Disconnecting games with a single packet: an Unreal untold story

Hugo Bertin[1], Iliès Benhabbour[2], Prof. Marc Dacier[2], Prof. Yérom-David Bromberg[1]

1. IRISA, Univ. Rennes; 2. KAUST

Hack.lu 2024

Table of Contents

Table of Contents

Video Games: a booming industry

Video Games: a booming industry

- A lot of money is involved!!!
  - Forecasted revenue of 455 billion USD in 2024
  - More than twice the global movie and entertainment market

Video Games: a booming industry

- A lot of money is involved!!!
  - Forecasted revenue of 455 billion USD in 2024
  - More than twice the global movie and entertainment market
- Massive investments
  - Saudi Arabia invested $20B in the gaming sector [1]

[1] A. Raza, “Ali Raza on LinkedIn: #gaming #esports #investment #tech | 100 comments,” *Linkedin.com*, Oct. 06, 2024. https://www.linkedin.com/feed/update/urn:li:activity:7248566123669925888/ (accessed Oct. 14, 2024).

Video Games: a booming industry

- A lot of money is involved!!!
  - Forecasted revenue of 455 billion USD in 2024
  - More than twice the global movie and entertainment market
- Massive investments
  - Saudi Arabia invested $20B in the gaming sector [1]
- Continuously growing interest in Esports
  - The 2024 Esports World Cup in Ryadh attracted 500 million viewers [1]
  - The International 10 tournament’s prize pool: $40 million [2]
  - The first Olympic Esports games will be host in Saudi Arabia in 2025 [1]

[1] A. Raza, “Ali Raza on LinkedIn: #gaming #esports #investment #tech | 100 comments,” *Linkedin.com*, Oct. 06, 2024. https://www.linkedin.com/feed/update/urn:li:activity:7248566123669925888/ (accessed Oct. 14, 2024).
[2] S. Nordmark and J. Heath, “The 10 Largest Prize Pools in Esports,” Dot Esports, Aug. 28, 2019. https://dotesports.com/general/news/biggest-prize-pools-esports-14605

Cheating: a major threat to the industry

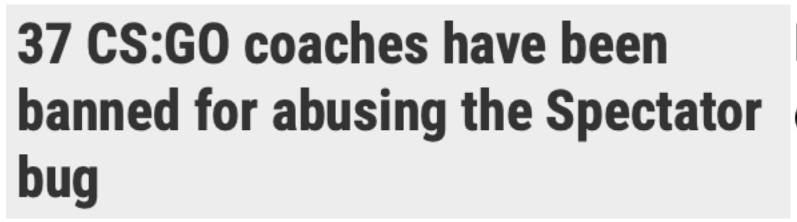
Money

Cheating: a major threat to the industry

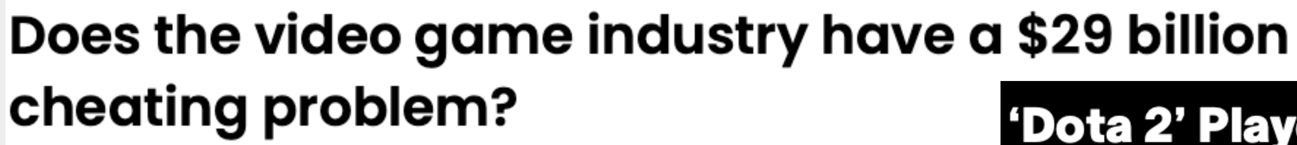
$$\begin{tikzcd}\text{Money} \arrow[r] & \text{Frauds}\end{tikzcd}$$

Cheating: a major threat to the industry

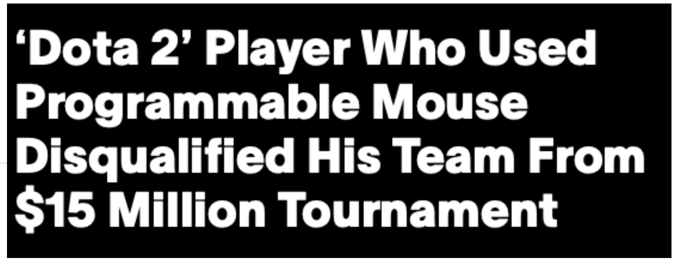
$$\begin{tikzcd}\text{Money} \arrow[r] & \text{Frauds} \arrow[r] & \text{Cheating}\end{tikzcd}$$


37 CS:GO coaches have been banned for abusing the Spectator bug

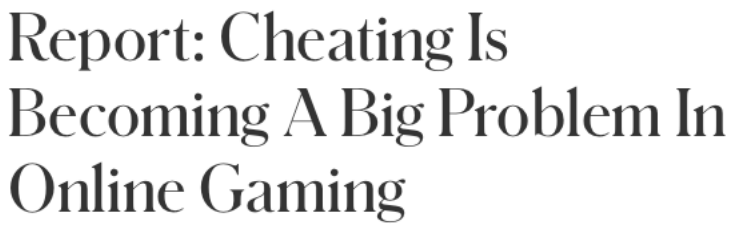
[1] E. Matthews, “37 CS:GO coaches have been banned for abusing the Spectator bug,” *PC Gamer*, Sep. 28, 2020. https://www.pcgamer.com/37-csgo-coaches-have-been-banned-for-abusing-the-spectator-bug/


Does the video game industry have a $29 billion cheating problem?

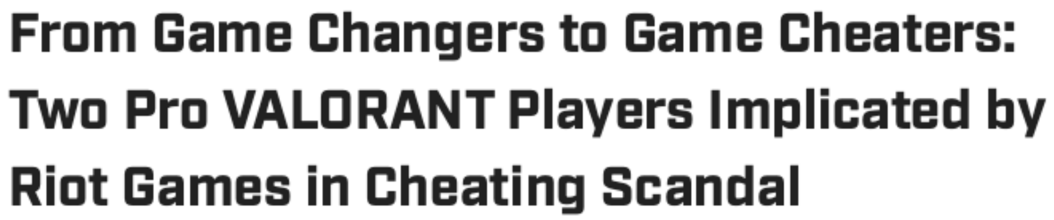
[2] “Does the video game industry have a $29 billion cheating problem?,” *Gamedeveloper.com*, 2020. https://www.gamedeveloper.com/business/does-the-video-game-industry-have-a-29-billion-cheating-problem-


‘Dota 2’ Player Who Used Programmable Mouse Disqualified His Team From $15 Million Tournament

[5] M. Gault, “‘Dota 2’ Player Who Used Programmable Mouse Disqualified His Team From $15 Million Tournament,” *VICE*, Jun. 25, 2018. https://www.vice.com/en/article/dota-2-player-who-used-programmable-mouse-disqualified-his-team-from-dollar15-million-tournament/


Report: Cheating Is Becoming A Big Problem In Online Gaming

[4] N. Granados, “Report: Cheating Is Becoming A Big Problem In Online Gaming,” *Forbes*. https://www.forbes.com/sites/nelsongranados/2018/04/30/report-cheating-is-becoming-a-big-problem-in-online-gaming/

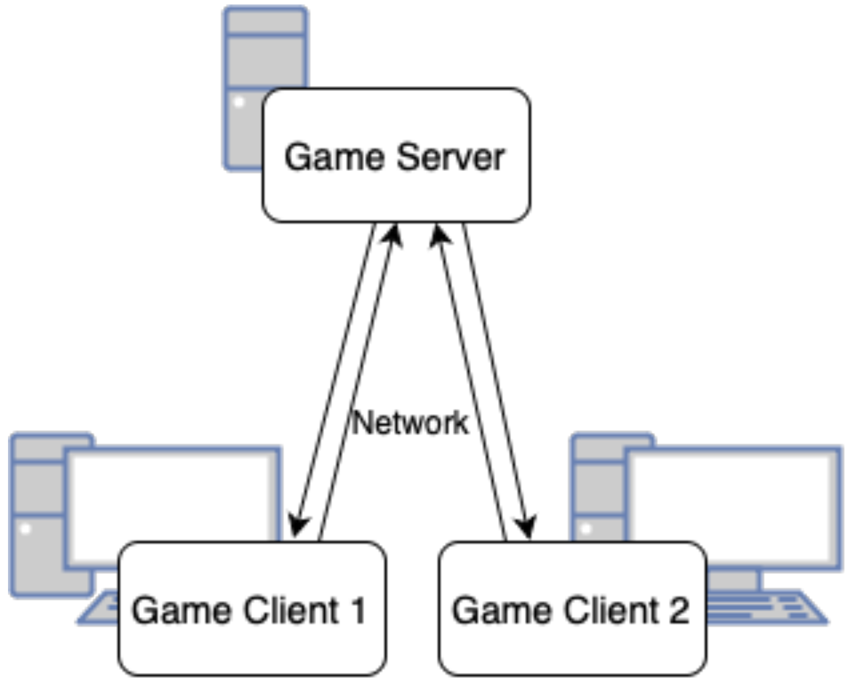

From Game Changers to Game Cheaters: Two Pro VALORANT Players Implicated by Riot Games in Cheating Scandal

[3] G. DeSena, “From Game Changers to Game Cheaters: Two Pro VALORANT Players Implicated by Riot Games in Cheating Scandal - Esports Illustrated,” *Esports Illustrated On SI*, Jan. 17, 2024. https://www.si.com/esports/valorant/noot-noot-cheating-scandal

Cheating in Online Games

Cheating in Online Games

- Focus on the client
  - Client cheats: aimbots, wall-hacks…
  - Mitigation: Anti-Cheats



Cheating in Online Games

- Focus on the client
  - Client cheats: aimbots, wall-hacks…
  - Mitigation: Anti-Cheats
- The network can also be leveraged to cheat

  => Our focus in this talk

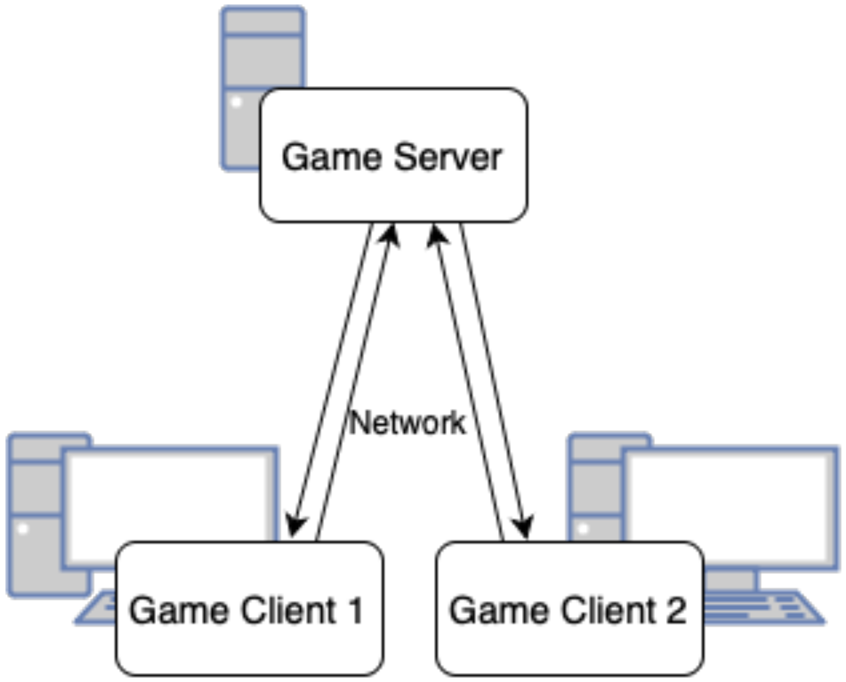
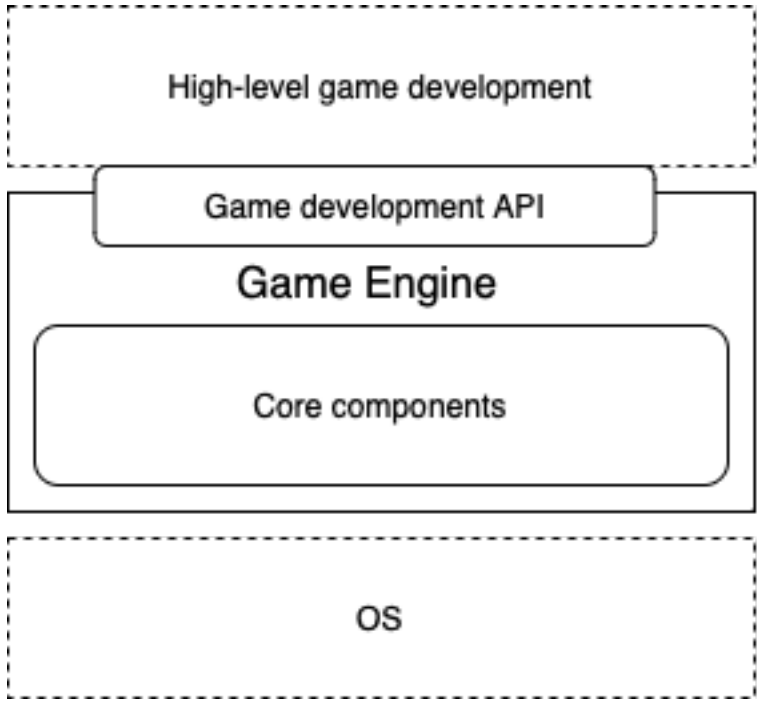


Table of Contents

Game Engines

- Unreal Engine, Source Engine, Unity…
- Game engines simplify video games development
  - Common libraries to re-use
  - Physics, graphics, networking
- Used for server and client

=> Many games share pieces of software



Unreal Engine (UE)

- Unreal Engine
  - Epic Games' game engine
  - Publicly available source code
  - Powering famous games:
    - Fortnite, Valorant, PUBG, The Finals, Sea of Thieves …

Unreal Engine Networking

Unreal Engine Networking

- Game communications use an application-layer protocol based on UDP

Unreal Engine Networking

- Game communications use an application-layer protocol based on UDP
- Joining a game session:
    1. First the player authenticates himself to the game provider's server to retrieve information to contact the game server (IP address, Port number).
    2. The game client contacts the game server to initiate a connection process to establish a connection through a handshake.
    3. Client and server can now exchange information about the game.

