NetSim®

Accelerate Network R & D

Cyber

Simulate network attacks on power systems

TETCOS LLP

The information contained in this document represents the current view of TETCOS LLP on the issues discussed as of the date of publication. Because TETCOS LLP must respond to changing market conditions, it should not be interpreted to be a commitment on the part of TETCOS LLP, and TETCOS LLP cannot guarantee the accuracy of any information presented after the date of publication.

This manual is for informational purposes only.

The publisher has taken care in the preparation of this document but makes no expressed or implied warranty of any kind and assumes no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

Warning! DO NOT COPY, redistribute, or modify this document without the prior written consent of TETCOS LLP.

Copyright in the whole and every part of this manual belongs to TETCOS LLP and may not be used, sold, transferred, copied or reproduced in whole or in part in any manner or in any media to any person, without the prior written consent of TETCOS LLP. If you use this manual you do so at your own risk and on the understanding that TETCOS LLP shall not be liable for any loss or damage of any kind.

TETCOS LLP may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from TETCOS LLP, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Rev 15.1 (V), Oct 2026, TETCOS LLP. All rights reserved.

All trademarks are property of their respective owner.

Contact us at

TETCOS LLP
214, 39th A Cross, 7th Main, 5th Block Jayanagar,
Bangalore - 560 041, Karnataka, INDIA.
Phone: +91 80 26630624
E-Mail: sales@tetcos.com
Visit: www.tetcos.com

Contents

1 Introduction **7**
- 1.1 Purpose of the Manual . . . 7
- 1.2 Intended Audience . . . 8
- 1.3 What NetSim Cyber Does . . . 8

2 NetSim Cyber Overview **9**
- 2.1 Operating Modes . . . 10
- 2.2 Single-System Deployment . . . 11
- 2.3 Multiple-System Deployment . . . 12
- 2.4 Threat Agent Concept . . . 13
- 2.5 Protocol-Aware Attack Emulation . . . 15

3 Supported Power-System Protocols **16**
- 3.1 IEEE C37.118 Synchrophasor . . . 17
- 3.2 IEC 61850 GOOSE, R-GOOSE, MMS, and SV . . . 19
- 3.3 Modbus TCP . . . 20
- 3.4 DNP3 . . . 22
- 3.5 IEC 60870-5-104 . . . 23
- 3.6 Custom TCP/IP Protocol Support . . . 25
- 3.7 Summary . . . 26

4 Simulator and Device Integration **27**
- 4.1 External Power-System Simulator Integration . . . 28
- 4.2 Built-in Open-Source Protocol Simulators . . . 30
- 4.3 Benefits of Built-in Simulators . . . 31
- 4.4 Benefits of External Simulator Integration . . . 31

5 Software Requirements and Installation **32**
- 5.1 Dependency Summary . . . 33
- 5.2 Python Runtime . . . 34
  - 5.2.1 Required Python Version . . . 34
  - 5.2.2 Installation Scope . . . 34
  - 5.2.3 Fresh Python Installation . . . 34
  - 5.2.4 Existing Python Installation . . . 35
  - 5.2.5 Multiple Python Versions . . . 36
- 5.3 Python Modules . . . 36
  - 5.3.1 Installing PySide6 . . . 36
  - 5.3.2 Installing pyqtgraph . . . 36
  - 5.3.3 Installing Modules Using the Correct Python . . . 37
- 5.4 .NET Runtime . . . 37
  - 5.4.1 Verifying .NET Runtime Installation . . . 37
  - 5.4.2 Symptoms of Missing .NET Runtime . . . 37
- 5.5 Administrator Privileges . . . 38
  - 5.5.1 Running NetSim Cyber as Administrator . . . 38
  - 5.5.2 Configuring NetSim Cyber to Always Run as Administrator . . . 38
  - 5.5.3 Running NetSim Cyber Client as Administrator . . . 39
- 5.6 Verification and Troubleshooting . . . 39
  - 5.6.1 Installation Verification Checklist . . . 39
  - 5.6.2 Workspace and PythonUserCode Folder Verification . . . 40

5.6.3 Administrator Permission Issues . . . 41
5.6.4 Troubleshooting Summary . . . 41
5.7 Workspace and PythonUserCode Folder . . . 42
5.7.1 Workspace . . . 42
5.7.2 PythonUserCode Folder . . . 43
5.7.3 L5-payload-modifier in Detail . . . 43
5.8 .NET Runtime . . . 44
5.8.1 Required Version . . . 44

6 Lab Setup and Network Configuration 44
6.1 Detailed Setup Configuration . . . 45
6.1.1 Method 1: Manual Route Configuration . . . 45
6.1.2 Method 2: Automatic Configuration using NetSim Cyber Client . . . 46
6.1.3 Method-3: Three-System Network Configuration for NetSim Cyber . . . 48
6.1.4 Connectivity Troubleshooting . . . 53

7 How to Use NetSim Cyber GUI 54
7.1 Overview . . . 54
7.2 End-to-End Workflow . . . 54
7.3 Step-by-Step Walkthrough . . . 54
7.3.1 Launch NetSim Cyber . . . 54
7.3.2 Select the Deployment Type . . . 55
7.3.3 Build the Scenario . . . 56
7.3.4 Configure the Application Traffic Filter . . . 56
7.3.5 Select the Attack Protocol . . . 57
7.3.6 Run the Simulation . . . 58
7.3.7 Configure and Launch the Attack . . . 59

8 Using the Protocol-Specific Attacker UIs 60
8.1 Purpose and Scope . . . 60
8.2 Synchrophasor Attacker . . . 60
8.2.1 Configuration Discovery . . . 61
8.2.2 Automatic Mode . . . 61
8.2.3 Manual Mode . . . 63
8.2.4 Replay Mode . . . 64
8.2.5 Profile Difference . . . 64
8.3 Modbus TCP Attacker . . . 64
8.3.1 Automatic Mode . . . 65
8.3.2 Manual Mode . . . 67
8.3.3 Replay Mode . . . 68
8.4 Troubleshooting . . . 68

9 Attack Library 69
9.1 Attack Categories . . . 70
9.2 Attack Types . . . 71
9.2.1 Noise Injection . . . 71
9.2.2 Ramp Attack . . . 72
9.2.3 Increment/Decrement Attack . . . 73
9.2.4 Pulse / Surge Attack . . . 74
9.2.5 Frequency / ROCOF Attack . . . 75
9.2.6 Time Synchronization Attack . . . 76
9.2.7 Replay Attack . . . 77

9.2.8 Fixed Delay Attack . . . 79
9.2.9 PER-Based Drop Attack . . . 80
9.2.10 Speed / Bandwidth Suppression Attack . . . 81
9.2.11 DoS Attack . . . 82
9.2.12 Virtual SYN Flood Attack . . . 83
9.2.13 Real SYN Flood Attack . . . 84
9.2.14 RST Flood Attack . . . 85
9.3 Custom Payload Modifier API . . . 86
9.3.1 Purpose . . . 86
9.3.2 API Entry Point . . . 86
9.3.3 Function Parameters . . . 87
9.3.4 Required Return Rule . . . 87
9.3.5 Safe Editing Guidelines . . . 88
9.3.6 Example Patterns . . . 88
9.3.7 Where This Fits in the Workflow . . . 89

10 Featured Examples **89**
10.1 IEEE C37.118 Synchrophasor Protocol . . . 90
10.1.1 IEEE C37.118 Standard Background . . . 90
10.1.2 Supported Synchrophasor Profiles in NetSim Cyber . . . 90
10.1.3 OpenPDC Reference . . . 90
10.1.4 NetSim Cyber Architecture . . . 91
10.1.5 Important Synchrophasor Run Order . . . 91
10.1.6 Demonstrated Attack Modes in This Example . . . 92
10.1.7 Experimental Procedure . . . 92
10.1.8 Results and Analysis . . . 96
10.1.9 Key Observations . . . 100
10.2 IEEE 9-Bus Integration . . . 100
10.2.1 Concept and Project Objective . . . 100
10.2.2 Study objectives . . . 101
10.2.3 System Architecture and Control Flow . . . 101
10.2.4 Data and Command Formats . . . 102
10.2.5 Simulink Integration . . . 103
10.2.6 TCP Example . . . 104
10.2.7 UDP Example . . . 105
10.2.8 NetSim Cyber Configuration for TCP and UDP . . . 106
10.2.9 TCP and UDP Baseline and Attack Results . . . 110
10.2.10 Modbus TCP Example . . . 112
10.2.11 TCP and UDP Packet Verification . . . 117
10.2.12 Run Procedure . . . 121
10.2.13 Example Package and Requirements . . . 122
10.2.14 Scope and Extensions . . . 123
10.2.15 Conclusion . . . 124
10.2.16 References . . . 124
10.3 Modbus Replay Attack Example . . . 124
10.3.1 Objective . . . 124
10.3.2 Saved Example Configuration . . . 124
10.3.3 Open the Saved Experiment . . . 125
10.3.4 Start Modbus Communication . . . 126
10.3.5 Record the FC04 Sequence . . . 130
10.3.6 Apply the Overvoltage Scenario . . . 133

10.3.7 Run the Replay Attack . . . 134
10.3.8 Verify Overvoltage Concealment . . . 136
10.3.9 Packet Verification . . . 137
10.3.10 Result . . . 141
10.3.11 Troubleshooting . . . 141
10.4 IEEE C37.118-2005 Synchrophasor Monitoring and Defense . . . 141
10.4.1 Overview . . . 141
10.4.2 Prerequisites . . . 142
10.4.3 The Unified Packet Verdict . . . 142
10.4.4 Example Setup . . . 144
10.4.5 Processing Workflow . . . 144
10.4.6 Detection Profile . . . 145
10.4.7 Configuration Reference . . . 146
10.4.8 Test Procedure . . . 146
10.4.9 Expected Results . . . 147
10.4.10 Packet-Level Behavior . . . 148
10.4.11 Error Handling . . . 149
10.4.12 Summary . . . 149

11 Limitations **149**

12 References **149**

1 Introduction

NetSim Cyber is a technology platform for modelling, simulating, and analysing cyber-attacks on power-system communication networks. It enables users to study how attacks on communication links, protocol messages, and measurement data can affect cyber-physical power systems such as SCADA networks, Synchrophasor systems, substation automation systems, and industrial control environments.

Modern power grids depend on communication networks for monitoring, control, protection, automation, and wide-area situational awareness. Devices such as PMUs, PDCs, IEDs, RTUs, PLCs, SCADA masters, and HMI systems exchange data using standard power-system protocols. While these communication systems improve observability and control, they also introduce cyber vulnerabilities that must be evaluated in a controlled and safe environment.

NetSim Cyber provides a laboratory environment where engineers, researchers, and security teams can emulate realistic attack scenarios on power-system protocols. It supports both simulated and real-device testbeds and can operate as a man-in-the-middle system between communicating endpoints. This allows users to observe the effect of attacks such as DoS, false data injection, replay, delay, packet modification, timestamp manipulation, and protocol-specific payload manipulation.

This manual describes the capabilities, installation requirements, supported protocols, lab setup procedures, attack configuration workflow, and example use cases for NetSim Cyber.

1.1 Purpose of the Manual

The purpose of this manual is to provide users with a practical and technical reference for using NetSim Cyber to simulate cyber-attacks on power-system communication networks.

This manual explains how to:

1. Understand the role of NetSim Cyber in cyber-physical power-system testing.
2. Set up NetSim Cyber in host-system and distributed test environments.
3. Interface NetSim Cyber with real devices, software simulators, and hardware-in-the-loop platforms.
4. Use built-in open-source protocol simulators for early-stage testing.
5. Configure power-system protocol traffic between source and destination nodes.
6. Insert a Threat Agent into the communication path.
7. Select and execute protocol-aware cyber-attacks.
8. Observe and analyze the impact of attacks on power-system measurements and communication flows.
9. Troubleshoot common setup, routing, dependency, and execution issues.

The manual is intended to serve both as a **user guide** and a **technology library**. As a user guide, it provides step-by-step procedures for installation, configuration, and operation. As a technology library, it explains the supported protocols, attack types, simulator integrations, and technical concepts required to understand NetSim Cyber experiments.

Copyright TETCOS LLP

1.2 Intended Audience

This manual is intended for users who work with power-system communication networks, cyber-physical power systems, or operational technology security.

The primary audience includes:

1. Power-system engineers working with PMUs, PDCs, IEDs, RTUs, PLCs, SCADA systems, or substation automation systems.
2. Protection and control engineers evaluating the impact of communication attacks on monitoring and control applications.
3. SCADA and OT engineers responsible for testing communication paths, protocol behavior, and network reliability.
4. Cybersecurity engineers assessing vulnerabilities in power-system protocols and control networks.
5. Researchers studying cyber-physical power systems, smart grids, wide-area monitoring, and attack detection.
6. Faculty, students, and laboratory instructors using NetSim Cyber for education and training.
7. Product validation teams testing power-system devices, simulators, or protocol implementations.

The document assumes that the reader has a basic understanding of power-system communication concepts and Windows-based network configuration. Prior experience with all supported protocols is not mandatory, but familiarity with SCADA, PMU/PDC systems, or industrial communication networks will help users understand the examples more easily.

