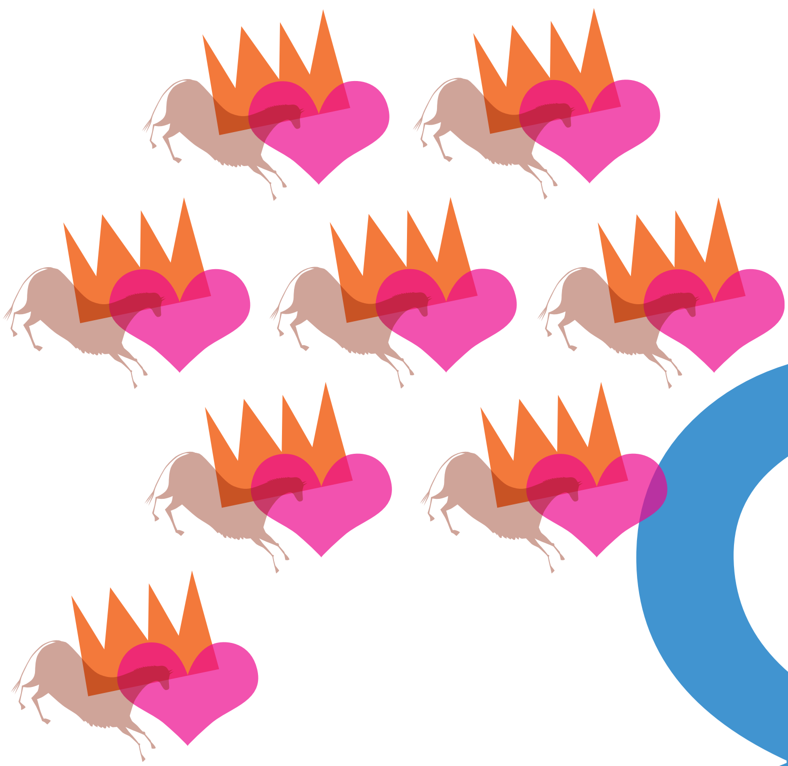


7

8

Elevation of Privilege

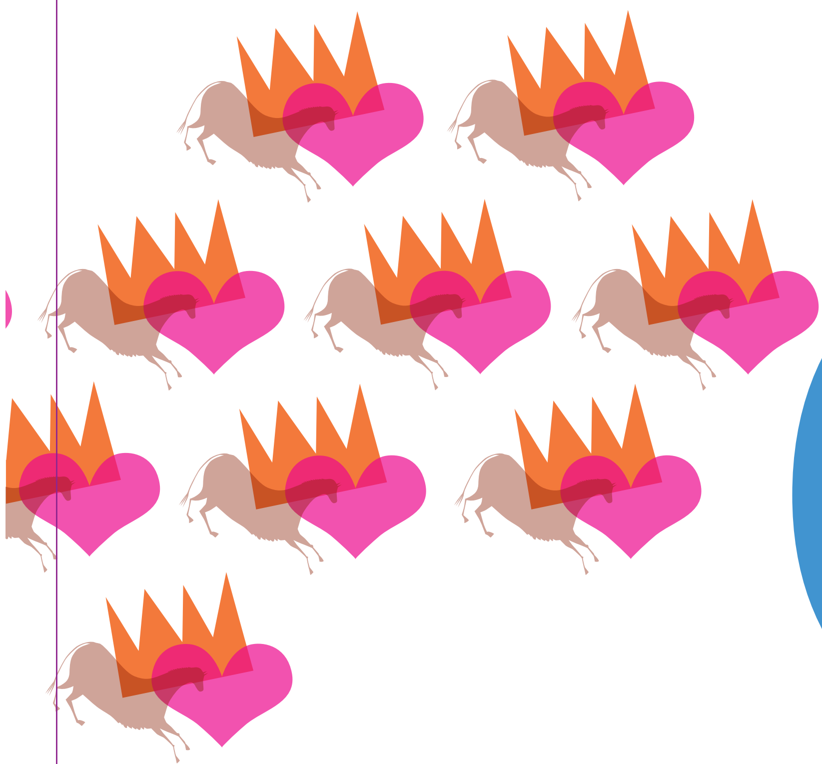
An attacker can enter data that is checked while still under the attacker’s control and used later on the other side of a trust boundary.