Unreal Engine Networking

- Game communications use an application-layer protocol based on UDP
- Joining a game session:
    1. First the player authenticates himself to the game provider’s server to retrieve information to contact the game server (IP address, Port number).
    2. The game client contacts the game server to initiate a connection process to establish a connection through a handshake.
    3. Client and server can now exchange information about the game.
- The game server identify clients by their IP:Port

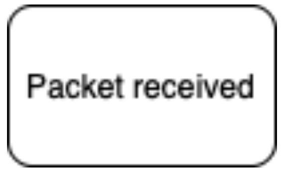
Unreal Engine Networking

- Game communications use an application-layer protocol based on UDP
- Joining a game session:
  1. First the player authenticates himself to the game provider’s server to retrieve information to contact the game server (IP address, Port number).
  2. The game client contacts the game server to initiate a connection process to establish a connection through a handshake.
  3. Client and server can now exchange information about the game.
- The game server identify clients by their IP:Port
- The game server’s IP:Port is the same for all the clients

High-level overview of the vulnerability

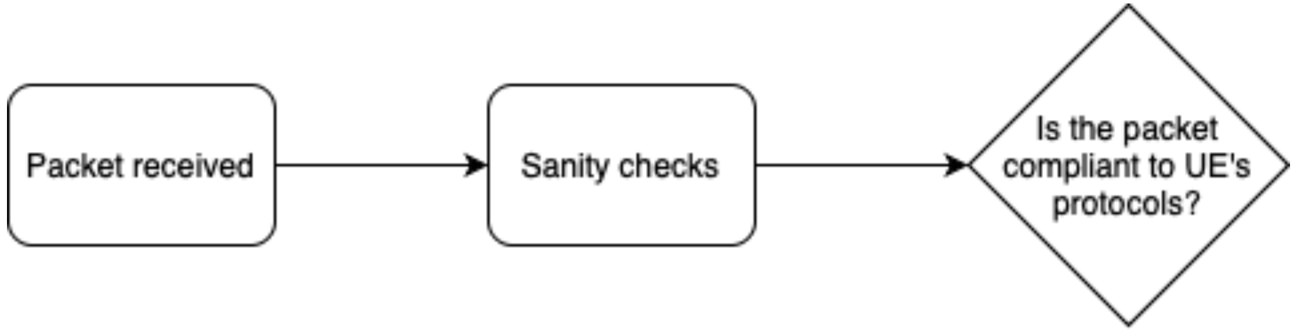
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine



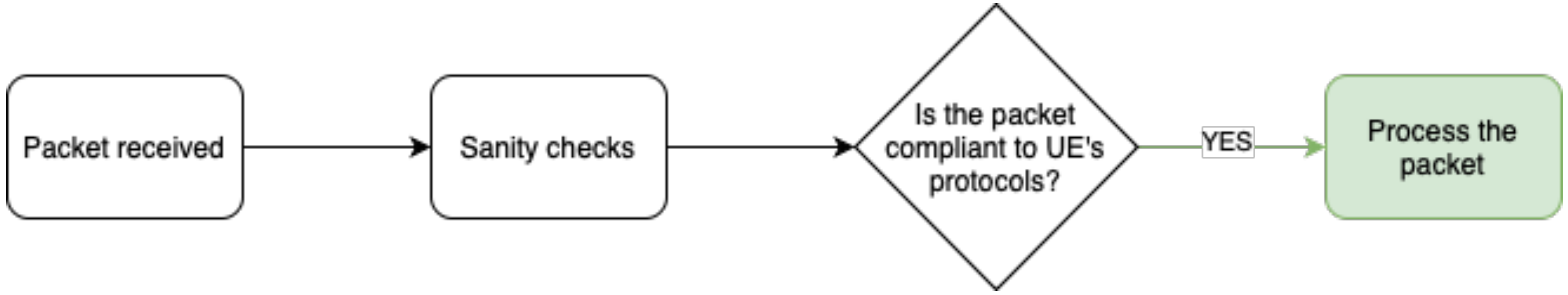
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine



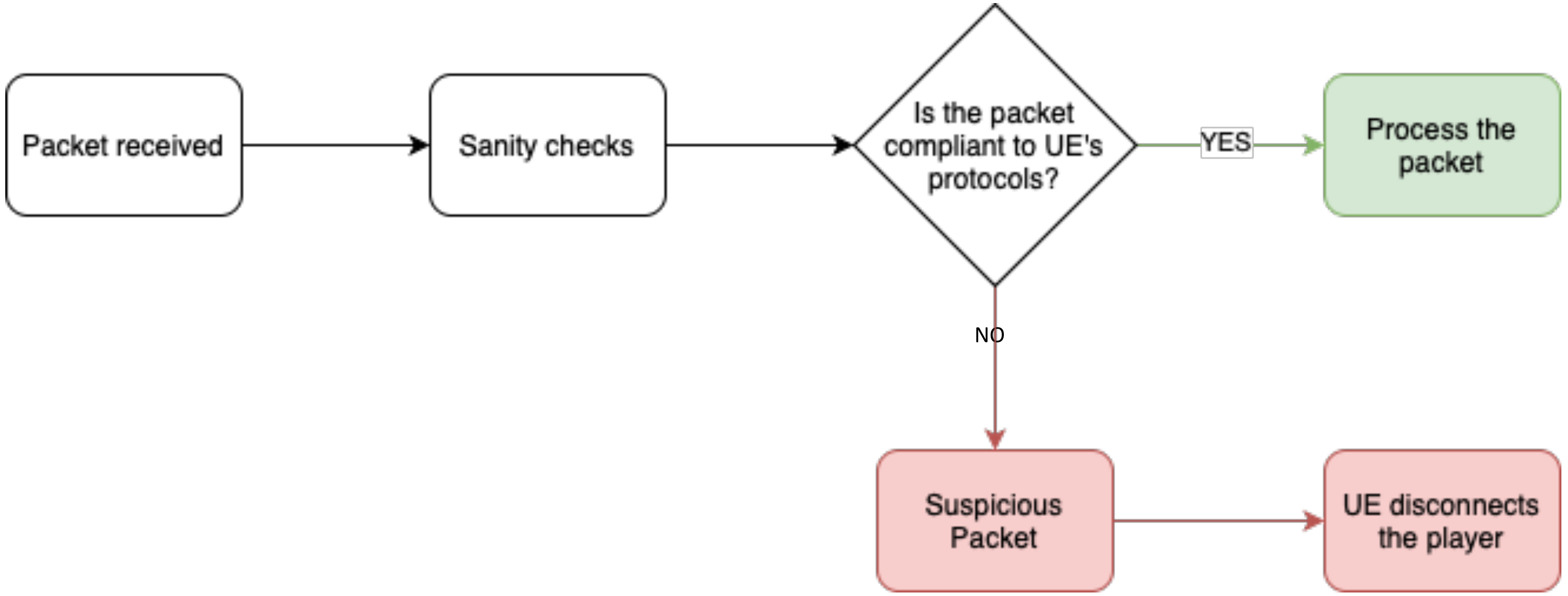
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine



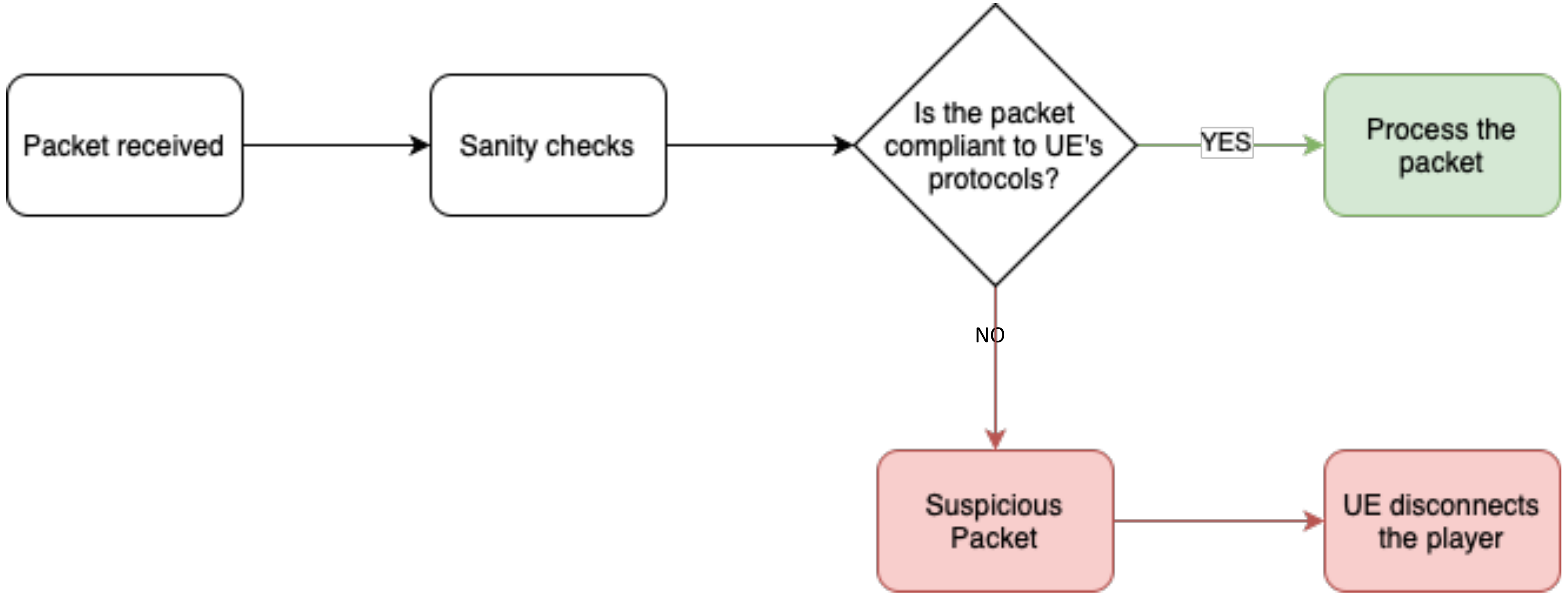
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine



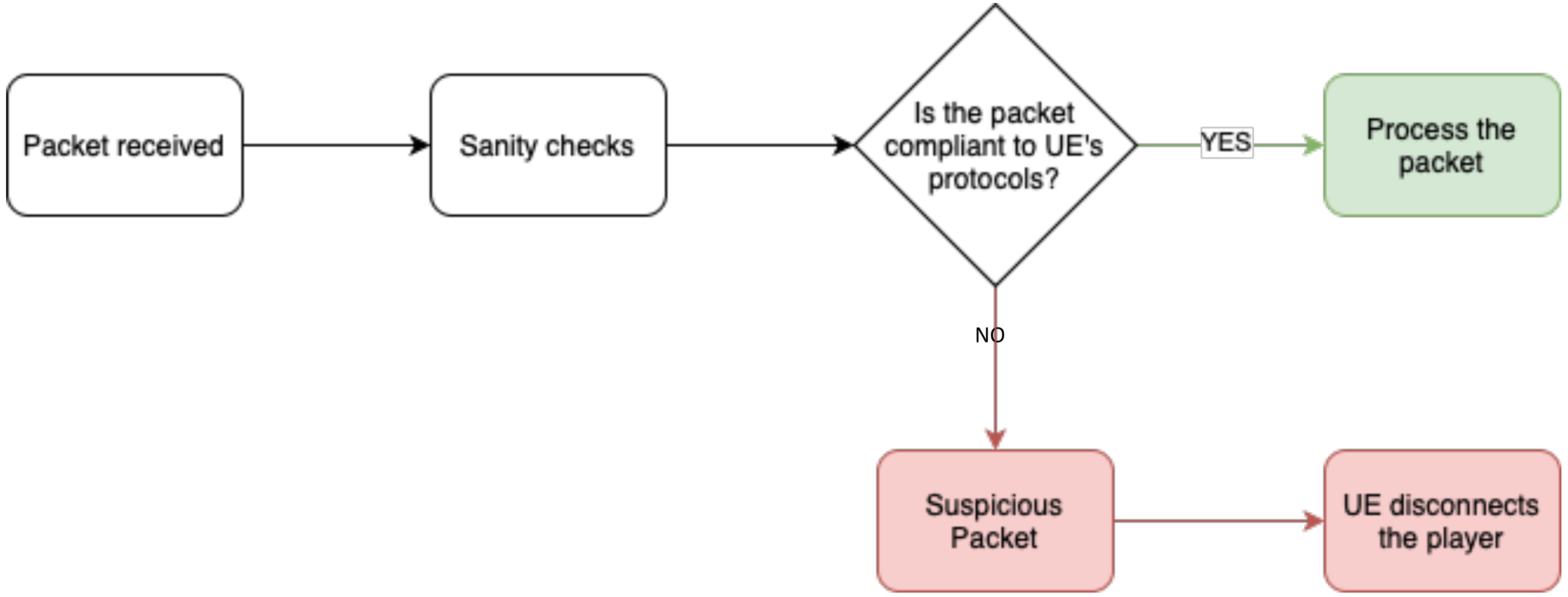
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine
- Disconnection: the client has to establish a new connection



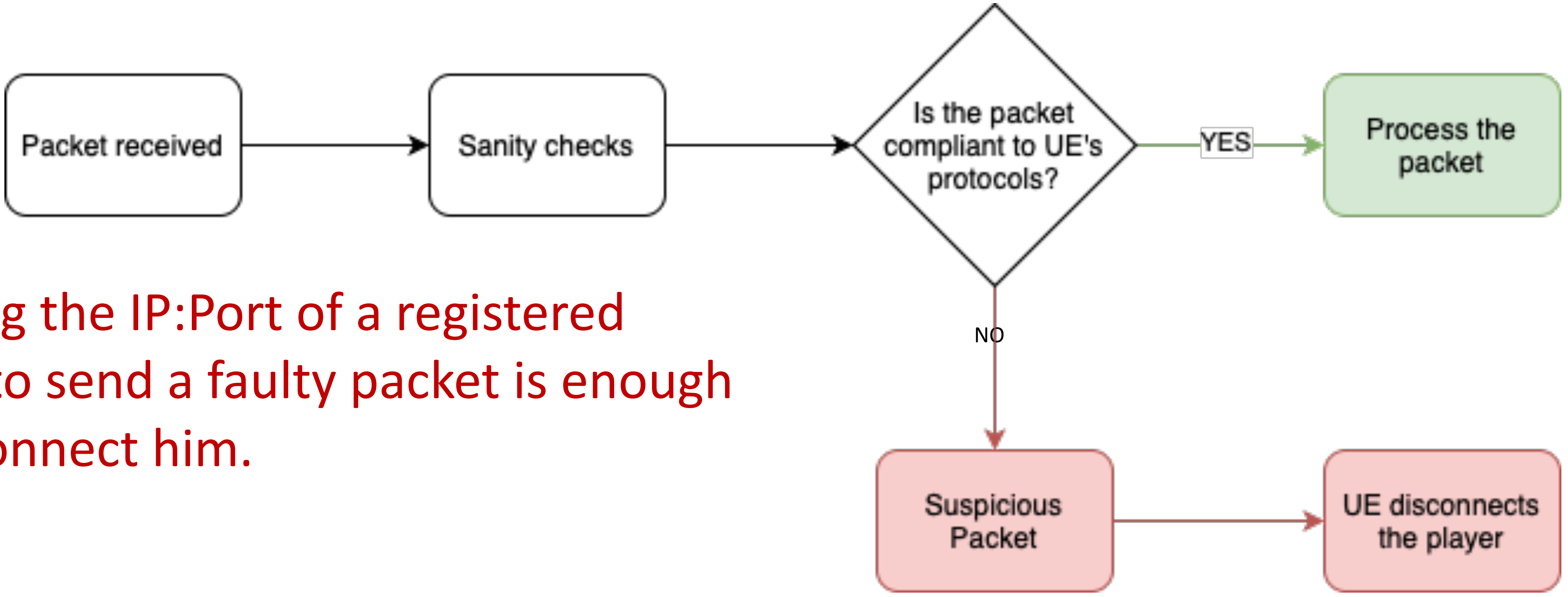
High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine
- Disconnection: the client has to establish a new connection
- Same behaviour for the client and the server



High-level overview of the vulnerability

- Related to the packets reception process in Unreal Engine
- Disconnection: the client has to establish a new connection
- Same behaviour for the client and the server



=> Using the IP:Port of a registered player to send a faulty packet is enough to disconnect him.

How to identify faulty packets?

- Static code analysis on Unreal Engine source code (using Doxygen [3])
  - Call/Caller Graphs
  - Inheritance Diagrams

=> Trace the paths leading to a disconnection

- Dynamic profiling on a toy game
  - Unreal Engine insight tools
  - Tracing, logging in Unreal Engine’s source code

=> Trace the engine behaviour

[3] Doxygen, "Doxygen: Generate documentation from source code," [Online]. Available: https://www.doxygen.nl.

How to identify faulty packets?

- Static code analysis on Unreal Engine source code (using Doxygen [3])
  - Call/Caller Graphs
  - Inheritance Diagrams

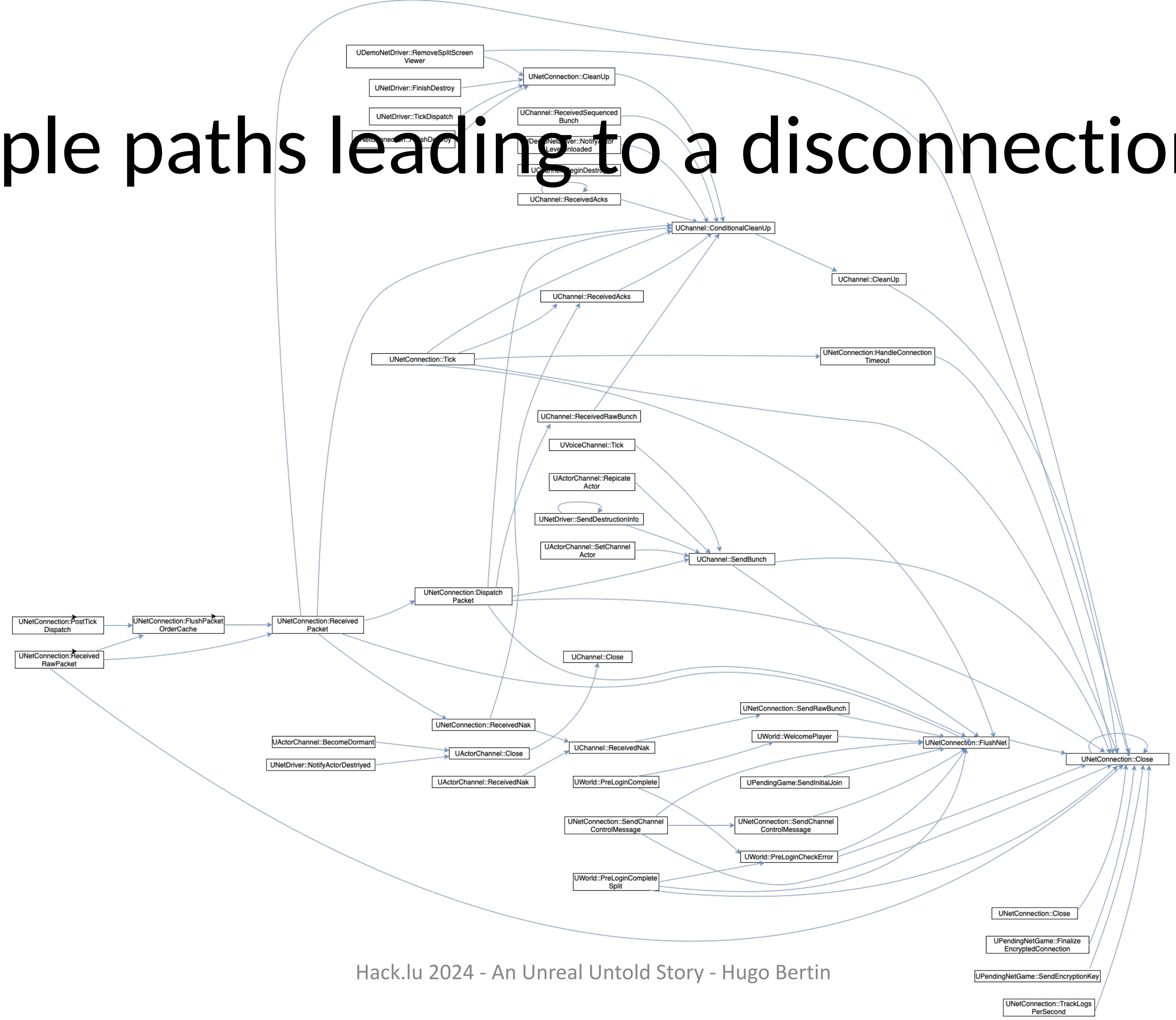
=> Trace the paths leading to a disconnection

- Dynamic profiling on a toy game
  - Unreal Engine insight tools
  - Tracing, logging in Unreal Engine’s source code

=> Trace the engine behaviour

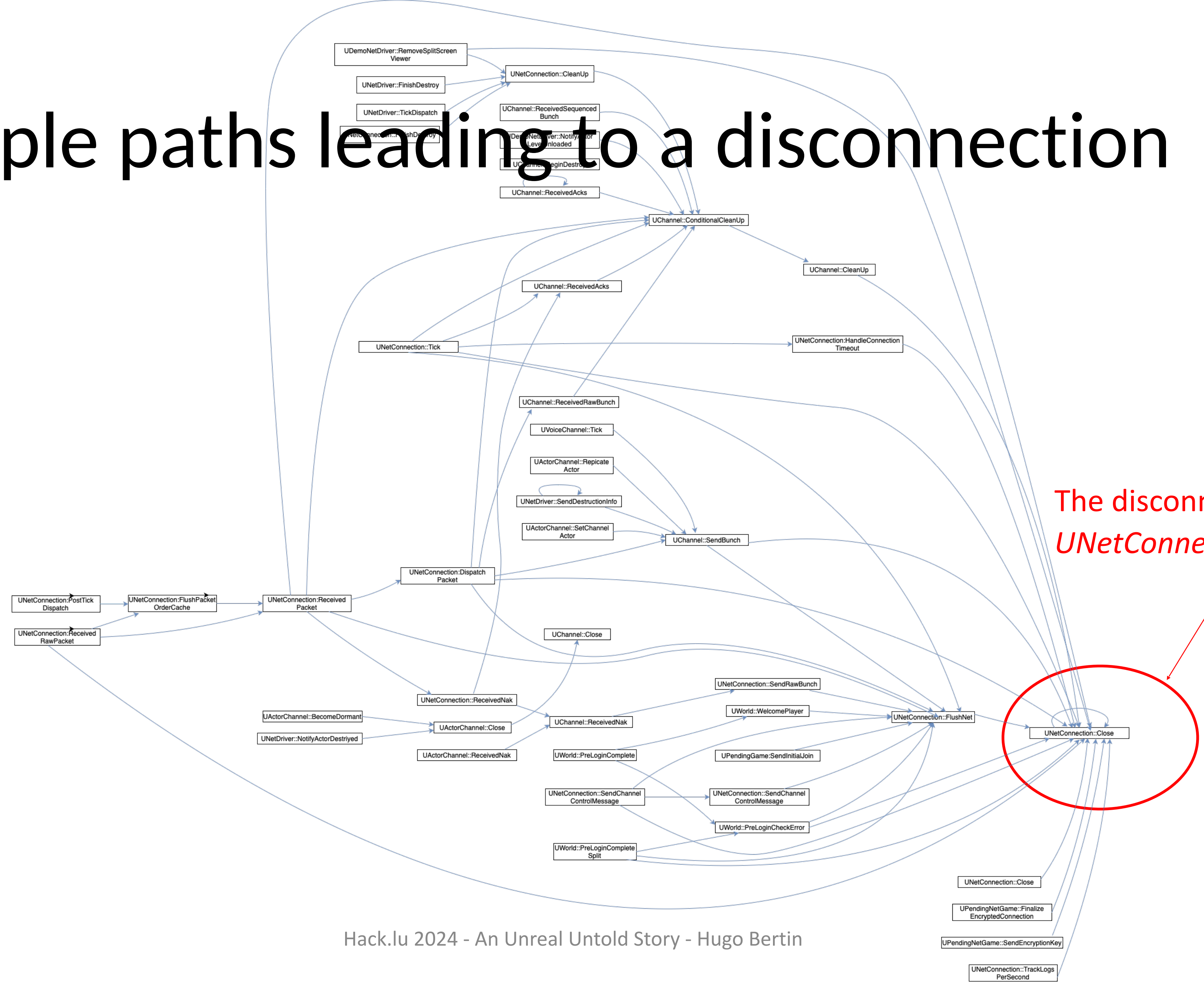
[3] Doxygen, "Doxygen: Generate documentation from source code," [Online]. Available: https://www.doxygen.nl.