1.3 What NetSim Cyber Does

NetSim Cyber enables users to model and simulate cyber-attacks on power-system communication traffic. It is designed to intercept, analyse, modify, and forward protocol traffic between communicating devices in a controlled test environment.

NetSim Cyber can be used with:

- Real power-system devices such as PMUs, PDCs, IEDs, RTUs, PLCs, and SCADA systems.
- Hardware-in-the-loop and real-time power-system simulators such as Typhoon HIL, OPAL-RT, and RTDS.
- Software-based simulators and protocol tools.
- Built-in open-source protocol simulators for early-stage protocol testing and attack validation.

NetSim Cyber supports major power-system and industrial communication protocols, including:

Copyright TETCOS LLP

- IEEE C37.118 Synchrophasor
- IEC 61850 GOOSE, Routed GOOSE, MMS, and Sampled Values
- Modbus TCP
- DNP3
- IEC 60870-5-104
- Custom TCP/IP-based protocols

In a typical experiment, NetSim Cyber is placed between a source device and a destination device. The source may be a PMU, RTU, PLC, IED, or simulator. The destination may be a PDC, SCADA master, HMI, subscriber, or control application. NetSim Cyber forwards traffic between these endpoints while a Threat Agent applies the selected attack logic.

The Threat Agent can perform protocol-aware operations such as:

- Modifying measurement values.
- Injecting noise, ramp, increment, decrement, or pulse disturbances.
- Manipulating frequency, ROCOF, timestamps, or protocol-specific fields.
- Replaying valid messages.
- Introducing delay or packet drops.
- Testing custom payload modification logic through user-defined scripts.

By using NetSim Cyber, users can evaluate how cyber-attacks affect power-system data quality, monitoring applications, control decisions, protection logic, and cyber-physical situational awareness. The tool helps users validate attack scenarios, generate datasets, test detection algorithms, and perform safe cybersecurity experiments without affecting production systems.

2 NetSim Cyber Overview

NetSim Cyber is a protocol-aware cyber-attack simulation and emulation platform for power-system communication networks. It allows users to model communication paths between power-system devices, insert an attack node into the communication path, and evaluate the effect of cyber-attacks on protocol traffic and power-system measurements.

In a typical NetSim Cyber experiment, a source device sends protocol traffic to a destination device. The source may be a PMU, IED, RTU, PLC, relay, software simulator, or hardware-in-the-loop system. The destination may be a PDC, SCADA master, HMI, subscriber, or monitoring application. NetSim Cyber places a **Threat Agent** between the source and destination. The Threat Agent intercepts selected traffic streams and applies user-configured attack logic before forwarding the modified traffic.

NetSim Cyber supports experiments using both real devices and built-in protocol simulators. This allows users to begin with a simple same-system setup and then move to more realistic distributed testbeds involving separate physical systems, HIL platforms, or field devices.

Copyright TETCOS LLP



Figure 2-1: *NetSim Cyber high-level architecture*

2.1 Operating Modes

NetSim Cyber can be used in different operating modes depending on the test objective, available hardware, and required level of realism. The same attack workflow can be applied in simple simulator-based studies as well as in distributed laboratory testbeds with real devices.

The two primary operating modes are:

- Single-system deployment
- Distributed deployment

In Host-system deployment, the source simulator, NetSim Cyber, Threat Agent, and destination simulator run on a single computer. This mode is useful for early-stage validation, training, protocol testing, and attack logic development.

In distributed deployment, the source and destination run on separate systems or devices, while NetSim Cyber operates as an intermediate system in the communication path. This mode is useful for realistic cyber-physical testing, hardware-in-the-loop integration, and experiments involving real PMUs, PDCs, IEDs, RTUs, PLCs, SCADA systems, or HIL platforms.

NetSim Cyber can therefore support a progressive testing workflow. Users can first develop and validate the attack in a host-system setup and then repeat the experiment in a distributed setup with real or hardware-based systems.

Table 2-1: *NetSim Cyber operating modes*

Operating Mode	Description	Typical Use Case	Systems Required
Single-system deployment	Source, destination, and NetSim Cyber run on one computer	Early-stage testing, protocol validation, training, attack logic development	One PC
	NetSim Cyber uses built-in open-source protocol simulators	Quick protocol testing without external devices	
Distributed deployment	Source and destination run on separate systems, with NetSim Cyber placed in the communication path	Real-device testing, HIL integration, laboratory cyber-physical experiments	Two or more systems

Operating Mode	Description	Typical Use Case	Systems Required
	NetSim Cyber interfaces with real devices or external simulators	Realistic testing with PMUs, PDCs, IEDs, RTUs, PLCs, SCADA, Typhoon HIL, OPAL-RT, or RTDS	Multiple systems/devices

2.2 Single-System Deployment

Single-system deployment is the simplest way to use NetSim Cyber. In this mode, all major experiment components run on a single computer. The source simulator generates traffic, the destination simulator receives traffic, and NetSim Cyber applies the selected attack through the Threat Agent.

This mode is useful when the user wants to validate protocol behavior, test an attack configuration, generate datasets, or demonstrate cyber-attack effects without setting up multiple machines or physical devices.

A host-system setup may include:

- A built-in PMU simulator sending IEEE C37.118 synchrophasor data.
- A built-in or external PDC subscriber receiving synchrophasor data.
- A Modbus master and slave simulator.
- A DNP3 master and outstation simulator.
- IEC 61850 GOOSE publisher and subscriber tools.
- A NetSim Cyber Threat Agent configured between the source and destination traffic flow.

The main advantage of same-system deployment is reduced setup complexity. Users can quickly create and test attack scenarios without configuring external routing, multiple network interfaces, or physical communication links. This is especially useful during training, demonstration, and initial attack development.

However, single-system deployment may not capture all timing, routing, device behavior, and network-interface characteristics of a real laboratory testbed. Therefore, once the attack logic is validated, users may repeat the experiment in distributed deployment for higher realism.

Copyright TETCOS LLP

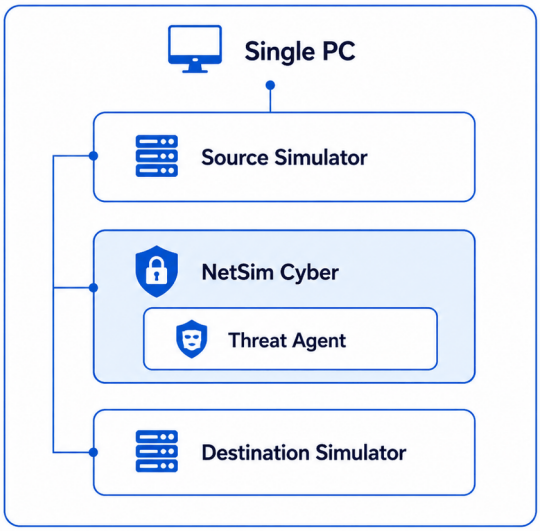


Figure 2-2: *Single-system deployment*

2.3 Multiple-System Deployment

Multiple-System deployment is used when the source and destination are located on separate systems or physical devices. In this mode, NetSim Cyber is placed between the communicating endpoints and acts as an intermediate network system. Traffic from the source is routed through NetSim Cyber before reaching the destination.

This mode is suitable for realistic cyber-physical experiments involving actual power-system devices, hardware-in-the-loop simulators, or operational technology testbeds. For example, a PMU simulator running on one system may send IEEE C37.118 data to an openPDC instance running on another system, while NetSim Cyber runs on a third system and modifies the traffic in transit.

A multiple-system setup may include:

- System 1: Source device or simulator: Example: PMU, IED, RTU, PLC, Typhoon HIL, RTDS, OPAL-RT, or protocol simulator.
- System 2: NetSim Cyber system: Runs the NetSim Cyber GUI, network scenario, and Threat Agent.
- System 3: Destination device or application: Example: PDC, SCADA master, HMI, GOOSE subscriber, DNP3 master, or IEC-104 master station.

In Multiple-system deployment, correct network configuration is essential. The source and destination traffic must pass through the NetSim Cyber system. This can be achieved through static routing, automatic configuration using NetSimCyberClient.exe, or a two-NIC gateway setup where the NetSim Cyber PC routes traffic between two separate networks.

Multiple-System deployment provides a more realistic test environment because it includes real network interfaces, actual device communication, system-level timing, and external application behaviour. It is recommended for validation studies, HIL testing, device interoperability testing, and cyber-resilience evaluation.

Copyright TETCOS LLP

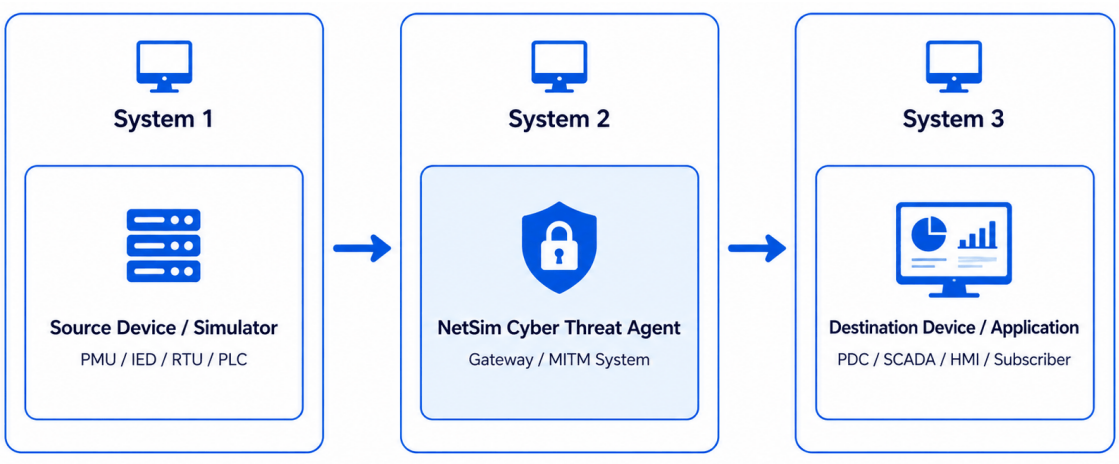


Figure 2-3: *Multiple-System Deployment*

Table 2-2: *Example distributed deployment components*

System	Role	Example Component	Function
System 1	Source	PMU simulator, IED, RTU, PLC, Typhoon HIL, RTDS	Generates power-system protocol traffic
System 2	NetSim Cyber	NetSim Cyber PC with Threat Agent	Intercepts, modifies, delays, replays, or forwards traffic
System 3	Destination	PDC, SCADA master, HMI, subscriber, control application	Receives and processes protocol traffic

2.4 Threat Agent Concept

The **Threat Agent** is the logical attack node in NetSim Cyber. It represents an adversarial or test-controlled entity placed in the communication path between a source and a destination. The Threat Agent intercepts selected packets, applies attack logic, and forwards the resulting traffic.

The Threat Agent does not simply drop or corrupt packets randomly. It can perform protocol-aware manipulation based on the selected protocol and attack type. For example, in a Synchrophasor experiment, the Threat Agent may modify voltage magnitude, current magnitude, frequency, ROCOF, or timestamp fields while keeping the frame valid. In a Modbus TCP experiment, it may alter register values. In a GOOSE experiment, it may replay or manipulate event-related fields.

The Threat Agent is configured using the NetSim Cyber GUI. The user selects the source node, destination node, protocol port, attack protocol, and attack type. Once the simulation starts, the Threat Agent monitors matching traffic streams and applies the selected modification logic.

The Threat Agent can be used to emulate attacks such as:

- False data injection
- Replay attacks

- Delay attacks
- Packet drops
- Random noise injection
- Ramp attacks
- Increment and decrement attacks
- Pulse or surge attacks
- Frequency and ROCOF manipulation
- Timestamp manipulation
- Protocol-specific payload modification
- User-defined custom payload modification

The Threat Agent is useful for studying how cyber-attacks affect monitoring, control, protection, and situational-awareness applications in power-system networks.

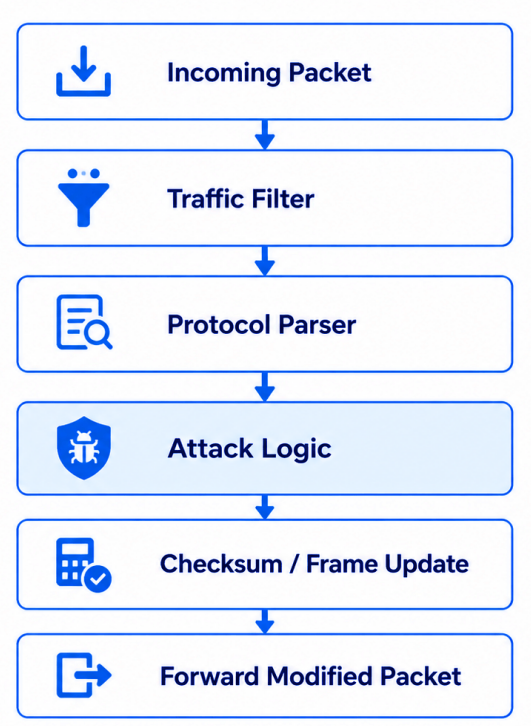


Figure 2-4: *Threat Agent packet-processing workflow*

Table 2-3: *Threat Agent functions*

Function	Description
Traffic filtering	Selects packets based on source, destination, protocol, and port
Packet interception	Captures matching packets flowing between endpoints
Protocol parsing	Identifies protocol fields that can be modified

Function	Description
Payload modification	Applies selected attack logic to measurement or control data
Timing manipulation	Introduces delay, replay, or time-related modifications
Frame forwarding	Sends the modified or unmodified packet to the destination
Custom attack support	Allows user-defined payload modification through scripts

2.5 Protocol-Aware Attack Emulation

Protocol-aware attack emulation is one of the main capabilities of NetSim Cyber. Instead of treating packets as generic byte streams, NetSim Cyber understands the structure of supported power-system protocols and allows attacks to be applied to meaningful protocol fields.