9

Elevation of Privilege

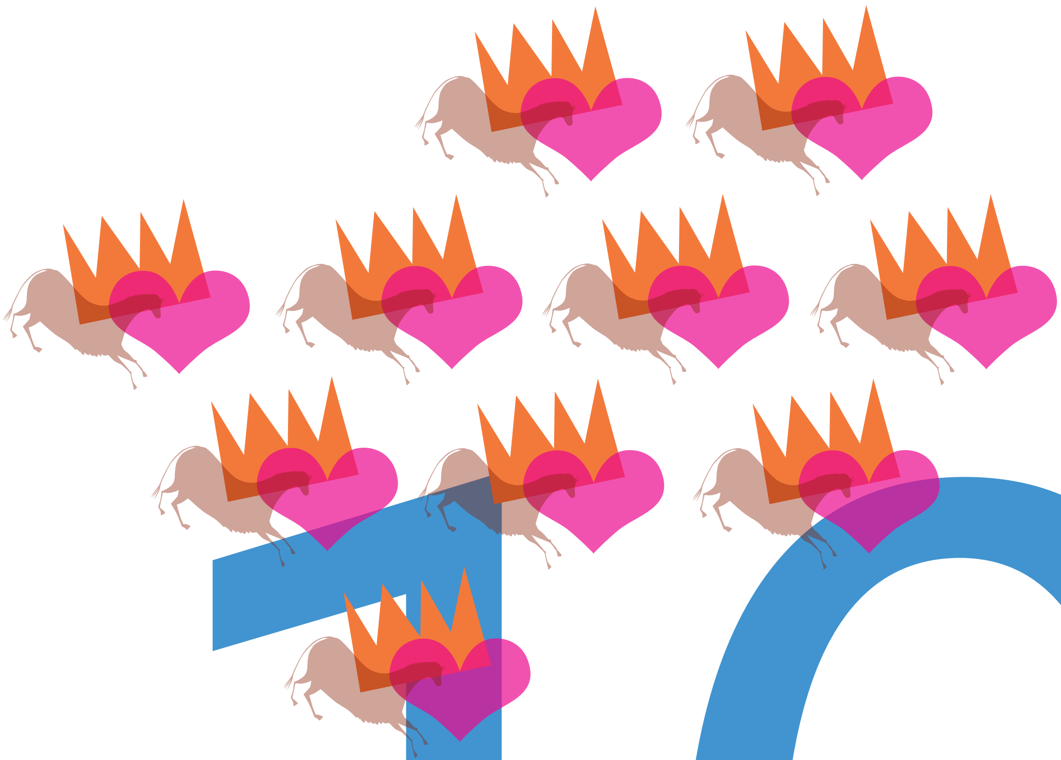
There's no reasonable way for callers to figure out what validation of tainted data you perform before passing it to them.



10

Elevation of Privilege

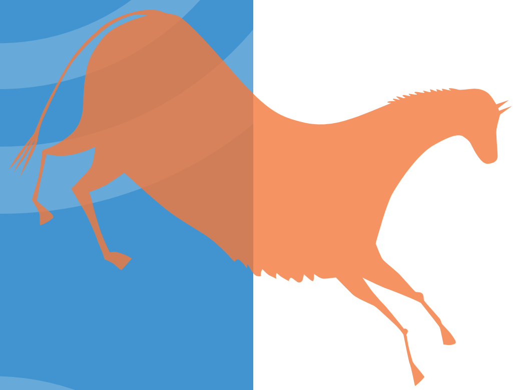
There’s no reasonable way for a caller to figure out what security assumptions you make.



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

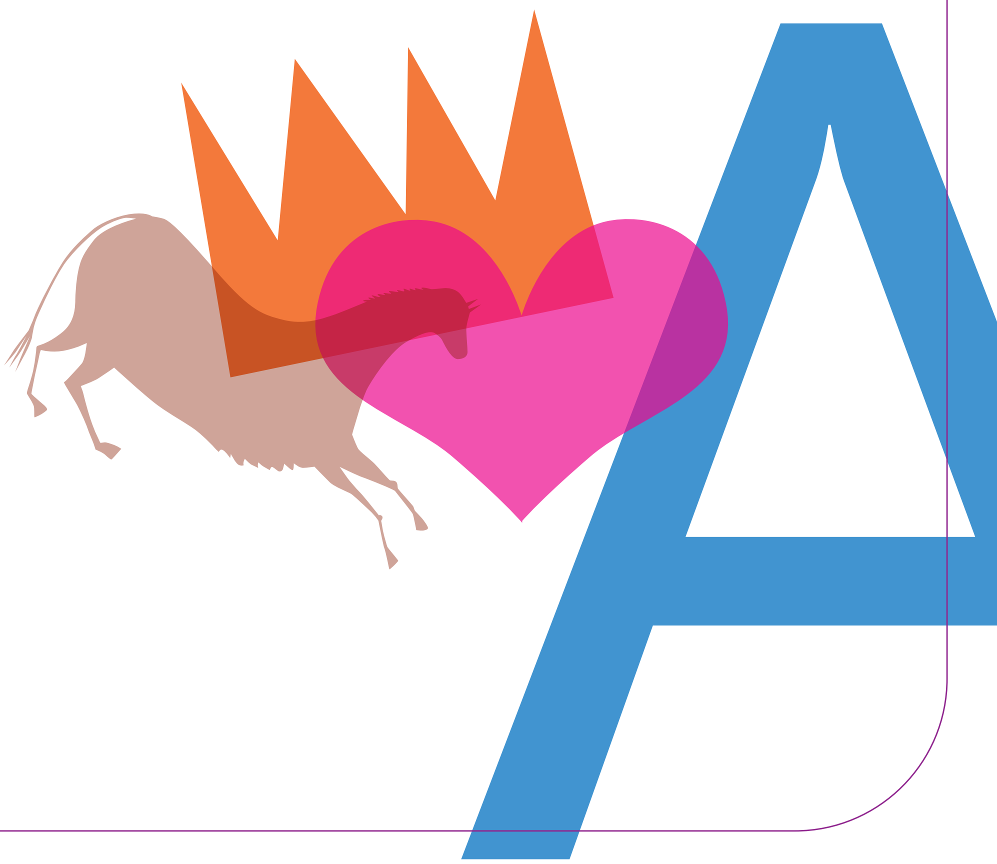
Elevation of Privilege

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).

continued on back

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").

continued on back

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 files or messages 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.

continued on back

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 (for example, 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.

continued on back

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)**.

continued on back

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 the attacker's control and used later on the other side of a trust boundary.

9. There's no reasonable way for callers 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

About

Threat Modeling

The Elevation of Privilege game 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 the game uses 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 not to have performed an action.

Information Disclosure: Exposing information to someone not authorized to see it.

Denial of Service: Denying or degrading service to users.

Elevation of Privilege: Gain capabilities without proper authorization.

At **www.microsoft.com/security/sdl/eop** we have videos, score sheets and tips and tricks for playing.

About