Multiple paths leading to a disconnection



UDemoNetDriver::RemoveSplitScreen Viewer
UNetConnection::CleanUp
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UNetConnection::FinishDestroy
UChannel::ReceivedSequenced Bunch
UDemoNetDriver::NotifyActor LevelUnloaded
UChannel::BeginDestroy
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection:HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection:Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection:FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::SendRawBunch
UNetConnection::ReceivedNak
UWorld::WelcomePlayer
UActorChannel::BecomeDormant
UActorChannel::Close
UChannel::ReceivedNak
UNetConnection::FlushNet
UNetConnection::Close
UNetDriver::NotifyActorDestriyed
UActorChannel::ReceivedNak
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

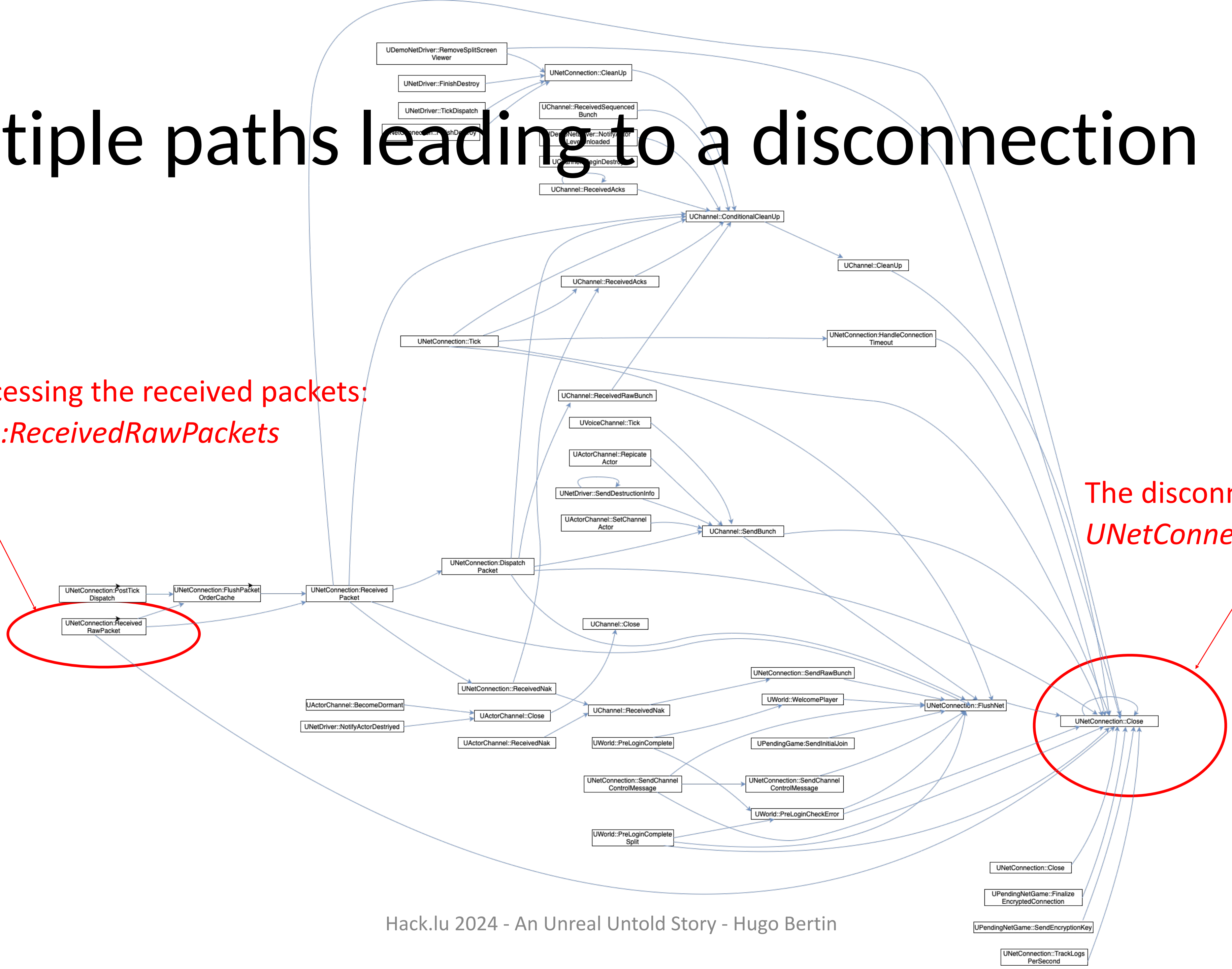
Multiple paths leading to a disconnection



UDemoNetDriver::RemoveSplitScreen Viewer
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UNetConnection::FinishDestroy
UNetConnection::CleanUp
UChannel::ReceivedSequenced Bunch
UDemoNetDriver::NotifyActor LevelUnloaded
UChannel::BeginDestroy
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection:HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::SendRawBunch
UNetConnection::ReceivedNak
UWorld::WelcomePlayer
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UChannel::ReceivedNak
UNetConnection::FlushNet
UNetConnection::Close
UActorChannel::ReceivedNak
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

The disconnection process:
UNetConnection::Close

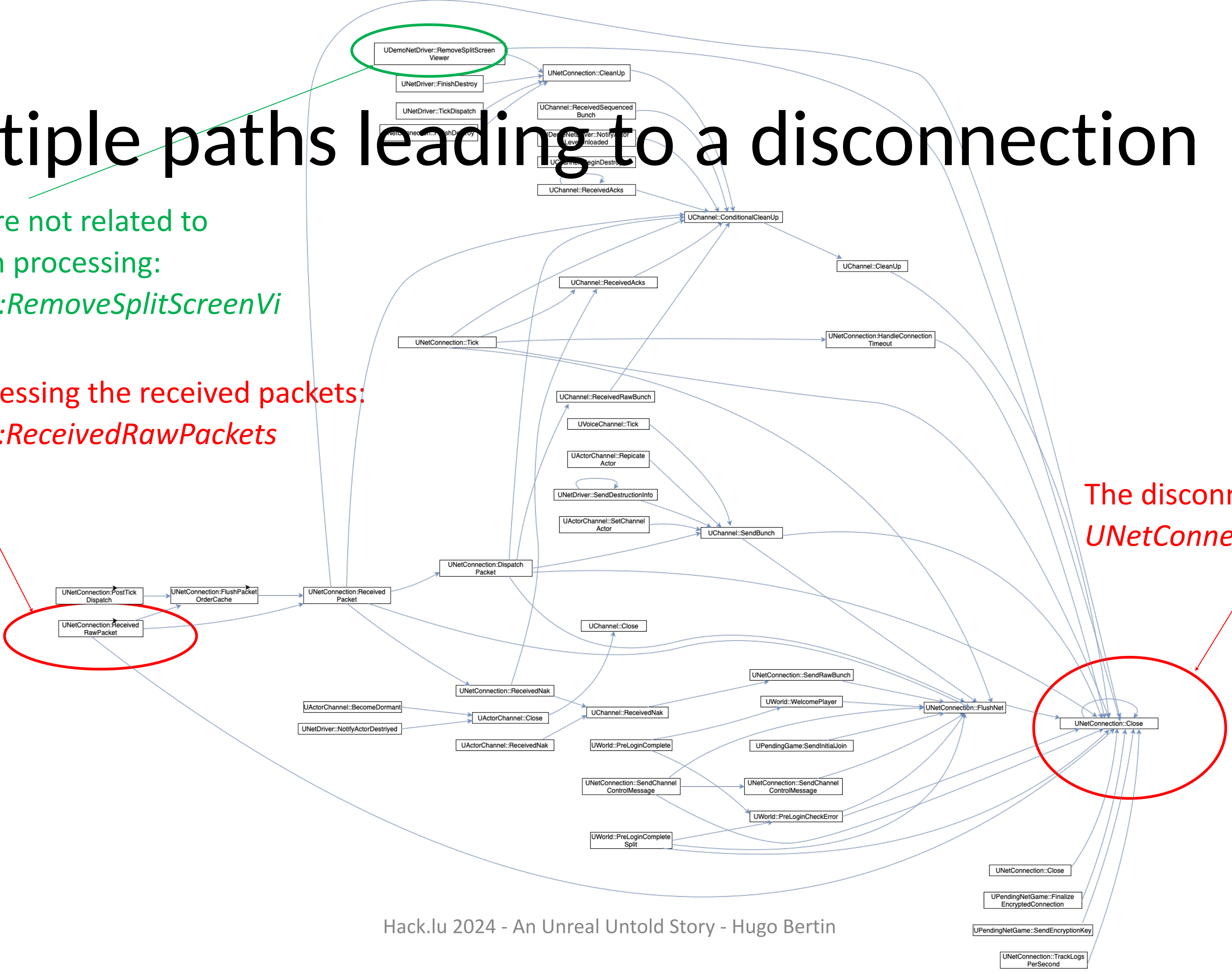
Multiple paths leading to a disconnection


UDemoNetDriver::RemoveSplitScreen Viewer
UNetConnection::CleanUp
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UChannel::ReceivedSequenced Bunch
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection::HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::ReceivedNak
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UActorChannel::ReceivedNak
UChannel::ReceivedNak
UNetConnection::SendRawBunch
UWorld::WelcomePlayer
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::FlushNet
UNetConnection::Close
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

The method processing the received packets:
UNetConnection::ReceivedRawPackets

The disconnection process:
UNetConnection::Close

Multiple paths leading to a disconnection



UDemoNetDriver::RemoveSplitScreen Viewer
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UNetConnection::FinishDestroy
UNetConnection::CleanUp
UChannel::ReceivedSequenced Bunch
UDemoNetDriver::NotifyActor LevelUnloaded
UChannel::BeginDestroy
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection::HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::ReceivedNak
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UActorChannel::ReceivedNak
UChannel::ReceivedNak
UNetConnection::SendRawBunch
UWorld::WelcomePlayer
UNetConnection::FlushNet
UNetConnection::Close
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

Some methods are not related to packets reception processing: *UDemoNetDriver:RemoveSplitScreenViewer*