This is important because many attacks on power-system communication networks do not need to break communication. A more realistic attack may keep the protocol session active while changing selected values inside valid frames. From the receiver's point of view, the communication may appear normal, but the measurement or control information may be incorrect.

For example:

- In IEEE C37.118 traffic, the attack may modify phasor magnitude, phase angle, frequency, ROCOF, or timestamp information.
- In Modbus TCP traffic, the attack may modify coil status, register values, or response data.
- In DNP3 traffic, the attack may alter object values, status indications, or measurement data.
- In IEC 61850 GOOSE traffic, the attack may replay messages or manipulate status/event-related fields.
- In IEC 60870-5-104 traffic, the attack may modify telemetry or telecontrol information.

Protocol-aware emulation allows NetSim Cyber to create realistic attack scenarios while preserving the communication format expected by the receiving application. This helps users evaluate whether monitoring systems, control logic, protection applications, or detection algorithms can identify abnormal data even when the communication session remains active.

This capability is useful for:

- Studying false data injection attacks.
- Evaluating anomaly detection and intrusion detection methods.
- Testing device and application response to abnormal protocol values.
- Generating labelled datasets for AI/ML-based cybersecurity research.
- Understanding the cyber-physical impact of communication attacks.
- Validating protocol implementation robustness.

Copyright TETCOS LLP

Protocol	Example Target Field	Example Attack	Possible Power-System Impact
IEEE C37.118 Synchrophasor	Voltage, current, frequency, ROCOF, timestamp	False data injection, ramp, noise, time-shift	Incorrect wide-area monitoring, false event interpretation, state-estimation errors
Modbus TCP	Holding registers, coils, input registers	Register value modification, replay	Incorrect SCADA/HMI display or control logic response
DNP3	Analog inputs, binary inputs, counters	Measurement manipulation, replay	Incorrect outstation data at SCADA master
IEC 61850 GOOSE / R-GOOSE	Status, event, sequence-related fields	Replay, spoofing, event manipulation	False protection or automation event indication
IEC 60870-5-104	Telemetry and telecontrol information	Data manipulation, replay, delay	Incorrect control-center visibility or command interpretation
Custom TCP/IP protocol	User-defined payload fields	Custom payload modification	Protocol-specific cyber-physical effect

3 Supported Power-System Protocols

NetSim Cyber supports major communication protocols used in power-system monitoring, control, protection, and automation networks. These protocols are commonly used between devices such as PMUs, PDCs, IEDs, RTUs, PLCs, SCADA masters, HMIs, substation gateways, and control-centre applications.

The supported protocols allow users to create realistic cyber-physical test scenarios and evaluate how communication attacks affect power-system visibility, measurement integrity, command reliability, and automation behaviour.

NetSim Cyber supports the following power-system protocols:

- IEEE C37.118 Synchrophasor
- IEC 61850 GOOSE, Routed GOOSE, MMS, and Sampled Values
- Modbus TCP
- DNP3
- IEC 60870-5-104
- Custom TCP/IP-based protocols

These protocols can be used with built-in simulators, external software tools, hardware-in-the-loop platforms, or real power-system devices depending on the test environment.

Copyright TETCOS LLP

Table 3-1: *Supported power-system protocols*

Protocol	Typical Source Device	Typical Destination Device	Common Use Case
IEEE C37.118 Synchrophasor	PMU	PDC	Wide-area monitoring and Synchrophasor data streaming
IEC 61850 GOOSE / R-GOOSE	IED / Relay	IED / Subscriber	Substation protection and event messaging
IEC 61850 MMS	IED / Server	SCADA Client / Engineering Tool	Substation data access and device communication
IEC 61850 Sampled Values	Merging Unit	Protection Relay / Subscriber	High-speed sampled measurement transmission
Modbus TCP	PLC / RTU / Slave	SCADA Master / HMI	Industrial monitoring and control
DNP3	RTU / Outstation	SCADA Master	SCADA telemetry and control
IEC 60870-5-104	IEC-104 Slave / Controlled Station	IEC-104 Master / Control Center	Telecontrol and control-center communication
Custom TCP/IP	User-defined source	User-defined destination	Testing proprietary or custom application protocols

3.1 IEEE C37.118 Synchrophasor

IEEE C37.118 is a communication standard used for transmitting Synchrophasor measurements from Phasor Measurement Units, commonly known as PMUs, to Phasor Data Concentrators, commonly known as PDCs. Synchrophasor data is widely used in wide-area monitoring systems, dynamic grid visualization, oscillation monitoring, event analysis, and power-system situational awareness.

A PMU measures electrical quantities such as voltage phasors, current phasors, frequency, and rate of change of frequency. These measurements are time-synchronized, usually using GPS or another precision time source, and are streamed to a PDC for aggregation, visualization, storage, or further analysis.

In NetSim Cyber, IEEE C37.118 traffic can be used to study attacks on Synchrophasor measurement integrity and timing. Since Synchrophasor applications depend heavily on accurate values and precise timestamps, even small modifications can affect monitoring and analysis systems.

Typical IEEE C37.118 data includes:

- Voltage magnitude and angle
- Current magnitude and angle
- Frequency

Copyright TETCOS LLP

- Rate of Change of Frequency, or ROCOF
- Analog values
- Digital status words
- Timestamp information
- Configuration frames
- Command frames

NetSim Cyber can emulate attacks that modify selected fields in the Synchrophasor data stream while allowing communication between the PMU and PDC to continue. This enables users to evaluate whether the receiving PDC, monitoring tool, or detection algorithm can identify abnormal data.

Typical attack scenarios include:

- Voltage or current magnitude manipulation
- Phase angle manipulation
- Frequency manipulation
- ROCOF manipulation
- Timestamp shifting
- Random noise injection
- Increment and decrement attacks
- Ramp attacks
- Pulse or surge attacks
- Replay of valid frames



Table 3-2: *IEEE C37.118 communication flow*

Item	Description
Typical source	PMU or PMU simulator
Typical destination	PDC or openPDC
Common traffic	Synchrophasor measurement frames
Typical data fields	Phasors, frequency, ROCOF, analog values, digital values, timestamps
Example attacks	False data injection, ramp, noise, timestamp shift, frequency manipulation
Typical application	Wide-area monitoring cybersecurity, PMU/PDC testing, dataset generation

3.2 IEC 61850 GOOSE, R-GOOSE, MMS, and SV

IEC 61850 is a communication standard widely used in substation automation systems. It defines communication models and services for intelligent electronic devices, protection relays, merging units, bay controllers, substation gateways, and automation systems.

NetSim Cyber supports IEC 61850-related traffic used in protection, monitoring, and control environments. The major IEC 61850 communication types relevant to NetSim Cyber are:

- GOOSE
- Routed GOOSE, or R-GOOSE
- MMS
- Sampled Values, or SV

GOOSE stands for Generic Object-Oriented Substation Event. It is used for fast event-based messaging between IEDs inside substations. GOOSE messages are commonly used for protection trips, interlocking, blocking, breaker status transfer, and automation logic.

GOOSE communication is time-critical. A replayed, delayed, dropped, or modified GOOSE message may affect protection or automation behavior in a test environment. NetSim Cyber can be used to study how GOOSE subscribers respond to abnormal or manipulated event messages.

Typical GOOSE attack scenarios include:

- GOOSE replay
- Message delay
- Packet drop
- Status value manipulation
- Sequence-related manipulation
- Event spoofing

Copyright TETCOS LLP

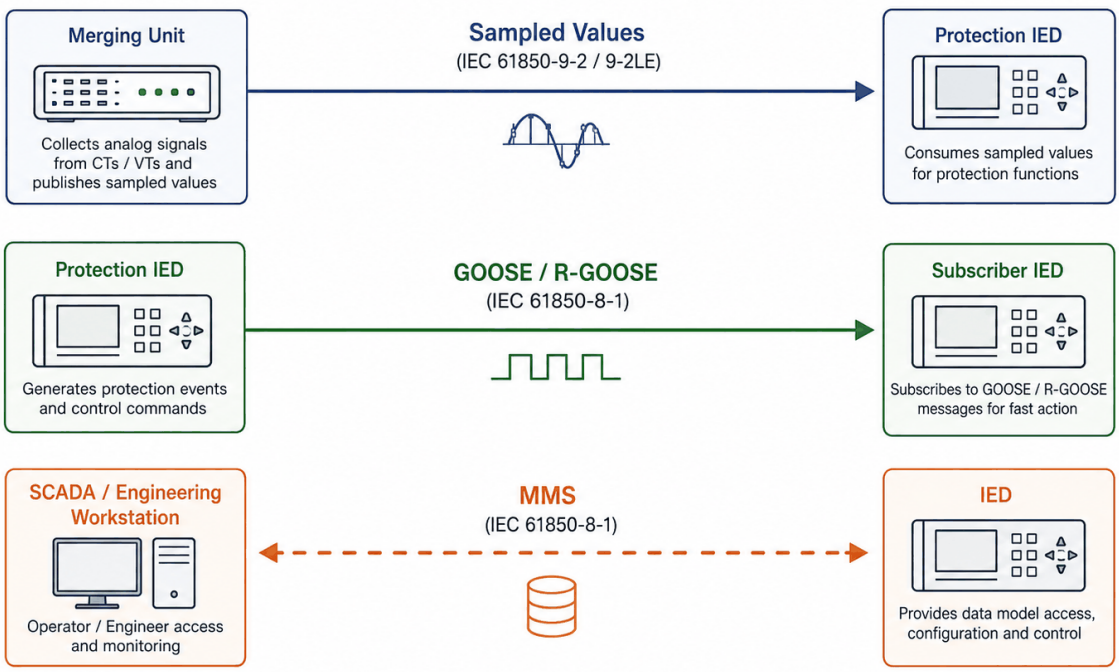


Figure 3-1: *IEC 61850 communication types in a substation*

Table 3-3: *IEC 61850 protocol support*

IEC 61850 Type	Typical Source	Typical Destination	Purpose	Example Attack Scenario
GOOSE	IED / Protection Relay	IED / Subscriber	Fast event and protection messaging	Replay, delay, status manipulation
R-GOOSE	IED / Gateway	Remote subscriber	Routed event communication	Replay, delay, packet manipulation
MMS	SCADA client / Engineering tool	IED / Server	Monitoring, reporting, control, configuration	Data manipulation, unauthorized value modification in test environment
Sampled Values	Merging Unit	Protection Relay / Subscriber	Sampled voltage/current measurement transfer	Measurement manipulation, delay, packet loss

3.3 Modbus TCP

Modbus TCP is an industrial communication protocol commonly used for monitoring and control between SCADA masters, HMIs, PLCs, RTUs, and field devices. It uses a client-server communication model where the client, usually a master or HMI, sends read or write requests to a server device, usually a PLC, RTU, or slave device.

Modbus TCP is widely used because it is simple, open, and easy to implement. However, many Modbus deployments do not include built-in authentication or encryption. This makes it useful

for studying how unauthorized or manipulated messages can affect industrial monitoring and control systems in a controlled testbed.

Typical Modbus data includes:

- Coils
- Discrete inputs
- Holding registers
- Input registers

In NetSim Cyber, Modbus TCP traffic can be intercepted and modified between the master and slave. The Threat Agent can alter register values, modify response data, replay valid messages, or simulate abnormal data conditions.

Typical attack scenarios include:

- Holding register value modification
- Input register value modification
- Coil status manipulation
- Replay of previous responses
- Delay of read or write responses
- Packet drop
- False process value injection

For power-system applications, Modbus TCP may be used to exchange measurements, breaker status, control commands, equipment status, or auxiliary system data. Manipulating these values in a lab setup can help users study incorrect SCADA display values, false alarms, or abnormal control behavior.



Figure 3-2: *Modbus TCP communication flow*

Table 3-4: *Modbus TCP support in NetSim Cyber*

Item	Description
Typical source	SCADA master, HMI, or Modbus client
Typical destination	PLC, RTU, or Modbus slave
Common port	TCP 502
Common data	Coils, discrete inputs, holding registers, input registers
Example attacks	Register modification, coil manipulation, replay, delay, packet drop
Typical application	Industrial control testing, SCADA cybersecurity, false process value analysis

3.4 DNP3

DNP3, or Distributed Network Protocol version 3, is commonly used in electric utility SCADA systems. It is used for communication between control centers, SCADA masters, RTUs, IEDs, and outstations. DNP3 is designed for reliable communication in utility environments and supports time-stamped measurements, events, binary states, analog values, counters, and control operations.

A typical DNP3 system consists of a master station and one or more outstations. The master requests data or sends control commands, while the outstation provides field data and event information.