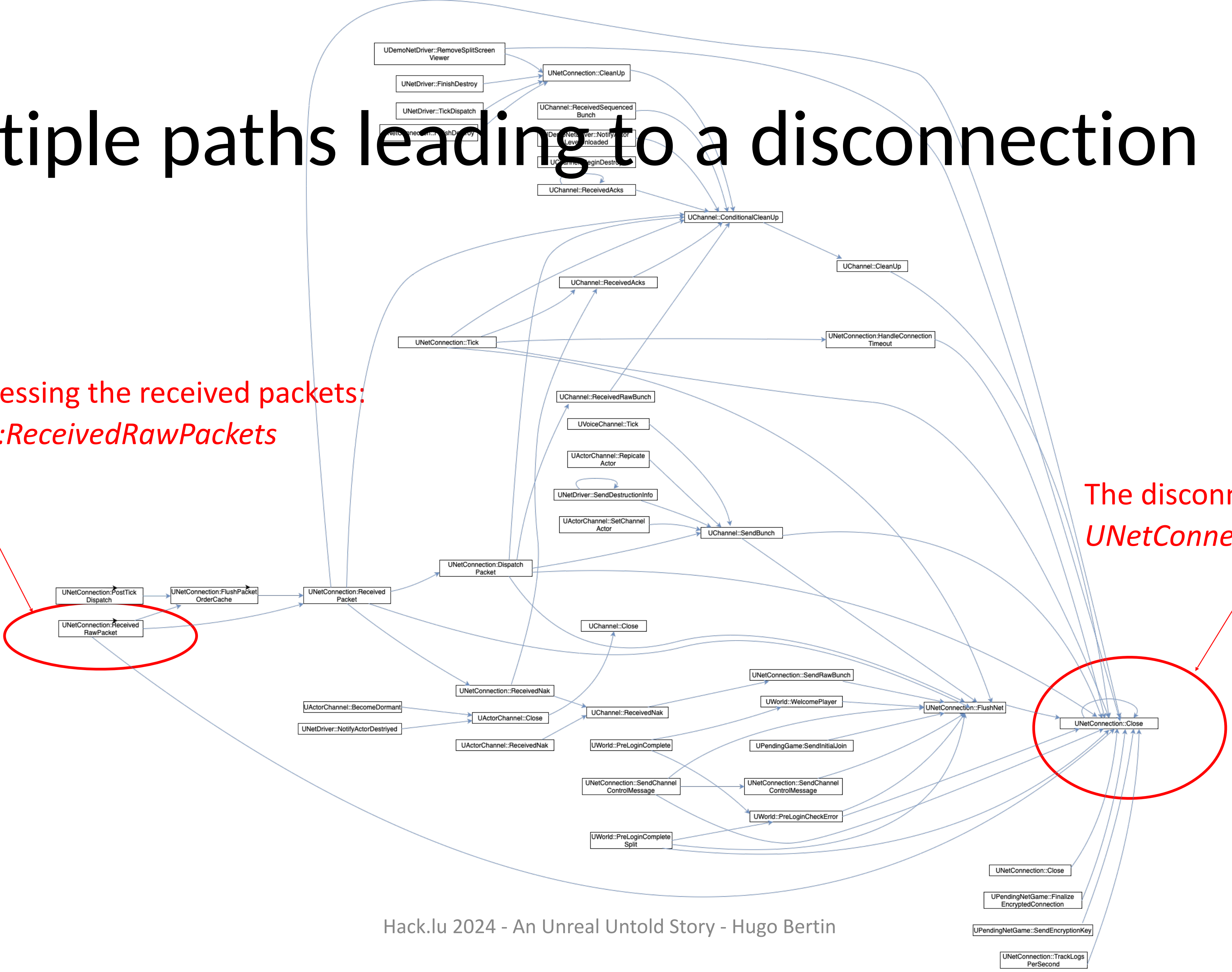
The method processing the received packets: *UNetConnection::ReceivedRawPackets*

The disconnection process: *UNetConnection::Close*

Multiple paths leading to a disconnection

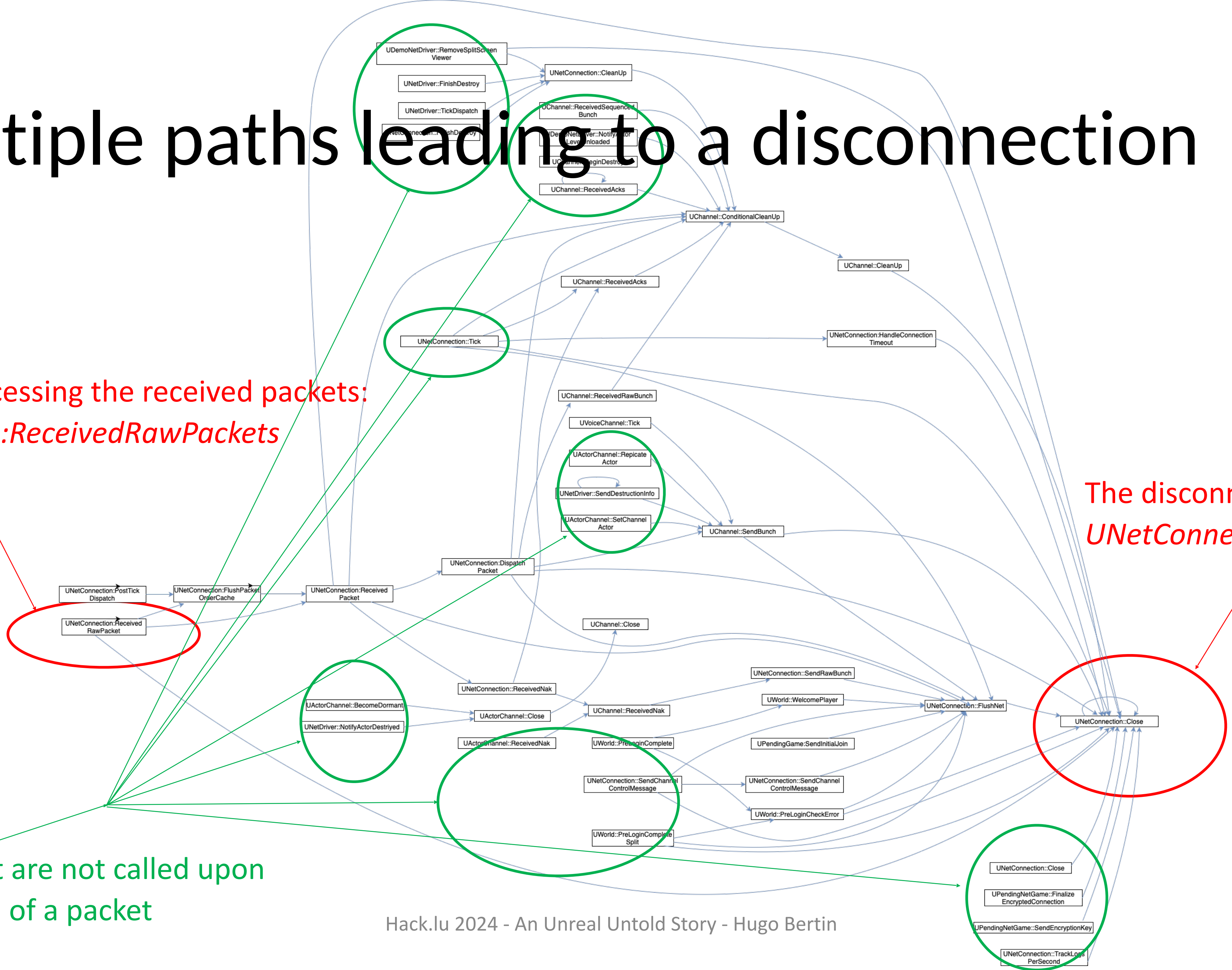
The method processing the received packets:
UNetConnection::ReceivedRawPackets

The disconnection process:
UNetConnection::Close



UDemoNetDriver::RemoveSplitScreen Viewer
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UNetConnection::CleanUp
UChannel::ReceivedSequenced Bunch
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection::HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::ReceivedNak
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UActorChannel::ReceivedNak
UChannel::ReceivedNak
UNetConnection::SendRawBunch
UWorld::WelcomePlayer
UNetConnection::FlushNet
UNetConnection::Close
UWorld::PreLoginComplete
UPendingGame::SendInitialJoin
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

Multiple paths leading to a disconnection


UDemoNetDriver::RemoveSplitScreenViewer
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UNetConnection::FlushDestroy
UNetConnection::CleanUp
UChannel::ReceivedSequencedBunch
UDemoNetDriver::NotifyActorLevelUnloaded
UChannel::BeginDestroy
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection::HandleConnectionTimeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::RepicateActor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannelActor
UChannel::SendBunch
UNetConnection::DispatchPacket
UNetConnection::PostTickDispatch
UNetConnection::FlushPacketOrderCache
UNetConnection::ReceivedPacket
UNetConnection::ReceivedRawPacket
UChannel::Close
UNetConnection::ReceivedNak
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UActorChannel::ReceivedNak
UChannel::ReceivedNak
UNetConnection::SendRawBunch
UWorld::WelcomePlayer
UNetConnection::FlushNet
UNetConnection::Close
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::SendChannelControlMessage
UNetConnection::SendChannelControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginCompleteSplit
UNetConnection::Close
UPendingNetGame::FinalizeEncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLossPerSecond

The method processing the received packets: *UNetConnection::ReceivedRawPackets*

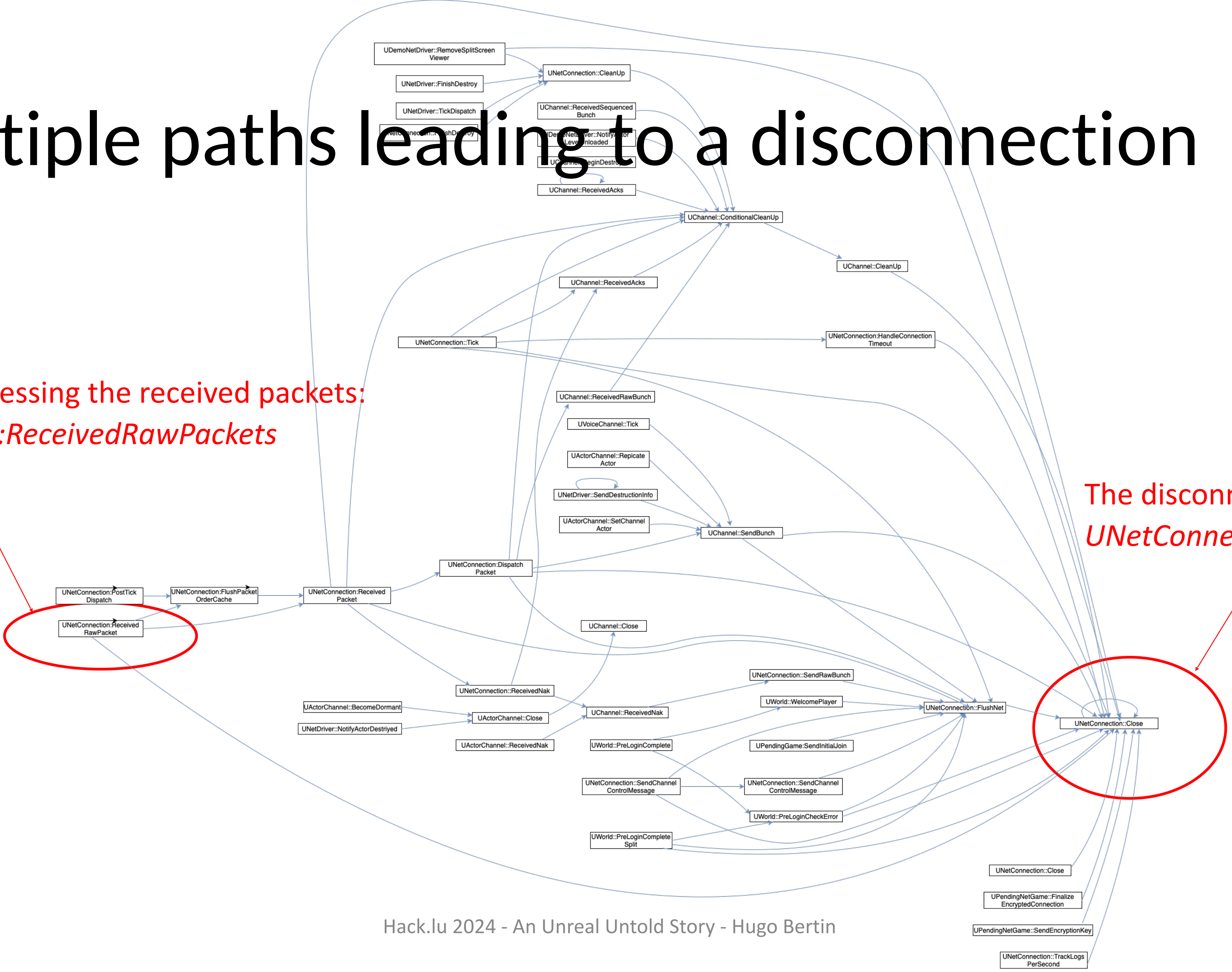
The disconnection process: *UNetConnection::Close*

Methods that are not called upon the reception of a packet

Multiple paths leading to a disconnection

The method processing the received packets:
UNetConnection::ReceivedRawPackets

The disconnection process:
UNetConnection::Close


UDemoNetDriver::RemoveSplitScreen Viewer
UNetConnection::CleanUp
UNetDriver::FinishDestroy
UNetDriver::TickDispatch
UChannel::ReceivedSequenced Bunch
UChannel::ReceivedAcks
UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UNetConnection::Tick
UNetConnection::HandleConnection Timeout
UChannel::ReceivedRawBunch
UVoiceChannel::Tick
UActorChannel::Repicate Actor
UNetDriver::SendDestructionInfo
UActorChannel::SetChannel Actor
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::PostTick Dispatch
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UChannel::Close
UNetConnection::SendRawBunch
UNetConnection::ReceivedNak
UActorChannel::BecomeDormant
UNetDriver::NotifyActorDestriyed
UActorChannel::Close
UChannel::ReceivedNak
UWorld::WelcomePlayer
UNetConnection::FlushNet
UNetConnection::Close
UActorChannel::ReceivedNak
UWorld::PreLoginComplete
UPendingGame:SendInitialJoin
UNetConnection::SendChannel ControlMessage
UNetConnection::SendChannel ControlMessage
UWorld::PreLoginCheckError
UWorld::PreLoginComplete Split
UNetConnection::Close
UPendingNetGame::Finalize EncryptedConnection
UPendingNetGame::SendEncryptionKey
UNetConnection::TrackLogs PerSecond

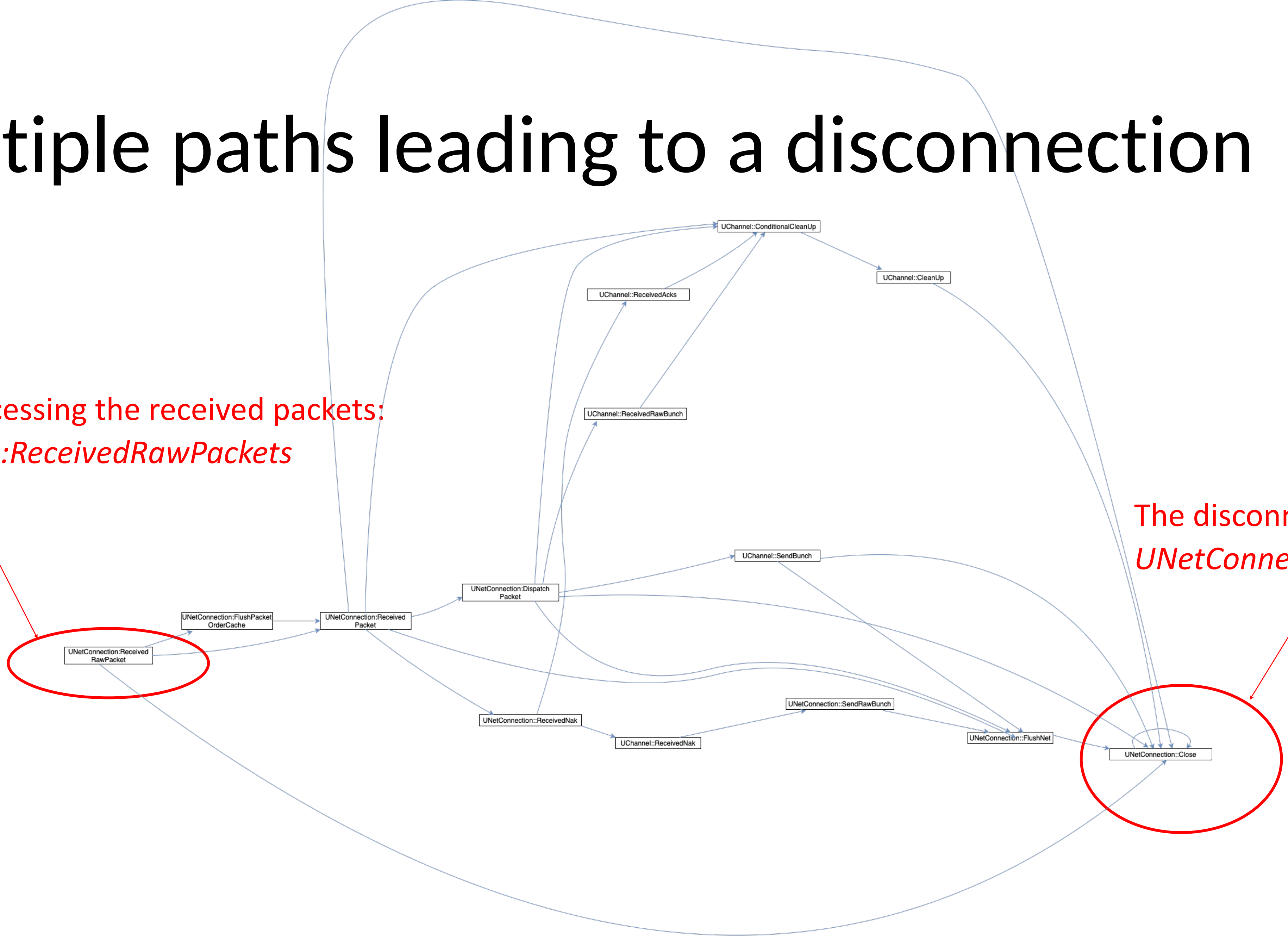
Multiple paths leading to a disconnection

<span style="color:red">The method processing the received packets:</span>

UNetConnection::ReceivedRawPackets

The disconnection process:

UNetConnection::Close


UChannel::ConditionalCleanUp
UChannel::CleanUp
UChannel::ReceivedAcks
UChannel::ReceivedRawBunch
UChannel::SendBunch
UNetConnection::Dispatch Packet
UNetConnection::FlushPacket OrderCache
UNetConnection::Received Packet
UNetConnection::Received RawPacket
UNetConnection::ReceivedNak
UChannel::ReceivedNak
UNetConnection::SendRawBunch
UNetConnection::FlushNet
UNetConnection::Close

Crafting a faulty packet to disconnect a player

Crafting a faulty packet to disconnect a player

- We identified 5 different ways of crafting a packet not passing the sanity checks

Crafting a faulty packet to disconnect a player

- We identified 5 different ways of crafting a packet not passing the sanity checks
- None of these packets require knowledge about a previous packet

Crafting a faulty packet to disconnect a player

- We identified 5 different ways of crafting a packet not passing the sanity checks
- None of these packets require knowledge about a previous packet
- Only information required:
  - Source IP address / Port number
  - Destination IP address / Port number

Direction	Source	Destination
Client -> Server	Client's IP:Port	Server's IP:Port
Server -> Client	Server's IP:Port	Client's IP:Port

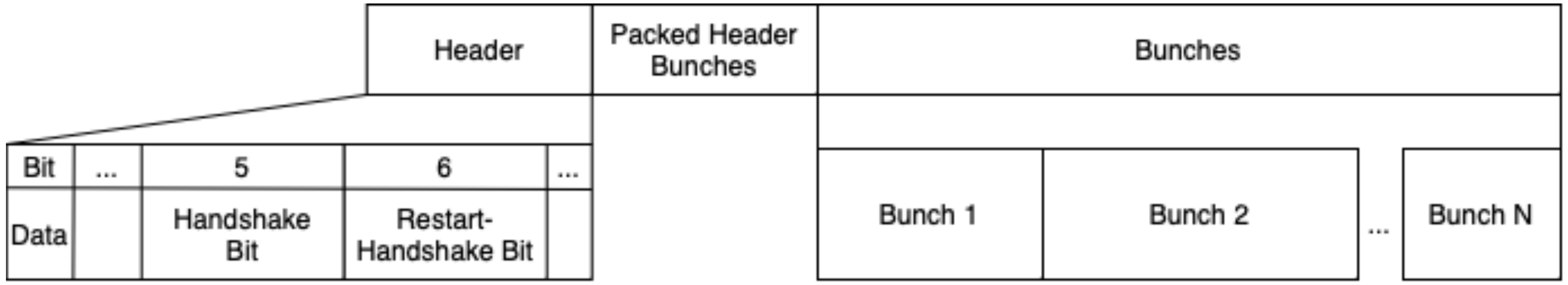
Can encryption protecting against crafting faulty packets?

Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

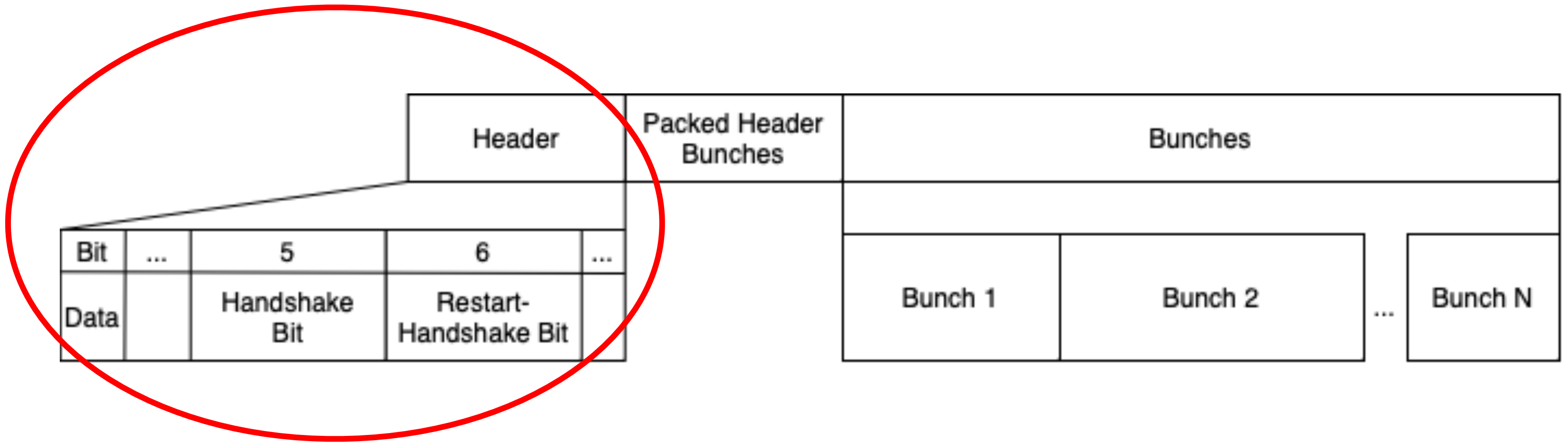


Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

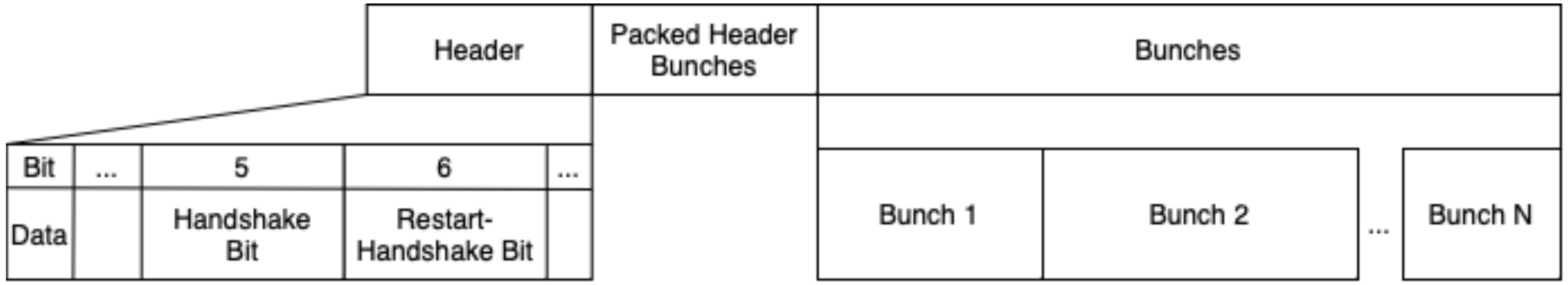


Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

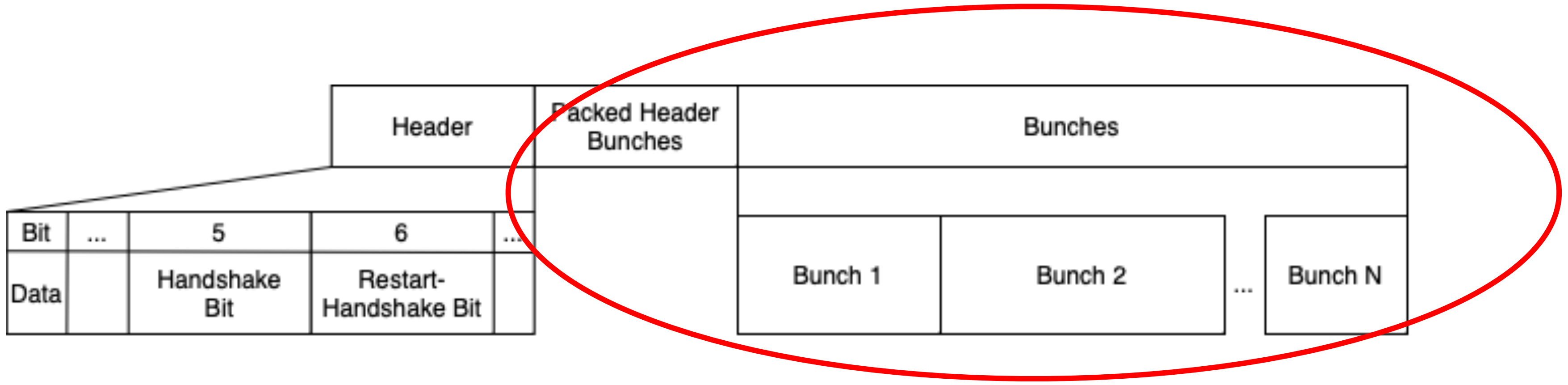


Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

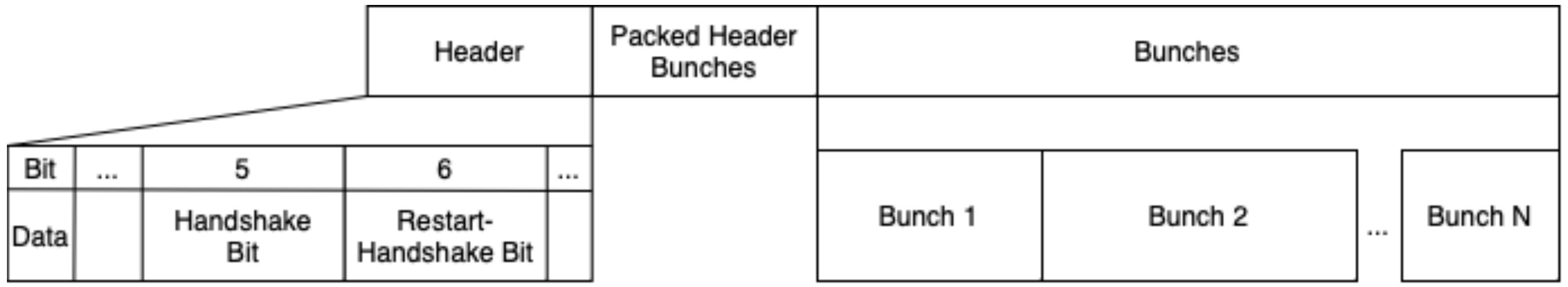


Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed



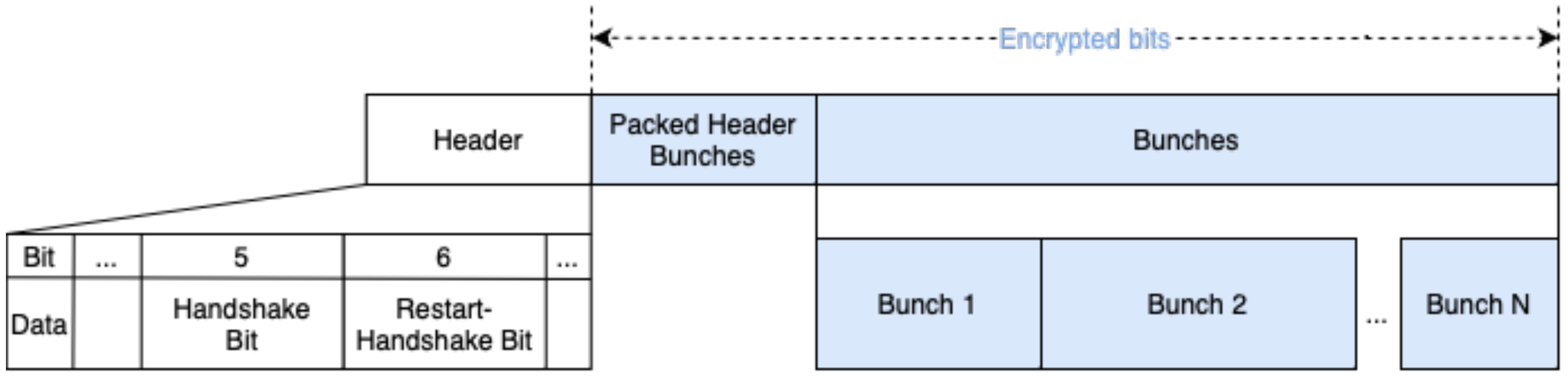
Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

- Encryption is not applied to the *Header*



Can encryption protecting against crafting faulty packets?

Packets have the following layout:

- *Header* contains protocol information
- *Packed Header Bunches* and *Bunches* contain game information

  => Sensitive information that should not be exposed

- Encryption is not applied to the *Header*

However, the header is attackable

=> Encryption is not a solution

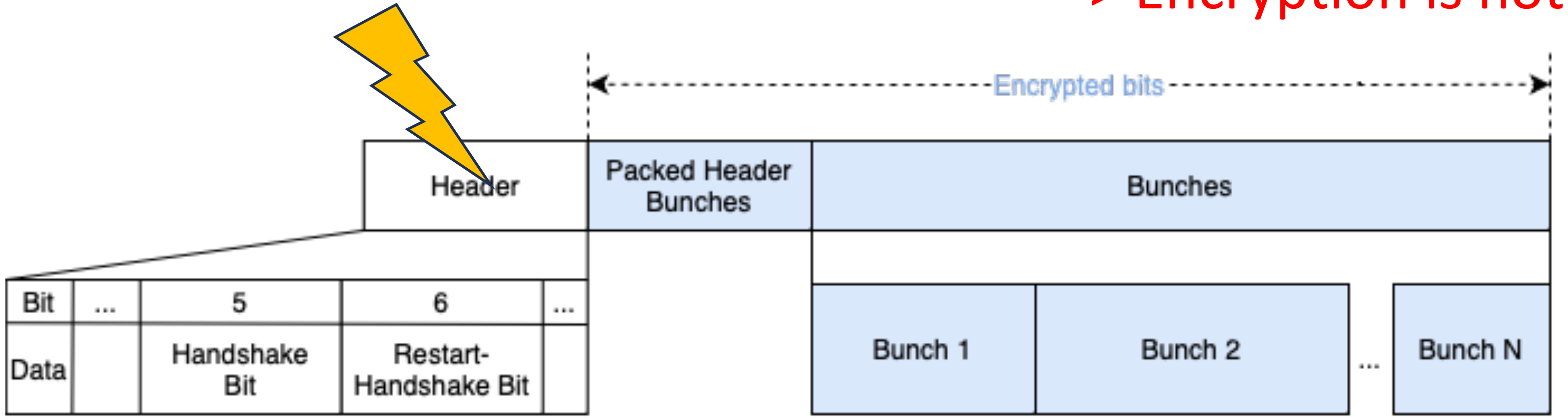


Table of Contents

Demonstration 1: Valorant

Setting at the beginning of the video:

- Valorant match within a LAN (Esport scenario).
- Attackers: They have to place a bomb, taking 30 seconds to detonate.
- Defenders: They defend against the bomb, they can defuse the bomb (taking 7 seconds).
- Viewing the screen of a defender.

Scenario in the video:

1. Attackers place the bomb.
2. A defender kills the last attacker alive.
3. Defenders are likely to win: they have time to defuse the bomb.
4. Attackers launch an attack.
5. Effect: Defenders are repeatedly disconnected. When they re-join the game, they have to start the defusing action again.

Demonstration 1: Valorant

https://youtu.be/dWy-L1ZQZw4

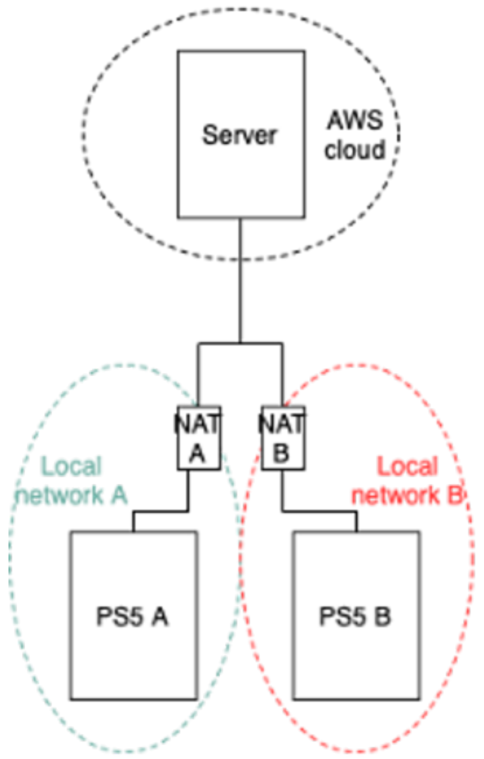
Demonstration 2: Fortnite

Setting at the beginning of the video:

- Fortnite game on PS5
- Online game scenario
- 3 players A, B and C (all opponents)

Scenario in the video:

1. Player B kills player C, the information is displayed on the screen.
2. Player A uses this information to try to connect to B to get his IP, using the PSN P2P calls.
3. Player B answers the incoming call.
4. Player A launches an attack.
5. Effect: Player B gets kicked out of the game.



Demonstration 2: Fortnite

https://youtu.be/NJEWW9E2rXk

Table of Contents

Responsible Disclosure

- Submitted to 5 video game studios impacted by the vulnerability
  - Epic Games (Fortnite), Riot Games (Valorant), Sea of Thieves (Rare/Xbox Game studio), Krafton (PUBG), Embark (The Finals)
- Detailed explanation on how to reproduce the attacks
  - Video demonstrations
  - A server they could use to reproduce the attacks

Responsible Disclosure

- Reports considered as informative: “Hard to reproduce in a concrete real-life case”

Responsible Disclosure

- Reports considered as informative: “Hard to reproduce in a concrete real-life case”
    i. “It is hard to find the IP address on the Internet.”

Responsible Disclosure

- Reports considered as informative: “Hard to reproduce in a concrete real-life case”
  i. “It is hard to find the IP address on the Internet.”
  ii. “It is hard to spoof an IP addresses on the Internet.”

Responsible Disclosure

- Reports considered as informative: “Hard to reproduce in a concrete real-life case”
    1. “It is hard to find the IP address on the Internet.”
    2. “It is hard to spoof an IP addresses on the Internet.”
    3. “Players are protected by firewalls.”

i. Finding the IP address of an opponent

i. Finding the IP address of an opponent

- Players can play within the same LAN
  - Esport
- In a LAN
  - Finding the victim’s IP addresses is not needed
    - Broadcast MAC adress: FF:FF:FF:FF:FF:FF
    - Broadcast IP address: 255.255.255.255
  - The only information needed: victim’s Port number
    - Possible ports: dynamic port range (49152 to 65535)

    => Easy to bruteforce
  - To avoid being incriminated, the sender can:
    - Spoof another source address (MAC layer)
    - Target himself

	Source	Dest
Port	Server’s Port	Bruteforce
IP	Server’s IP	255.255.255.255
MAC	00-B0-D0-63-C2-26	FF:FF:FF:FF:FF:FF

i. Finding the IP address of an opponent

In an online game scenario:

i. Finding the IP address of an opponent

In an online game scenario:

- Other mediums can be exploited
  - Voice communication channels
  - Redirection in text cheats or Discord servers
  - Gamertag-IP databases
  - Social engineering

i. Finding the IP address of an opponent

In an online game scenario:

- Other mediums can be exploited
  - Voice communication channels
  - Redirection in text cheats or Discord servers
  - Gamertag-IP databases
  - Social engineering
- In our demonstration on Fortnite: Play Station Network
  - PlayStation’s voice communication channel
  - Peer-to-peer (P2P) architecture

  => Answering a call leaks IP address

ii. Hard to spoof an IP address on the Internet?

- IP spoofing: falsifying the source IP address
- Internet Service Providers (ISPs) theoretically implement Source Address Validation (SAV) as a countermeasure
  - Packets filtering implemented at the edge of Autonomous Systems
  - Ensures that the source IP address in a packet align with the network it is from

ii. Hard to spoof an IP address on the Internet?

- IP spoofing: falsifying the source IP address
- Internet Service Providers (ISPs) theoretically implement Source Address Validation (SAV) as a countermeasure
  - Packets filtering implemented at the edge of Autonomous Systems
  - Ensures that the source IP address in a packet align with the network it is from
- However, not everyone is doing the job!!
  - Some ISPs do not enforce SAV well
  - “69.8% of all the Autonomous Systems (ASes) in the Internet do not filter spoofed packets” [4]

[4] Tianxiang Dai and Haya Shulman. 2021. SMap: Internet-wide Scanning for Spoofing. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21). Association for Computing Machinery, New York, NY, USA, 1039–1050. https://doi.org/10.1145/3485832.3485917

ii. Hard to spoof an IP address on the Internet?

- How to find them? The Spoofer Project [5]
  - Measurement platforms pinpointing Autonomous Systems (AS) where SAV is not well-implemented.
  - Results are published on their website.
  - Steps to set up a server:
    1. Identify a vulnerable AS with spoofer [6].
    2. Contact the service provider to rent a dedicated server.
    3. You are now able to spoof IP addresses!!