DNP3 is used for:

- Substation telemetry
- Feeder automation
- Remote terminal unit communication
- Distribution automation
- Control-center communication
- Event reporting

In NetSim Cyber, DNP3 traffic can be used to study the effect of cyber-attacks on SCADA data integrity and communication behavior. The Threat Agent can be configured to manipulate selected values, replay messages, delay traffic, or drop packets in a controlled test environment.

Typical DNP3 attack scenarios include:

- Analog input value manipulation
- Binary input status manipulation
- Counter value modification
- Replay of valid outstation responses
- Delay of telemetry messages

Copyright TETCOS LLP

- Packet drop
- False event injection in a test environment

For power-system engineers, DNP3 attack studies are useful for understanding how manipulated field data may affect SCADA displays, alarms, event logs, operator awareness, and automation decisions.



Figure 3-3: *DNP3 Communication Flow*

Table 3-5: *DNP3 support in NetSim Cyber*

Item	Description
Typical source	DNP3 master or SCADA system
Typical destination	RTU, IED, or DNP3 outstation
Common port	TCP/UDP 20000
Common data	Analog inputs, binary inputs, counters, events, controls
Example attacks	Measurement manipulation, replay, delay, packet drop, status modification
Typical application	Utility SCADA cybersecurity, RTU/outstation testing, telemetry integrity analysis

3.5 IEC 60870-5-104

IEC 60870-5-104, commonly called IEC-104, is a telecontrol protocol used for communication between control centers and substations or remote stations. It is widely used in electric power transmission and distribution systems, especially for supervisory control and data acquisition over TCP/IP networks.

IEC-104 is based on the IEC 60870-5-101 application layer adapted for TCP/IP communication. It supports telemetry, telecontrol, event reporting, and status monitoring between a master station and a controlled station.

A typical IEC-104 system includes:

- Control center or master station
- Remote terminal unit or controlled station
- Substation gateway

- Telemetry and telecontrol data exchange

IEC-104 communication may include:

- Single-point information
- Double-point information
- Measured values
- Integrated totals
- Commands
- Time-tagged events

In NetSim Cyber, IEC-104 traffic can be used to study attacks on telecontrol and telemetry communication. The Threat Agent can emulate traffic manipulation, delay, replay, or packet loss in a controlled environment.

Typical IEC-104 attack scenarios include:

- Telemetry value modification
- Status indication manipulation
- Replay of previously valid messages
- Delay of telecontrol or telemetry messages
- Packet drop
- Time-tag manipulation

For power-system engineers, IEC-104 attack simulation is useful for evaluating control-center visibility, remote station communication behavior, and the impact of incorrect telemetry or delayed messages.



Figure 3-4: *IEC 60870-5-104 communication flow*

Table 3-6: *IEC 60870-5-104 support in NetSim Cyber*

Item	Description
Typical source	IEC-104 master or control center
Typical destination	IEC-104 slave, RTU, gateway, or controlled station
Common port	TCP 2404
Common data	Telemetry, status indications, measured values, commands, time-tagged events
Example attacks	Telemetry manipulation, replay, delay, packet drop, timestamp modification
Typical application	Transmission/distribution SCADA testing, telecontrol cybersecurity, control-center visibility analysis

3.6 Custom TCP/IP Protocol Support

In addition to built-in support for standard power-system protocols, NetSim Cyber can also support custom TCP/IP-based application protocols. This capability is useful when users want to test proprietary protocol traffic, research prototypes, laboratory tools, or application-specific communication formats.

Custom protocol support allows users to define payload modification logic using user scripts. The user can inspect packet payload bytes, identify fields of interest, apply custom transformations, and return the modified payload for forwarding through the NetSim Cyber Threat Agent.

This is useful for cases where:

- The protocol is proprietary or not included in the built-in protocol list.
- The user wants to test a research protocol or experimental communication format.
- The payload structure is known to the user.
- The user wants to implement a custom false data injection, replay, delay, or payload modification experiment.
- The user wants to extend NetSim Cyber for a specific laboratory or product validation use case.

Custom TCP/IP protocol support is typically implemented through the payload_modifier_custom.py file in the Workspace PythonUserCode folder. Users can edit this file to define how payload data should be modified during the simulation.

A custom protocol workflow usually includes:

1. Identify the source and destination applications.
2. Configure the application traffic filter in NetSim Cyber.
3. Select or attach the custom payload modifier.
4. Define the parsing and modification logic in the custom script.

5. Run baseline traffic without attack.
6. Enable the custom attack.
7. Verify the effect at the receiving application.

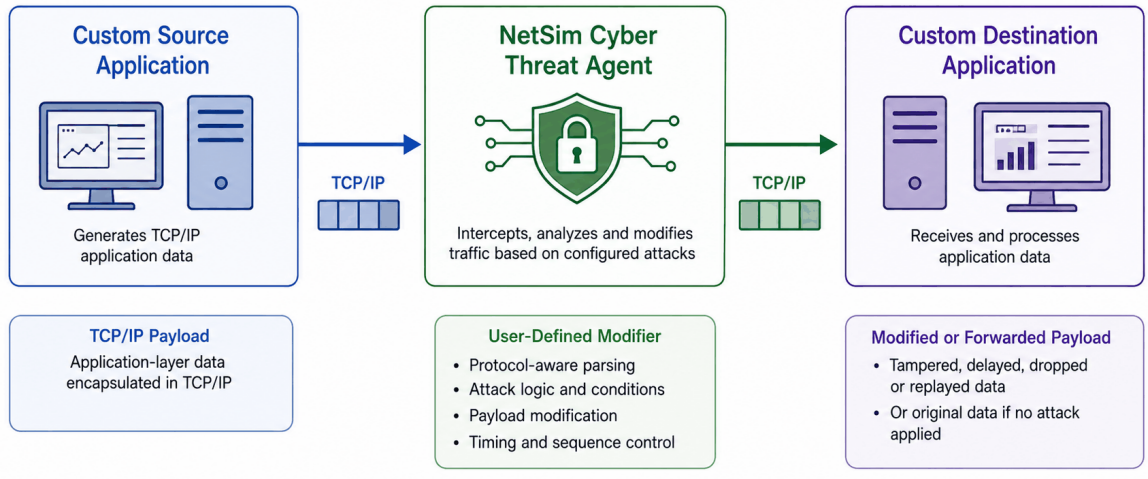


Table 3-7: *Custom TCP/IP protocol attack workflow*

Item	Description
Supported traffic	TCP/IP-based application traffic
Modifier file	payload_modifier_custom.py
User responsibility	Define payload parsing and modification logic
Typical use case	Proprietary protocol testing, research protocols, laboratory experiments
Example attacks	Custom field modification, replay, delay, packet drop, payload transformation
Requirement	User must understand the payload structure of the target protocol

3.7 Summary

Table 3-8: *Protocol comparison summary*

Protocol	Communication Model	Typical Power-System Area	Main Cybersecurity Concern
IEEE C37.118	PMU to PDC streaming	Wide-area monitoring	False synchrophasor data, timestamp manipulation

Protocol	Communication Model	Typical Power-System Area	Main Cybersecurity Concern
IEC 61850 GOOSE / R-GOOSE	Publisher-subscriber	Substation automation and protection	Replay, spoofing, delayed event messages
IEC 61850 MMS	Client-server	Substation monitoring and control	Unauthorized or manipulated data access
IEC 61850 SV	Publisher-subscriber	Digital substations	Measurement manipulation or packet loss
Modbus TCP	Client-server	Industrial control and SCADA	Register manipulation and false process values
DNP3	Master-outstation	Utility SCADA	Telemetry and event manipulation
IEC 60870-5-104	Master-controlled station	Control-center telecontrol	Telemetry, command, and time-tag manipulation
Custom TCP/IP	User-defined	Laboratory or proprietary systems	Payload-specific manipulation

4 Simulator and Device Integration

NetSim Cyber can be integrated with both external power-system simulators and built-in protocol simulators. This flexibility allows users to begin with simple software-based experiments and then extend the same workflow to hardware-in-the-loop platforms, real-time simulators, and physical power-system devices.

In a cyber-physical power-system testbed, the physical system behaviour and the communication network behaviour are closely linked. A power-system simulator may generate voltage, current, frequency, breaker status, or control data, while NetSim Cyber models the communication path and applies cyber-attack logic to the data exchanged between devices.

NetSim Cyber supports two main integration approaches:

- **External simulator and real-device integration**: NetSim Cyber interfaces with external power-system simulators, HIL platforms, SCADA systems, and real devices.
- **Built-in open-source protocol simulator integration**: NetSim Cyber includes built-in protocol simulators that can generate and receive traffic for supported power-system protocols without requiring external devices.

Together, these capabilities allow users to perform both early-stage attack validation and realistic laboratory testing.

Copyright TETCOS LLP

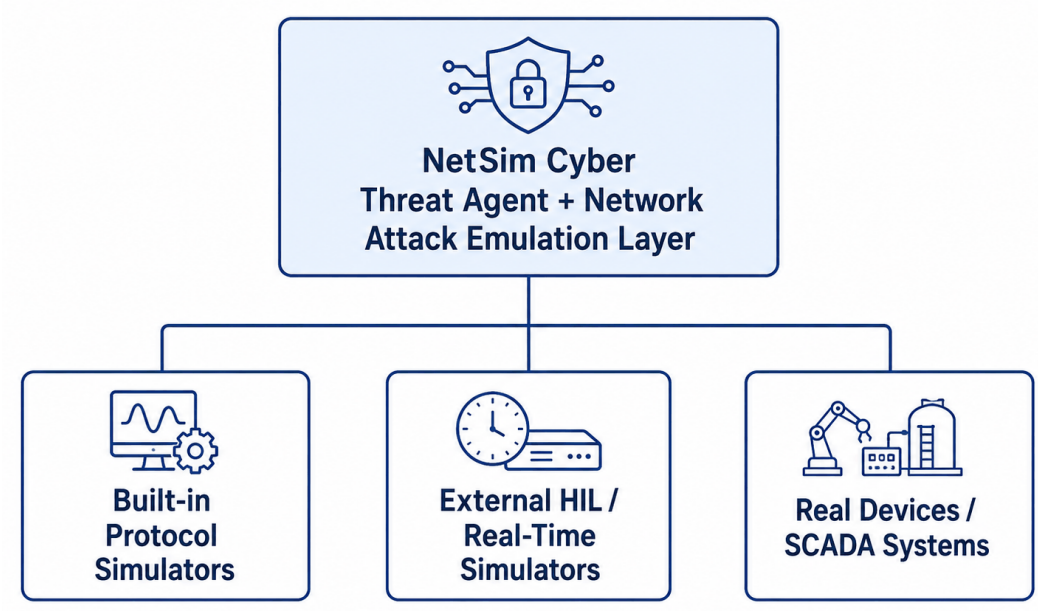


Figure 4-1: *NetSim Cyber simulator and device integration options*

4.1 External Power-System Simulator Integration

NetSim Cyber can interface with external power-system simulators and real-time simulation platforms to support cyber-physical power-system studies. In this type of setup, the external simulator models the electrical system, while NetSim Cyber models the communication network and applies cyber-attack behavior to the protocol traffic.

External simulator integration is useful when users want to evaluate the impact of cyber-attacks on realistic power-system dynamics, device behavior, control actions, protection logic, or SCADA monitoring.

NetSim Cyber can be integrated with platforms and systems such as:

- Typhoon HIL
- OPAL-RT
- RTDS
- MATLAB
- Simulink
- SCADA systems
- PMUs
- PDCs
- IEDs
- RTUs
- PLCs
- Protection relays

Copyright TETCOS LLP

- Substation gateways
- Control-center applications

In an external simulator setup, the simulator or physical device acts as the source or destination of protocol traffic. NetSim Cyber is placed between the communicating endpoints and intercepts the protocol messages exchanged between them. The Threat Agent can then modify, delay, replay, drop, or forward traffic based on the configured attack scenario.

For example:

- A Typhoon HIL system may generate Modbus TCP or DNP3 traffic representing simulated field measurements.
- An RTDS system with a communication module may transmit PMU data to a PDC.
- A MATLAB/Simulink model may interact with external communication tools or controllers.
- A protection relay or IED may exchange GOOSE or MMS traffic with another substation device.
- A SCADA master may communicate with an RTU or PLC using Modbus TCP, DNP3, or IEC 60870-5-104.

Table 4-1: *External simulator and device integration examples*

External System / Device	Typical Role	Example Protocols	Example Use Case
Typhoon HIL	Real-time power-system simulator	Modbus TCP, DNP3, IEC 61850, custom TCP/IP	HIL-based attack impact studies
OPAL-RT	Real-time power-system simulator	C37.118, Modbus TCP, DNP3, IEC 61850	Real-time grid simulation and communication testing
RTDS	Real-time digital simulator	C37.118, IEC 61850, DNP3	PMU/PDC and protection communication studies
MATLAB / Simulink	Power-system model or control model	TCP/IP, custom protocol, external interface	Algorithm validation and co-simulation
SCADA System	Master / HMI / control application	Modbus TCP, DNP3, IEC-104	SCADA cybersecurity testing
PMU	Measurement source	IEEE C37.118	Synchrophasor attack studies
PDC	Measurement concentrator	IEEE C37.118	PMU data validation and monitoring
IED / Relay	Protection and automation device	IEC 61850 GOOSE, MMS, SV	Substation automation testing

External System / Device	Typical Role	Example Protocols	Example Use Case
RTU / PLC	Field device or controller	Modbus TCP, DNP3, IEC-104	Industrial and utility communication testing

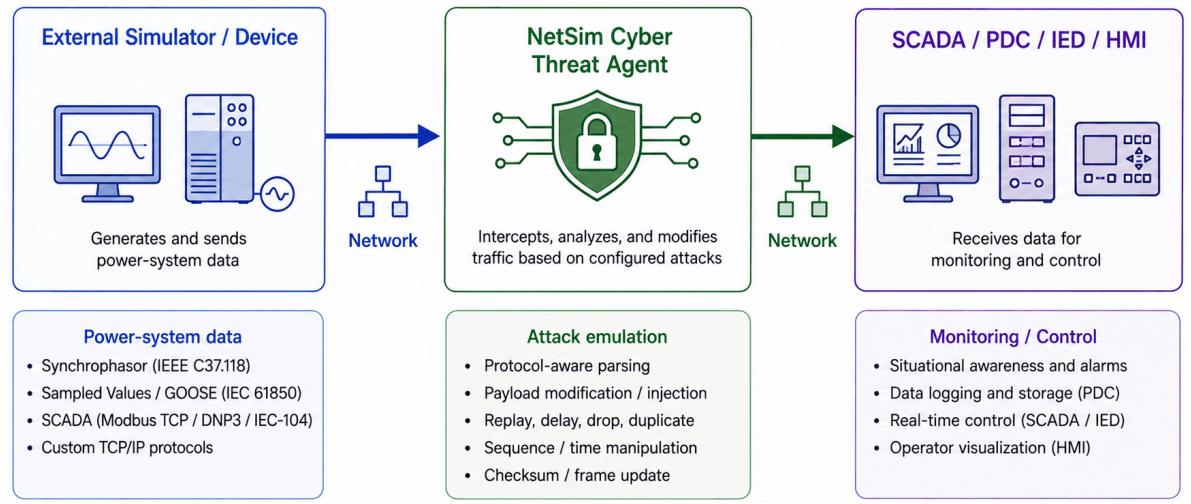


Figure 4-2: *External simulator integration workflow*

4.2 Built-in Open-Source Protocol Simulators

NetSim Cyber includes built-in open-source protocol simulators for major power-system communication protocols. These simulators allow users to create complete protocol test environments without requiring physical PMUs, PDCs, IEDs, RTUs, PLCs, SCADA systems, or hardware-in-the-loop platforms during early-stage testing.

The built-in simulators are useful for protocol validation, attack emulation, training, demonstration, dataset generation, and rapid testing of detection algorithms. Users can first validate attack logic using built-in simulators and then move the experiment to an external simulator or real-device testbed.

NetSim Cyber supports built-in simulation for protocols such as:

- IEEE C37.118 Synchrophasor
- IEC 61850 GOOSE, Routed GOOSE, MMS, and Sampled Values
- Modbus TCP
- DNP3
- IEC 60870-5-104

These simulators provide both source-side and destination-side components. For example, a PMU simulator can generate synchrophasor data, while a PDC subscriber can receive and analyze it. Similarly, a Modbus slave can generate register data while a Modbus master reads or writes values.

Table 4-2: *Built-in Open-Source Protocol Simulators*

Protocol	Source / Publisher Simulator	Destination / Subscriber Simulator	Open-Source Framework Used
IEEE C37.118 Synchrophasor	PMU Simulator	OpenPDC / PDC Subscriber	pyPMU, OpenPDC
IEC 61850 GOOSE, R-GOOSE, MMS, SV	IED / Relay Publisher	GOOSE Subscriber / Analyzer	libiec61850
MODBUS TCP	PLC / RTU Simulator	SCADA HMI / Modbus Master	pymodbus, QModMaster
DNP3	RTU Simulator	SCADA Master	OpenDNP3, DNP3 Station
IEC 60870-5-104	IEC-104 Slave Simulator	IEC-104 Master Station	lib60870

4.3 Benefits of Built-in Simulators

The built-in simulator capability provides a ready-to-use environment for protocol testing, attack emulation, and cyber-security research.

Table 4-3: *Benefits of built-in simulators*

Benefit	Description
Rapid attack prototyping	Quickly test attack logic before moving to larger testbeds
Protocol-level validation	Verify protocol parsing and field-level manipulation
Single-system deployment	Run source, destination, network, and attack logic on one PC
Reduced complexity	Avoid dependence on physical devices during early-stage testing
Safe experimentation	Test attacks without affecting real equipment
Dataset generation	Generate labeled normal and attack traffic for AI/ML research
Faster iteration	Modify scenarios and attack settings quickly
Real-data support	Import or replay real measurement data for realistic testing

4.4 Benefits of External Simulator Integration

Benefit	Description
Realistic cyber-physical testing	Study attack impact on simulated electrical behavior and communication networks

Benefit	Description
HIL validation	Test cyber-attacks with real-time simulators and hardware interfaces
Real-device testing	Evaluate response of PMUs, PDCs, IEDs, RTUs, PLCs, relays, and SCADA systems
Timing realism	Include real network interfaces, device delays, and physical communication paths
End-to-end validation	Observe the full chain from measurement generation to monitoring or control response
OT testbed support	Integrate NetSim Cyber into laboratory SCADA and substation automation environments
Progressive testing	Move from built-in simulators to real-device experiments after initial validation

NetSim Cyber supports both lightweight simulator-based experimentation and realistic external testbed integration. Built-in simulators help users quickly create repeatable cyber-attack scenarios, while external simulator and device integration allows the same concepts to be validated in more realistic cyber-physical environments. This makes NetSim Cyber suitable for training, research, device validation, protocol testing, and power-system cybersecurity studies.

5 Software Requirements and Installation

This section describes the software dependencies required to install and run NetSim Cyber. These dependencies are required for launching the NetSim Cyber application, running protocol simulators, executing Python-based payload modifiers, and supporting real-time packet interception and attack emulation.

Before using NetSim Cyber, ensure that the required runtime components, Python packages, workspace folders, and administrator permissions are correctly configured. Incorrect dependency versions or missing permissions may prevent the application, protocol simulators, or Threat Agent from working correctly.

NetSim Cyber requires the following software components:

- NetSim Cyber application
- Python runtime
- Required Python modules
- .NET runtime
- Administrator privileges
- NetSim Cyber workspace and PythonUserCode folder
- Protocol-specific simulator dependencies, if applicable

Copyright TETCOS LLP

Table 5-1: *Software requirement summary*

Requirement	Purpose	Required For
NetSim Cyber	Main application for scenario creation, simulation, and attack emulation	All users
Python Runtime	Executes Python-based payload modifiers and simulator utilities	Payload modification, custom attacks, simulator tools
Python Modules	Supports GUI tools, plotting, and simulator interfaces	PMU/PDC UI, Modbus GUI, custom tools
.NET Runtime	Supports NetSim Cyber application components	NetSim Cyber GUI and core services
Administrator Privileges	Allows packet capture, routing changes, and traffic injection	Real-time traffic handling and attack emulation
Workspace Folder	Stores scenario files and user scripts	Project-specific simulation and attack logic
PythonUserCode Folder	Stores Layer 3, Layer 4, and Layer 5 packet modification scripts	Custom and protocol-aware attacks

5.1 Dependency Summary

NetSim Cyber depends on specific runtime components and supporting libraries. These dependencies must be installed before running protocol-based experiments.

The exact version requirements may vary depending on the NetSim Cyber release. Users should always follow the dependency versions provided with the installed NetSim Cyber version.

Table 5-2: *NetSim Cyber dependency summary*

Dependency	Required Version	Installation Scope	Used By	Symptom if Missing or Incorrect
Python Runtime	Python 3.14.x, 64-bit	Install for all users	Python plugin bridge, payload modifiers, simulator utilities	Failed to load UserCodePyPlugin.dll
PySide6	Latest compatible version	System-wide Python installation	PMU/PDC UI and custom simulator GUIs	GUI fails to launch, ImportError
pyqtgraph	Latest compatible version	System-wide Python installation	Plotting and simulator visualization tools	Graphs not rendered, ImportError

Dependency	Required Version	Installation Scope	Used By	Symptom if Missing or Incorrect
.NET Runtime	Version required by NetSim Cyber installer	System-wide	NetSim Cyber application	Application fails to start
Administrator Privileges	Windows administrator access	Run-time permission	Packet capture, route updates, traffic injection	Traffic not intercepted or route changes fail

Important: The Python runtime version must match the version expected by NetSim Cyber. Installing a different Python version, or installing Python only for the current user, may prevent the Python plugin bridge from loading correctly.

5.2 Python Runtime

NetSim Cyber uses Python for payload modification, custom attack scripts, protocol simulator utilities, and selected user-interface components. The Python runtime must be installed correctly before using these features.

5.2.1 Required Python Version

NetSim Cyber requires the Python version specified for the installed release. For NetSim Cyber v15.1, the required version is:

Python 3.14.x, 64-bit

The Python version must match the version used to build the NetSim Cyber Python plugin bridge. If the installed Python version does not match the expected version, NetSim Cyber may fail to load the Python plugin.

Typical error: *Failed to load UserCodePyPlugin.dll*

5.2.2 Installation Scope

Python must be installed using the **Install for all users** option.

This is important because some NetSim Cyber components may run under system-level processes or require access outside the current user profile. A current-user-only Python installation may not be visible to those components.

Mandatory installation path: C:\Program Files\Python314\

Avoid installing on local user: C:\Users\<UserName>\AppData\Local\Programs\Python\

5.2.3 Fresh Python Installation

Use the following procedure to install Python for NetSim Cyber.

1. Download the required 64-bit Python installer.

2. Run the installer as Administrator.
3. On the first installer screen, enable:
    - **Install launcher for all users**
    - **Add Python to PATH**
4. Select **Customize installation**.
5. Continue through the optional features screen.
6. On the advanced options screen, enable:
    - **Install for all users**

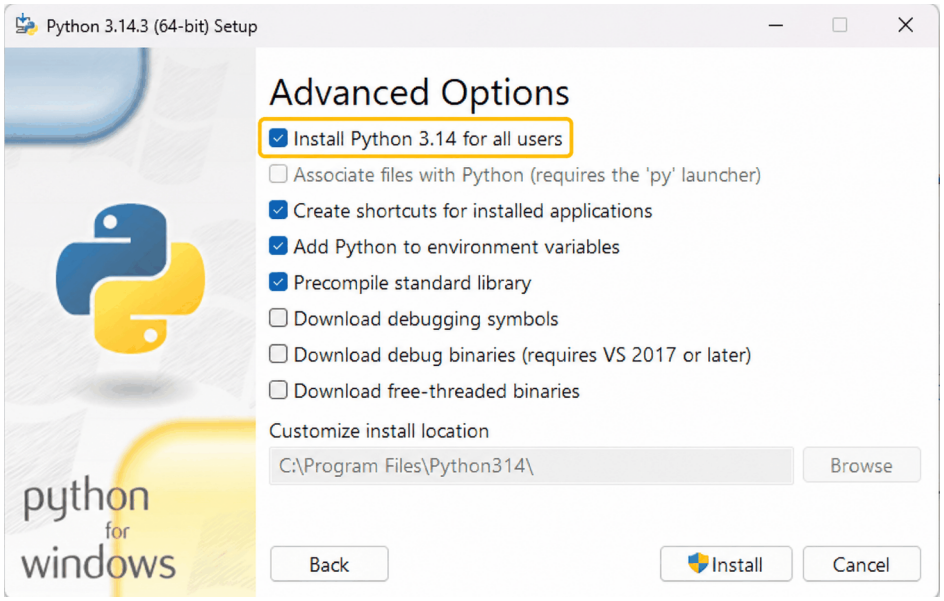


Figure 5-1: *Python Installer: Choose Install Python 3.14 for all users*

7. Confirm that the installation path is: C:\Program Files\Python314\
8. Complete the installation.
9. Open Command Prompt as Administrator.
10. Run:
    - *python –version*
    - Expected output: Python 3.14.x

5.2.4 Existing Python Installation

If Python is already installed on the system, check whether it matches the version required by NetSim Cyber.

Open Command Prompt as Administrator and run: ***python --version***

Then run: ***where python***

The output should point to the system-wide Python installation: ***C:\Program Files\Python314\python.exe***

If the output points to a user profile path or a different Python version, install the required Python version system-wide.

5.2.5 Multiple Python Versions

Some systems may already contain multiple Python installations, such as Python installed by Anaconda, Visual Studio, or another software package. In such cases, ensure that the system PATH resolves to the Python version required by NetSim Cyber.

To check this, run: ***where python***

The first path in the list should correspond to the required Python installation.

Note: If multiple Python versions are installed, NetSim Cyber may load the wrong version if the system PATH is not configured correctly. Ensure that the required Python version appears first in the system PATH.

5.3 Python Modules

After installing the required Python runtime, install the Python modules used by NetSim Cyber simulator tools and user interfaces.

Open Command Prompt as Administrator before running the installation commands. This ensures that the packages are installed into the system-wide Python environment.