[5] Center for Applied Internet Data Analysis, "Spoofer Project," *CAIDA*. [Online]. Available: https://www.caida.org/projects/spoofer/.
[6] Center for Applied Internet Data Analysis, "Spoofer: Recent Tests," *CAIDA*. [Online]. Available: https://spoofer.caida.org/recent_tests.php.

iii. Players are protected by firewalls

iii. Players are protected by firewalls

- Players are playing behind firewalls, the address exposed is owned by the firewall and not by the victim

	Source	Dest
Port		
IP		

iii. Players are protected by firewalls

- Players are playing behind firewalls, the address exposed is owned by the firewall and not by the victim
- Players are continuously communicating with the game server
    - Any packet with the right quadruple Source IP:Port; Destination IP:Port is accepted by the firewall and rooted to the player

	Source	Dest
Port		
IP		

iii. Players are protected by firewalls

- Players are playing behind firewalls, the address exposed is owned by the firewall and not by the victim
- Players are continuously communicating with the game server
  - Any packet with the right quadruple Source IP:Port; Destination IP:Port is accepted by the firewall and rooted to the player
- The game server IP:Port are the same for all the players

  => The Source IP:Port is known

	Source	Dest
Port	Server's Port	
IP	Server's IP	

iii. Players are protected by firewalls

- Players are playing behind firewalls, the address exposed is owned by the firewall and not by the victim
- Players are continuously communicating with the game server
  - Any packet with the right quadruple Source IP:Port; Destination IP:Port is accepted by the firewall and rooted to the player
- The game server IP:Port are the same for all the players

  => The Source IP:Port is known

	Source	Dest
Port	Server's Port	
IP	Server's IP	Firewall's IP

iii. Players are protected by firewalls

- Players are playing behind firewalls, the address exposed is owned by the firewall and not by the victim
- Players are continuously communicating with the game server
  - Any packet with the right quadruple Source IP:Port; Destination IP:Port is accepted by the firewall and rooted to the player
- The game server IP:Port are the same for all the players

  => The Source IP:Port is known
- We need to find the port used by the player
  - Range of possible ports 0-65535
  - Packets of 48 bytes on wire are enough to disconnect
  - At 25 Mbps, it takes 1 s to bruteforce

  => Can be bruteforced easily

	Source	Dest
Port	Server's Port	Bruteforce
IP	Server's IP	Firewall's IP

Potential mitigations

- Modifying Unreal Engine source code to:
  - Encrypt the whole packet
  - Drop suspicious packets (without disconnecting)
- Securing the transport layer
  - E.g. using TLS (Transport Layer Security) providing authentication, confidentiality and integrity to TCP, or UDP with DTLS.
- Despite reporting the vulnerability to Epic Games, no fix was observed
- Unreal Engine’s code is available => Game developers can directly implement their modifications.

Table of Contents

Conclusion

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

=> Enforcing security at the engine level is primordial to strengthen the overall game industry.

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

=> Enforcing security at the engine level is primordial to strengthen the overall game industry.

- We disclosed and explained new DoS attacks, which are easy to put in practice.

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

=> Enforcing security at the engine level is primordial to strengthen the overall game industry.

- We disclosed and explained new DoS attacks, which are easy to put in practice.

=> Eye-opener for that community. The erroneous reply form the the video games editors shows a lack of awareness and consideration about the network-level security in online games.

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

=> Enforcing security at the engine level is primordial to strengthen the overall game industry.

- We disclosed and explained new DoS attacks, which are easy to put in practice.

=> Eye-opener for that community. The erroneous reply form the the video games editors shows a lack of awareness and consideration about the network-level security in online games.

- Potential impact beyond video games. Unreal Engine also used in: VR, digital twins, automotive HMI…
  - It might lead to a more critical issue tomorrow.

Conclusion

- While being powerful tools, game engine inadvertently broaden the scope of vulnerabilities.

=> Enforcing security at the engine level is primordial to strengthen the overall game industry.

- We disclosed and explained new DoS attacks, which are easy to put in practice.

=> Eye-opener for that community. The erroneous reply form the the video games editors shows a lack of awareness and consideration about the network-level security in online games.

- Potential impact beyond video games. Unreal Engine also used in: VR, digital twins, automotive HMI…
  - It might lead to a more critical issue tomorrow.

=> Extremely important to let people know about it and fix it.

Thank you for your attention!!

Questions?

hugo.bertin35@gmail.com

linkedin.com/in/bertin-hugo/

References

[1] A. Raza, “Ali Raza on LinkedIn: #gaming #esports #investment #tech | 100 comments,” *Linkedin.com*, Oct. 06, 2024. https://www.linkedin.com/feed/update/urn:li:activity:7248566123669925888/. [Accessed: Oct. 7, 2024].

[2] S. Nordmark and J. Heath, “The 10 Largest Prize Pools in Esports,” Dot Esports, Aug. 28, 2019. https://dotesports.com/general/news/biggest-prize-pools-esports-14605. [Accessed: Oct. 7, 2024].

[3] Doxygen, "Doxygen: Generate documentation from source code," [Online]. Available: https://www.doxygen.nl. [Accessed: Oct. 7, 2024].

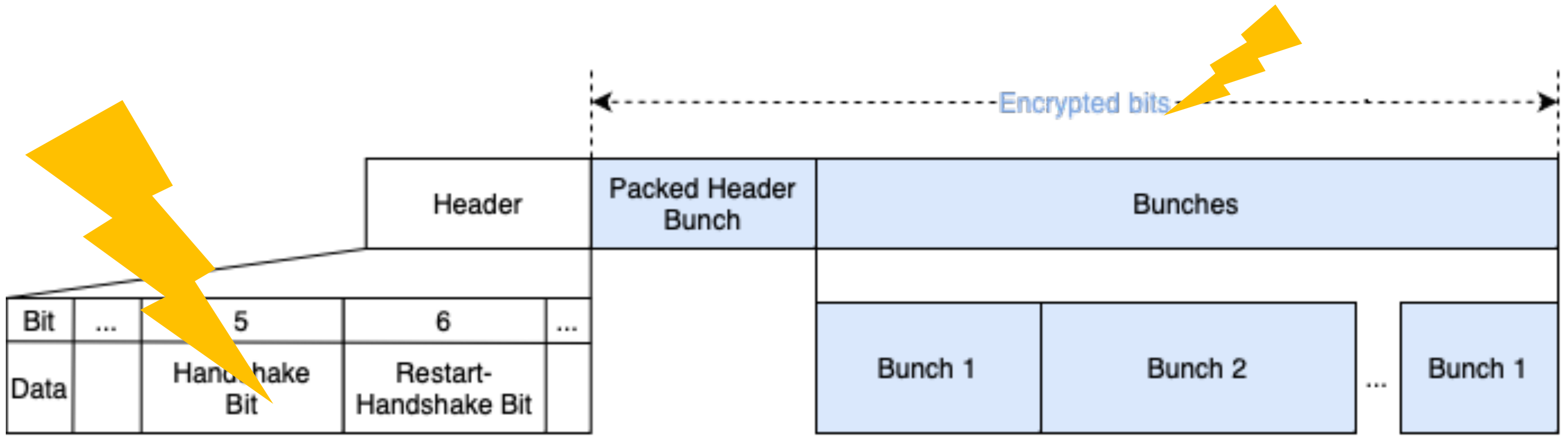
[4] Tianxiang Dai and Haya Shulman. 2021. SMap: Internet-wide Scanning for Spoofing. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21). Association for Computing Machinery, New York, NY, USA, 1039–1050. https://doi.org/10.1145/3485832.3485917

[5] Center for Applied Internet Data Analysis, "Spoofer Project," *CAIDA*. [Online]. Available: https://www.caida.org/projects/spoofer/. [Accessed: Oct. 7, 2024].

[6] Center for Applied Internet Data Analysis, "Spoofer: Recent Tests," *CAIDA*. [Online]. Available: https://spoofer.caida.org/recent_tests.php. [Accessed: Oct. 7, 2024].

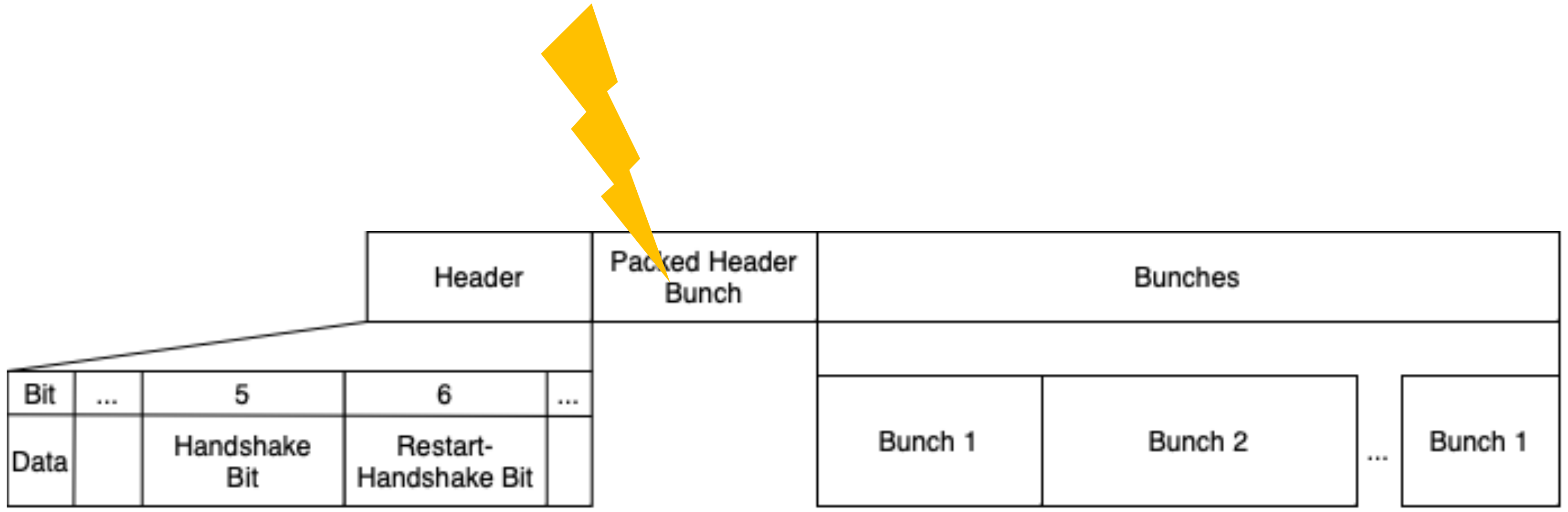
Multiple paths leading to a disconnection

Method	Condition to close	Exploit
UNetConnection::ReceivedRawPacket	PacketHandler returns an error processing the packet: • Handshake packet handler: *StatelessConnectHandlerComponent* • Decryption handlers	• Handshake bit to 1, packet not compliant to the handshake protocol • If using encryption, garbage packet



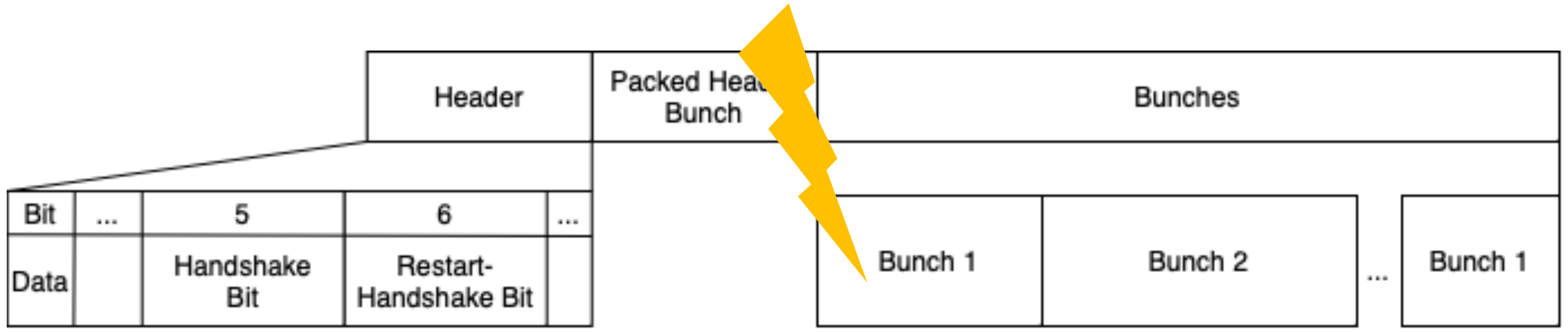
Multiple paths leading to a disconnection

Method	Condition to close	Exploit
UNetConnection::ReceivedPacket	Wrong Acked Sequence (!=Last Notified PacketId)	Bad information in the Packed Header



Multiple paths leading to a disconnection

Method	Condition to close	
UNetConnection::DispatchPacket	Bad channel index	Bad information in the Bunch


Header
Packed Hea
Bunch
Bunches
Bit
...
5
6
...
Data
Handshake Bit
Restart-Handshake Bit
Bunch 1
Bunch 2
...
Bunch 1