5.3.1 Installing PySide6

The PySide6 package is required by selected NetSim Cyber simulator GUIs, such as PMU/PDC user interfaces and custom simulator windows.

Install command: ***pip install PySide6***

Verification command: ***pip show PySide6***

Error if missing: ***ImportError: No module named PySide6***

Possible effect:

- PMU simulator UI may not open.
- PDC simulator UI may not open.
- Custom simulator GUI may fail to launch.

5.3.2 Installing pyqtgraph

The pyqtgraph package is required for graphing, plotting, and visual display of simulator outputs.

Install command: ***pip install pyqtgraph***

Verification command: ***pip show pyqtgraph***

Error if missing: ***ImportError: No module named pyqtgraph***

Copyright TETCOS LLP

Possible effect:

- Graphs may not render.
- Simulator measurement plots may not open.
- Modbus or other protocol visualization tools may fail.

5.3.3 Installing Modules Using the Correct Python

If multiple Python versions are installed, use the full Python path to ensure packages are installed into the correct environment.

Example:

"C:\Program Files\Python314\python.exe" -m pip install PySide6

"C:\Program Files\Python314\python.exe" -m pip install pyqtgraph

Table 5-3: *Python module requirements*

Python Module	Used By	Install Command	Verification Command
PySide6	PMU/PDC UI, custom simulator GUI	pip install PySide6	pip show PySide6
pyqtgraph	Plotting and visualization tools	pip install pyqtgraph	pip show pyqtgraph

5.4 .NET Runtime

NetSim Cyber requires the .NET Runtime version specified for the installed release. The .NET Runtime supports application-level components used by the NetSim Cyber GUI and core services.

For NetSim Cyber v15.1, use the .NET Runtime version provided or required by the NetSim Cyber installer.

In most installations, this dependency is handled automatically by the NetSim Cyber installer. Users normally do not need to install .NET separately unless the application fails to start or the installer specifically requests it.

5.4.1 Verifying .NET Runtime Installation

Open Command Prompt and run: dotnet --list-runtimes

The installed .NET runtime versions will be displayed.

If the required runtime is missing, install the required version and restart the system before launching NetSim Cyber again.

5.4.2 Symptoms of Missing .NET Runtime

If the required .NET Runtime is missing or corrupted, the following symptoms may occur:

- NetSim Cyber application does not start.
- Application window closes immediately after launch.
- Windows displays a missing runtime message.
- NetSim Cyber service or GUI components fail to initialize.

Note: The required .NET Runtime should be verified against the NetSim Cyber release notes or installer package. Do not replace it with a different runtime version unless specified by TETCOS support or the installation guide.

5.5 Administrator Privileges

NetSim Cyber must be run with administrator privileges for packet capture, route updates, traffic forwarding, and packet injection to work correctly.

Administrator access is required for operations such as:

- Capturing live packets from network interfaces.
- Injecting or forwarding modified packets.
- Updating route tables.
- Running NetSimCyberClient.exe.
- Enabling or modifying network forwarding behavior.
- Accessing system-wide Python and runtime components.
- Opening restricted protocol ports.

5.5.1 Running NetSim Cyber as Administrator

To run NetSim Cyber as Administrator:

1. Right-click the NetSim Cyber shortcut.
2. Select **Run as administrator**.
3. Accept the User Account Control prompt.

5.5.2 Configuring NetSim Cyber to Always Run as Administrator

To avoid repeating this step every time:

1. Right-click the NetSim Cyber shortcut.
2. Select **Properties**.
3. Open the **Compatibility** tab.
4. Enable **Run this program as an administrator**.

5. Click **Apply**.
6. Click **OK**.

If the option should apply to all users:

1. Click **Change settings for all users**.
2. Enable **Run this program as an administrator**.
3. Click **Apply**.
4. Click **OK**.

5.5.3 Running NetSim Cyber Client as Administrator

NetSimCyberClient.exe should also be run as Administrator on source and destination systems when automatic route configuration is used.

To launch it:

1. Right-click NetSimCyberClient.exe.
2. Select **Run as administrator**.
3. Confirm that it connects to the NetSim Cyber system.

Important: If NetSim Cyber or NetSimCyberClient.exe is not run as Administrator, traffic may not be routed through the Threat Agent, and attacks may not be applied even if the simulation appears to run.

5.6 Verification and Troubleshooting

After installing the required software components, verify the environment before running any attack scenario.

5.6.1 Installation Verification Checklist

Use the following checklist before starting NetSim Cyber experiments.

Table 5-4: *Installation verification checklist*

Check	Command / Action	Expected Result
Verify Python version	python --version	Required Python version is displayed
Verify Python path	where python	Path points to system-wide Python installation
Verify PySide6	pip show PySide6	Package details are displayed
Verify pyqtgraph	pip show pyqtgraph	Package details are displayed
Verify .NET Runtime	dotnet --list-runtimes	Required runtime is listed

Check	Command / Action	Expected Result
Verify administrator mode	Launch NetSim Cyber as Administrator	No permission-related startup errors
Verify workspace creation	Open or create a workspace	Workspace folder is created successfully
Verify PythonUserCode folder	Check workspace contents	L3, L4, and L5 modifier folders are present

5.6.2 Workspace and PythonUserCode Folder Verification

NetSim Cyber uses a workspace folder to store scenario-specific files, topology information, and user-defined packet modification scripts. A typical workspace contains a PythonUserCode folder. This folder includes subfolders for different layers of packet modification.

Table 5-5: *PythonUserCode folder structure*

Folder	Entry-Point Function	Layer	Purpose
L3-ipv4-packet-modifier	modify_ipv4_header()	Layer 3	Modify IPv4 header fields
L4-tcp-packet-modifier	modify_tcp_header()	Layer 4	Modify TCP header fields
L5-payload-modifier	modify_payload()	Layer 5	Modify application-layer payload

The L5-payload-modifier folder contains protocol-specific and custom payload modifier files.

Table 5-6: *Layer 5 payload modifier files*

File	Description
payload_modifier_modbus.pyd	Pre-built Modbus TCP attack logic
payload_modifier_dnp3.pyd	Pre-built DNP3 attack logic
payload_modifier_goose_sv.pyd	Pre-built IEC 61850 GOOSE / Sampled Values attack logic
payload_modifier_synchrophasor_2011.pyd	Pre-built IEEE C37.118-2011 Synchrophasor attack logic
payload_modifier_synchrophasor_2005.pyd	Pre-built IEEE C37.118-2005 Synchrophasor attack logic
payload_modifier_custom.py	Editable template for custom protocol attack logic

Important: The .pyd files are compiled binaries and should not be edited. To create a custom attack, edit payload_modifier_custom.py in the workspace folder.

5.6.3 Administrator Permission Issues

If NetSim Cyber is not run with administrator privileges, the following issues may occur:

- Route table updates fail.
- Traffic does not pass through NetSim Cyber.
- Packets are not captured.
- Packets are not injected.
- NetSimCyberClient.exe does not configure routes.
- The Threat Agent does not receive matching traffic.

Solution:

1. Close NetSim Cyber.
2. Relaunch NetSim Cyber as Administrator.
3. Relaunch NetSimCyberClient.exe as Administrator on source and destination systems.
4. Recheck routing and traffic flow.

5.6.4 Troubleshooting Summary

Table 5-7: *Common installation and runtime issues*

Problem	Likely Cause	Recommended Action
Failed to load UserCodePyPlugin.dll	Wrong Python version or user-only Python installation	Install required 64-bit Python for all users
Simulator GUI does not open	Missing PySide6	Run pip install PySide6
Graphs do not display	Missing pyqtgraph	Run pip install pyqtgraph
NetSim Cyber does not start	Missing or incorrect .NET Runtime	Verify using dotnet --list-runtimes
Packets are not intercepted	NetSim Cyber not run as Administrator or wrong route configuration	Relaunch as Administrator and verify routes
Attack does not affect destination	Traffic filter mismatch or wrong protocol selected	Check source, destination, port, and attack protocol
Client does not connect to gateway	Firewall or discovery port blocked	Allow required ports and run client as Administrator
Python command opens wrong version	Multiple Python versions in PATH	Update system PATH or use full Python path

Copyright TETCOS LLP

5.7 Workspace and PythonUserCode Folder

5.7.1 Workspace

On first launch, NetSim Cyber prompts the user to create a default Workspace. A Workspace is a folder under:

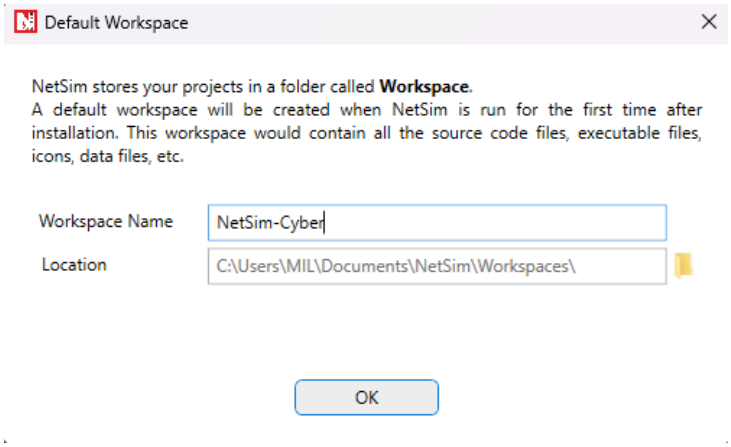


Figure 5-2: *Default Workspace*

It holds the network topology, simulation configuration, and the PythonUserCode scripts for that scenario. Additional workspaces can be created at any time from the Your Work tab in the NetSim Cyber interface by clicking the book icon to open the list of workspaces.

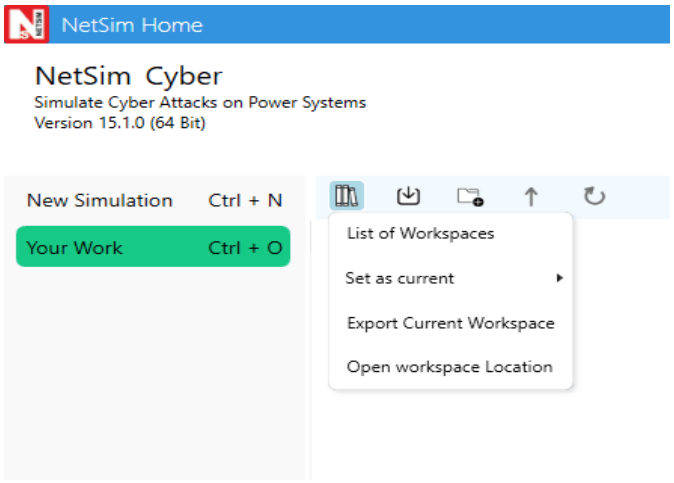


Figure 5-3: *Manage Workspaces*

From the list of workspaces, click the book icon with the plus (+) sign, enter a workspace name, and click Create to create a new workspace.

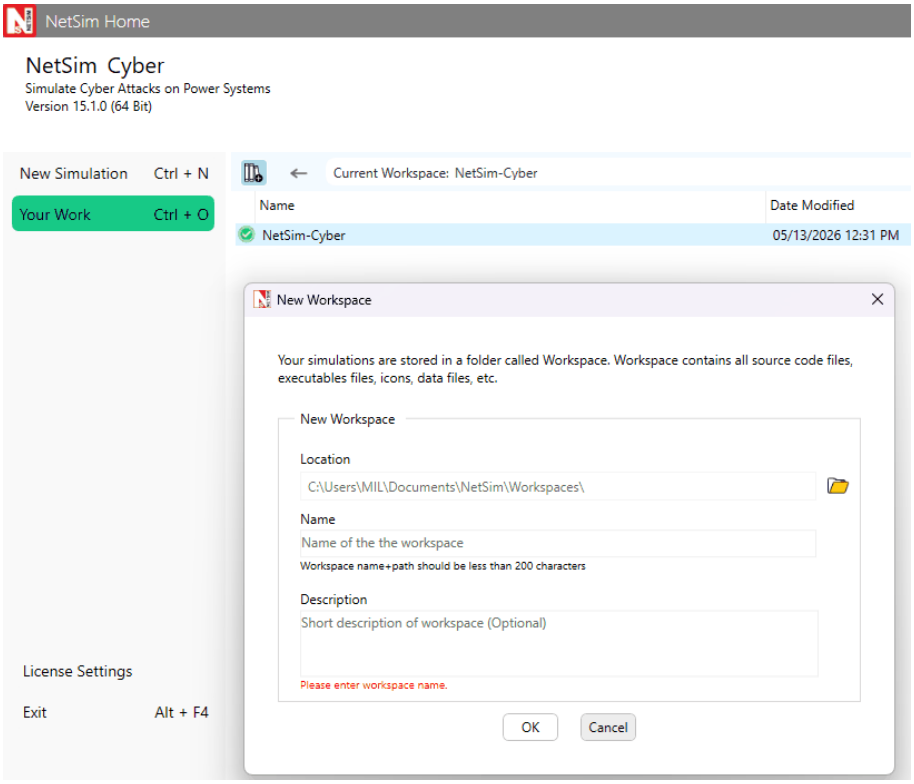


Figure 5-4: *Create New Workspace*

5.7.2 PythonUserCode Folder

Each Workspace contains a PythonUserCode folder with three sub-folders, one for each layer of the network stack where packet-level attack logic can be applied:

Folder	Entry-point Function	Layer
L3-ipv4-packet-modifier\	modify_ipv4_header()	IPv4 header fields
L4-tcp-packet-modifier\	modify_tcp_header()	TCP header fields
L5-payload-modifier\	modify_payload()	Application-layer payload bytes

The scripts in these folders are copied from the NetSim Cyber installation directory when the Workspace is created, so each Workspace starts with a clean set of templates. NetSim loads the scripts from the Workspace at simulation start and calls the entry-point function once for every packet passing through the attacker node.

5.7.3 L5-payload-modifier in Detail

The L5-payload-modifier folder includes pre-built compiled modules for common industrial protocols and an editable template for custom protocols:

File	Description
payload_modifier_modbus.pyd	Pre-built Modbus attack logic.

File	Description
payload_modifier_dnp3.pyd	Pre-built DNP3 attack logic.
payload_modifier_goose_sv.pyd	Pre-built IEC 61850 GOOSE / Sampled Values attack logic.
payload_modifier_synchrophasor_2011.pyd	Pre-built IEEE C37.118-2011 Synchrophasor attack logic.
payload_modifier_synchrophasor_2005.pyd	Pre-built IEEE C37.118-2005 Synchrophasor attack logic.
payload_modifier_custom.py	Blank template for any custom protocol attack.

Note: To write a custom attack, edit payload_modifier_custom.py inside the Workspace's PythonUserCode\L5-payload-modifier\ folder. The .pyd files are compiled binaries and should not be modified.

5.8 .NET Runtime

5.8.1 Required Version

NetSim Cyber v15.1 requires .NET Runtime 10.0.3.

Note: This dependency is automatically handled during NetSim installation. Users do not need to install .NET separately.

6 Lab Setup and Network Configuration

This section describes how to set up NetSim Cyber for same-system and distributed cyber-attack simulation experiments. The purpose of the lab setup is to ensure that protocol traffic flows through the NetSim Cyber Threat Agent before reaching the destination application or device.

NetSim Cyber can be used in different network configurations depending on the experiment. For early-stage testing, all components may run on a single system. For realistic testing, the source, NetSim Cyber system, and destination may run on separate systems or devices.

The main lab setup options are:

- Single-system setup
- Three-system setup
- Manual route configuration
- Automatic configuration using NetSim Cyber Client
- Two-NIC routing setup

Table 6-1: *NetSim Cyber lab setup options*

Setup Type	Description	Recommended Use
Same-system setup	Source simulator, NetSim Cyber, and destination simulator run on one PC	Training, quick testing, protocol validation
Three-system setup	Source, NetSim Cyber, and destination run on separate systems	Real-device testing, distributed lab setup
Manual route configuration	User manually configures routes through the NetSim Cyber system	Controlled routing and fixed IP environments
Automatic configuration	NetSimCyberClient.exe automatically detects NetSim Cyber and updates routes	Faster distributed setup
Two-NIC routing setup	NetSim Cyber PC routes traffic between two separate networks	HIL, RTDS, Typhoon HIL, and isolated subnet testing

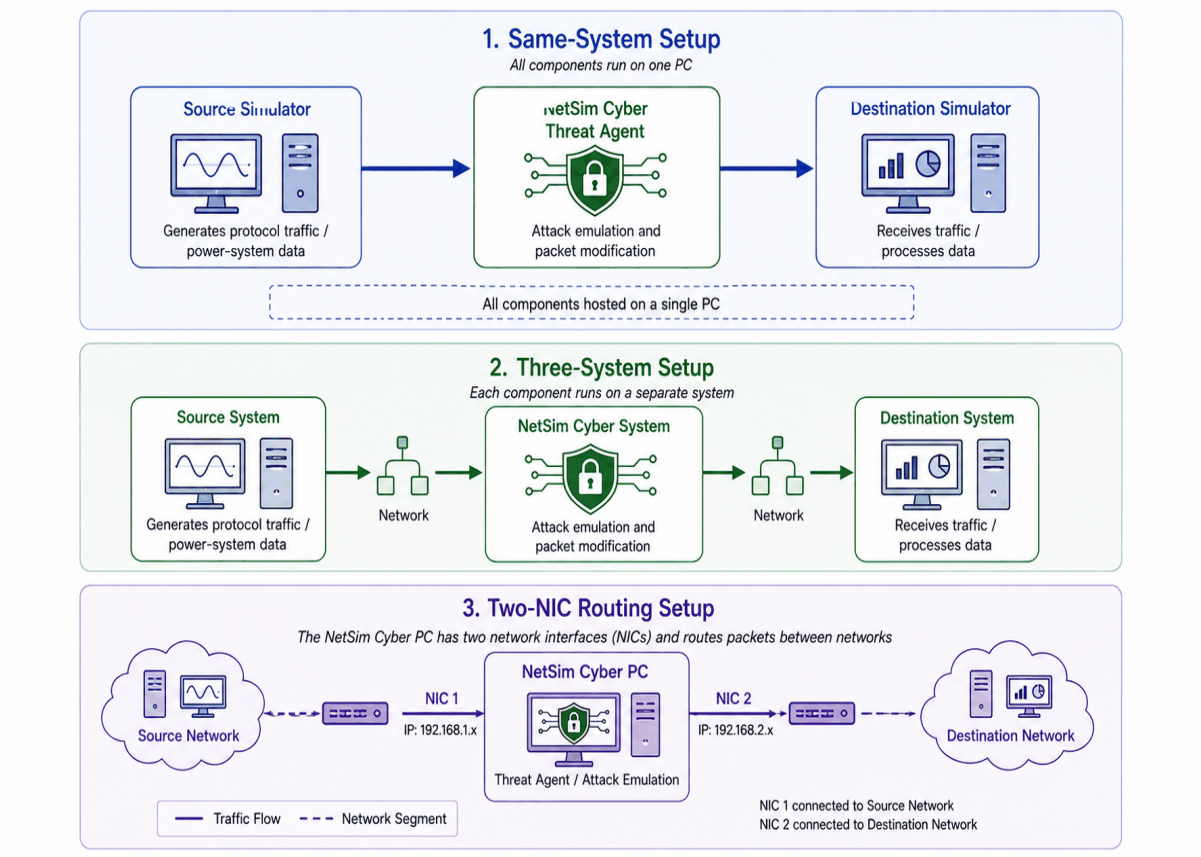


Figure 6-1: *System Configuration Setup*

6.1 Detailed Setup Configuration

6.1.1 Method 1: Manual Route Configuration

1. Source system (192.168.0.83)

To add a static route, you'll need to open the command prompt in administrator mode. Once

open, input the following command:

route add <Destination IP address> mask <Subnet mask> <Gateway IP address> metric 1

Command Example:

route add 192.168.0.82 mask 255.255.255.255 192.168.0.81 metric 1

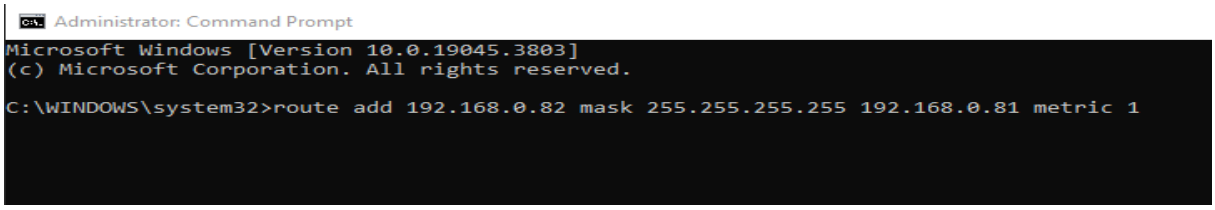


Figure 6-2: *Example for configuring static route in source system*

Replace *<Destination IP address> with* the IP address of the destination system, *<Subnet mask>* with the appropriate subnet mask, and *<Gateway IP address>* with the IP address of the gateway. This command will effectively add the static route.

2. Destination system (192.168.0.82)

To facilitate communication for TCP traffic, it's essential to configure a reverse route on the destination system as well.

route add <IP address of Destination system> mask <Subnet mask> <Gateway IP address> metric 1

Command Example:

route add 192.168.0.83 mask 255.255.255.255 192.168.0.81 metric 1

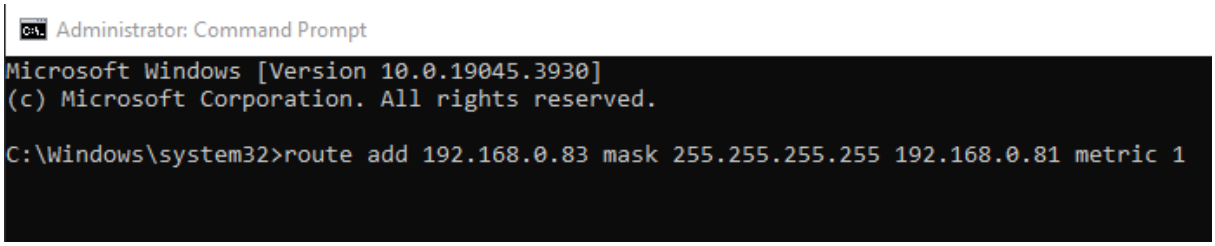


Figure 6-3: *Example for configuring static route in Destination system*

6.1.2 Method 2: Automatic Configuration using NetSim Cyber Client

As an alternative to manually configuring routes on the source and destination systems, users can utilize the **NetSimCyberClient.exe** utility to automate the process.

Prerequisites: Before proceeding, ensure the NetSim Cyber Suite UI is running on the gateway system (192.168.0.81)

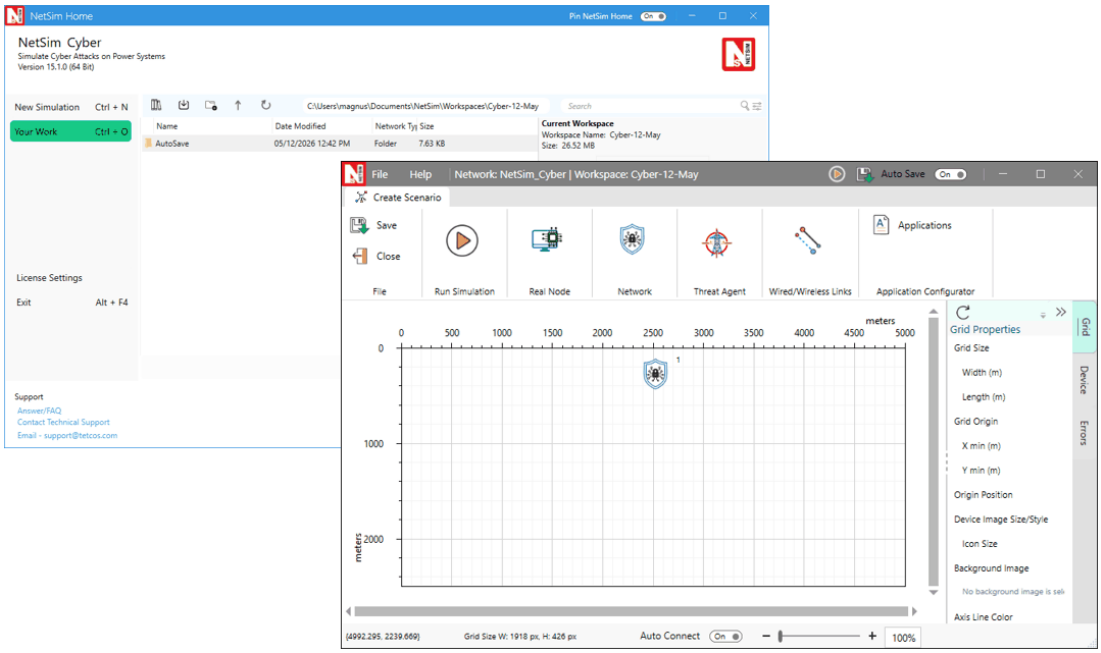


Figure 6-4: *Launching the NetSim Cyber on the gateway system (IP: 192.168.0.81)*

Steps:

1. Place the NetSimCyberClient.exe utility on both the Source/Client system (192.168.0.83) and the Destination/Server system (192.168.0.82).
2. Run the **NetSimCyberClient.exe** as run as administrator on both systems.

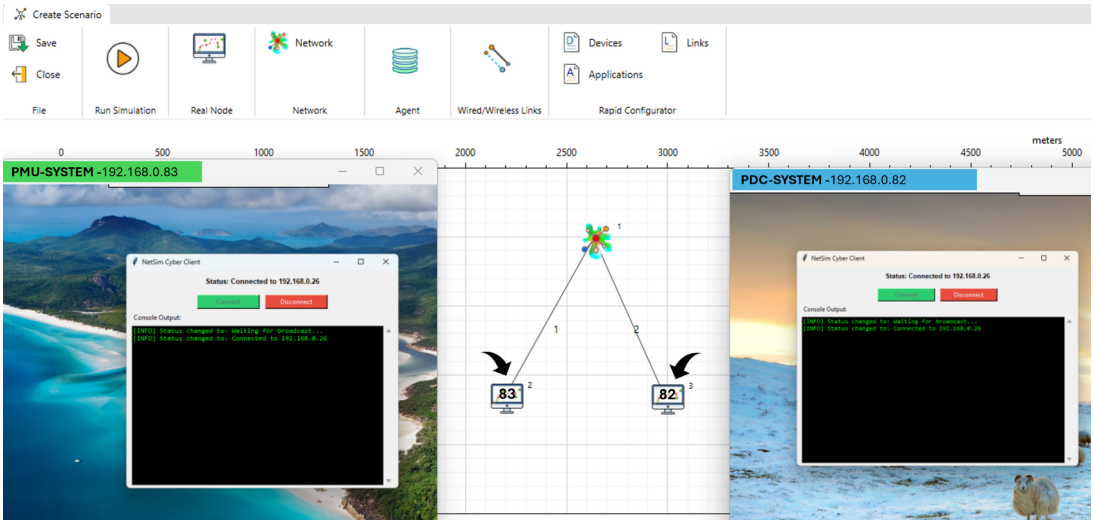


Figure 6-5: *NetSimCyberClient.exe utility interface showing the **Connect** option and traffic tools, launched with administrator privileges on both source/destination systems*

Upon running the utility, the nodes will automatically be dropped into the NetSim environment, as seen in the above example layout where distinct nodes connect to the central network cloud. The routing tables on the client and server systems will be updated automatically to direct traffic through the NetSim gateway.

How it works:

When the NetSim Cyber tile starts on the gateway system, it begins broadcasting a message containing its IP address across the entire network on port 19001 every second.

- NetSimCyberClient utility runs on the client or server system, it starts listening for these broadcast messages on port 19001.
- Upon receiving a message from the network on this port, the utility establishes a connection with the NetSim system.
- It then automatically alters the local route table so that packets are transmitted through the system running NetSim.
- Finally, it starts sending packets containing its own IP address and Device Name to NetSim. When NetSim receives these packets on port 19001, it reads the information and automatically drops a device node with the respective IP address and Device Name into the simulation scenario.

Troubleshooting Automatic Configuration

Table 6-2: *NetSim Cyber Client troubleshooting*

Problem	Possible Cause	Recommended Action
Client does not detect NetSim Cyber	Discovery traffic blocked	Check firewall and network reachability
Client does not update routes	Client not run as Administrator	Relaunch NetSimCyberClient.exe as Administrator
Device does not appear in NetSim Cyber	Client not connected or wrong network	Check IP address and network adapter
Traffic bypasses NetSim Cyber	Route update failed	Run route print and verify gateway
Connection works but attack not applied	Application filter mismatch	Check source, destination, protocol, and port

6.1.3 Method-3: Three-System Network Configuration for NetSim Cyber

This document describes how to configure a three-system network setup for NetSim Cyber. The same procedure can be used with:

- Typhoon HIL
- RTDS systems (GTNET Card, PMU, PDC)
- RTAC devices
- Other Ethernet-based simulation or control systems

The setup uses a Windows PC running NetSim Cyber as a router between two separate networks.

Overview

The setup contains three systems:

1. **Source-side device or simulator**
   - Example: Typhoon HIL
2. **Routing system**
   - Windows PC with NetSim Cyber installed
   - Contains two Ethernet interfaces
   - Routes traffic between both networks
3. **Destination-side device or simulator**
   - Example: RTAC, PMU, or PDC

Example Network Topology

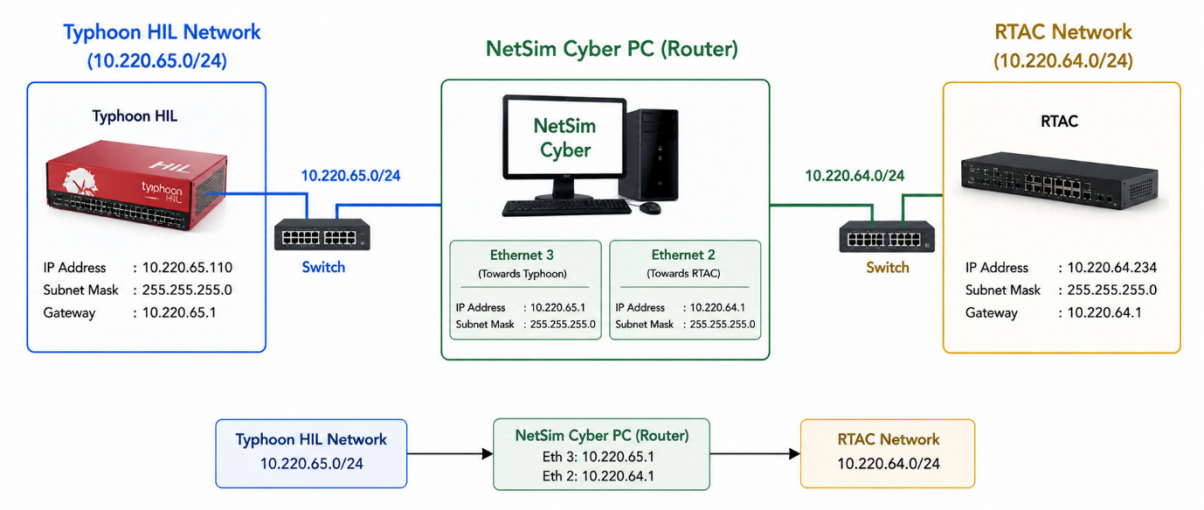


Figure 6-6: *Three-System Network Configuration*

Example IP Configuration

Source-Side Network:

Table 6-3: *Source-Side Network Configuration*

Parameter	Value
Device	Typhoon HIL
IP Address	10.220.65.110
Subnet Mask	255.255.255.0
Default Gateway	10.220.65.1

NetSim Cyber PC

The NetSim Cyber system must contain two Ethernet interfaces.

Ethernet Interface Connected to Source Network:

Table 6-4: *Ethernet Interface Configuration for Source Network*

Parameter	Value
Interface	Ethernet 3
IP Address	10.220.65.1
Subnet Mask	255.255.255.0

Ethernet Interface Connected to Destination Network:

Table 6-5: *Ethernet Interface Configuration for Destination Network*

Parameter	Value
Interface	Ethernet 2
IP Address	10.220.64.1
Subnet Mask	255.255.255.0

Destination-Side Network:

Table 6-6: *Destination-Side Network Configuration*

Parameter	Value
Device	RTAC
IP Address	10.220.64.234
Subnet Mask	255.255.255.0
Default Gateway	10.220.64.1

Requirements

Before starting the configuration, ensure the following requirements are met.

Hardware Requirements:

- Three systems or devices connected through Ethernet
- Two Ethernet ports available on the NetSim Cyber PC
- Ethernet cables and network connectivity

Software Requirements:

- Windows operating system on the NetSim Cyber PC
- NetSim Cyber installed
- Administrator access on the NetSim Cyber PC

Copyright TETCOS LLP

Configure IP Address and Gateway

Perform the following steps on each system.

Step 1: Open Network Adapter Settings

1. Open: Control Panel
2. Navigate to: Network and Internet > Network and Sharing Centre
3. Click: Change adapter settings
4. Right-click the required Ethernet adapter.
5. Select: Properties

Step 2: Open IPv4 Settings

1. Select: Internet Protocol Version 4 (TCP/IPv4)
2. Click: Properties
3. Select: Use the following IP address

Step 3: Enter Network Settings

Enter the IP address, subnet mask, and default gateway based on the configuration tables shown earlier.

After entering the values:

1. Click: OK
2. Close all remaining windows.

Repeat the same procedure for all required systems and interfaces.

Enable IP Forwarding on the NetSim Cyber PC

The NetSim Cyber PC acts as a router between the two networks. IP forwarding must therefore be enabled.

Step 1: Open PowerShell as Administrator

1. Open the Windows Start menu.
2. Search for: PowerShell
3. Right-click Windows PowerShell.
4. Select: Run as administrator

Step 2: Enable Packet Forwarding

- Execute the following command:
- Set-NetIPInterface -Forwarding Enabled
- **What This Command Does?**
  - This command enables packet forwarding on the network interfaces.
  - Without this setting, Windows will not route packets between the two Ethernet interfaces.

Step 3: Enable IP Routing in Windows

- Execute the following command:
- Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters" -Name "IPEnableRouter" -Value 1
- **What This Command Does?**

- This command enables Windows IP routing through the registry.
- It updates the following registry setting:
- IPEnableRouter = 1
- This allows the NetSim Cyber PC to function as a network router.

Step 4: Restart the System

- Restart the NetSim Cyber PC after executing the PowerShell commands.
- This ensures that the routing configuration is applied correctly.

Verification

After completing the configuration, verify that communication works correctly between both networks.

Verify IP Configuration

Open Command Prompt or PowerShell and execute:

ipconfig

Verify that:

- All interfaces show the correct IP addresses
- Subnet masks are correct
- Gateways are configured properly

Verify Network Connectivity from the Source-Side System

Execute: ***ping 10.220.64.234***

Expected Result

Successful replies confirm that the source-side system can reach the destination-side system through the NetSim Cyber PC.

From the Destination-Side System

Execute: ***ping 10.220.65.110***

Expected Result

Successful replies confirm bidirectional communication between both networks.

Verify Interface Forwarding

On the NetSim Cyber PC, execute: ***Get-NetIPInterface***

Verify that forwarding is enabled on the required Ethernet interfaces.

Verify NetSim Cyber Operation

Start traffic generation or communication between the connected systems and verify that:

- Traffic passes between both networks
- NetSim Cyber captures or processes the traffic correctly
- Communication remains stable during operation

6.1.4 Connectivity Troubleshooting

Table 6-7: *Connectivity troubleshooting*

Problem	Likely Cause	Recommended Action
Source cannot ping destination	Incorrect IP, subnet, gateway, or cable connection	Verify ipconfig, cable, switch, and adapter settings
Destination cannot ping source	Reverse route missing	Add reverse route through NetSim Cyber
Ping works but protocol does not connect	Firewall or wrong port	Allow protocol port and verify application configuration
Traffic bypasses NetSim Cyber	Route not configured correctly	Run route print and update static routes
NetSim Cyber Client does not connect	Discovery blocked or client not run as Administrator	Check firewall and relaunch client as Administrator
Attack does not apply	Wrong traffic filter or protocol selection	Verify source, destination, protocol, and port
Two-NIC setup does not route traffic	IP forwarding not enabled	Enable forwarding and restart NetSim Cyber PC

Copyright TETCOS LLP

Problem	Likely Cause	Recommended Action
Destination receives no data after attack starts	Payload corrupted or wrong attack selected	Stop attack, verify protocol selection, and retest baseline

After completing the lab setup and connectivity verification, the NetSim Cyber environment is ready for attack configuration. Users should always verify baseline communication before enabling any attack. This ensures that any observed change at the destination is due to the configured Threat Agent behavior and not due to routing, firewall, or protocol setup errors.

7 How to Use NetSim Cyber GUI

7.1 Overview

The NetSim Cyber GUI provides a guided workflow for configuring and running protocol-aware cyber attacks. The workflow begins by launching NetSim Cyber, selecting the appropriate deployment type, and building the communication scenario between the source and destination nodes. A Threat Agent is then placed in the communication path, an Application Traffic Filter is configured to identify the target protocol stream, and the required built-in or custom attacker is selected. After reviewing the Run Simulation summary, the simulation is started and the attacker interface is used to modify, or attack the selected live traffic stream

7.2 End-to-End Workflow

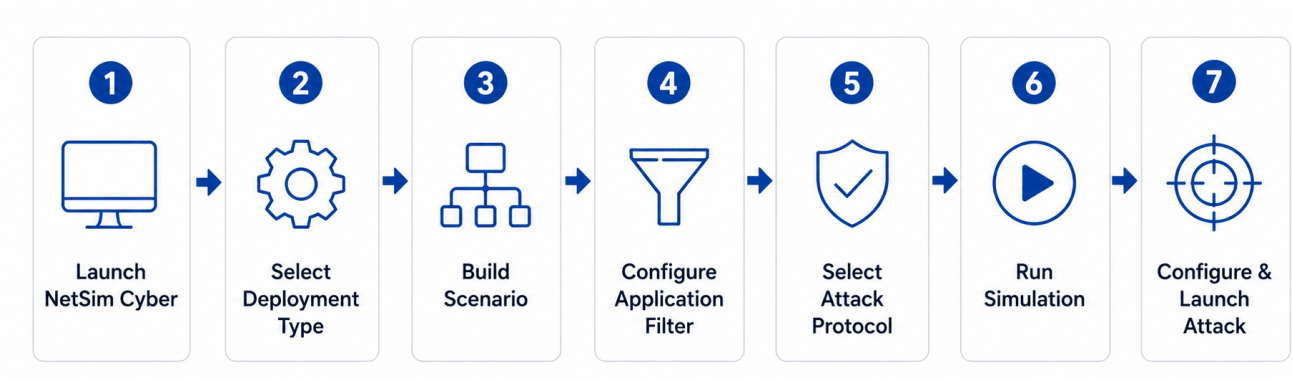


Figure 7-1: *NetSim Cyber GUI workflow for configuring and running a protocol-aware attack experiment.*

7.3 Step-by-Step Walkthrough

7.3.1 Launch NetSim Cyber

Start NetSim Cyber from the NetSim home screen. Open the required Cyber simulation or create a new simulation before configuring protocol traffic and attacks.

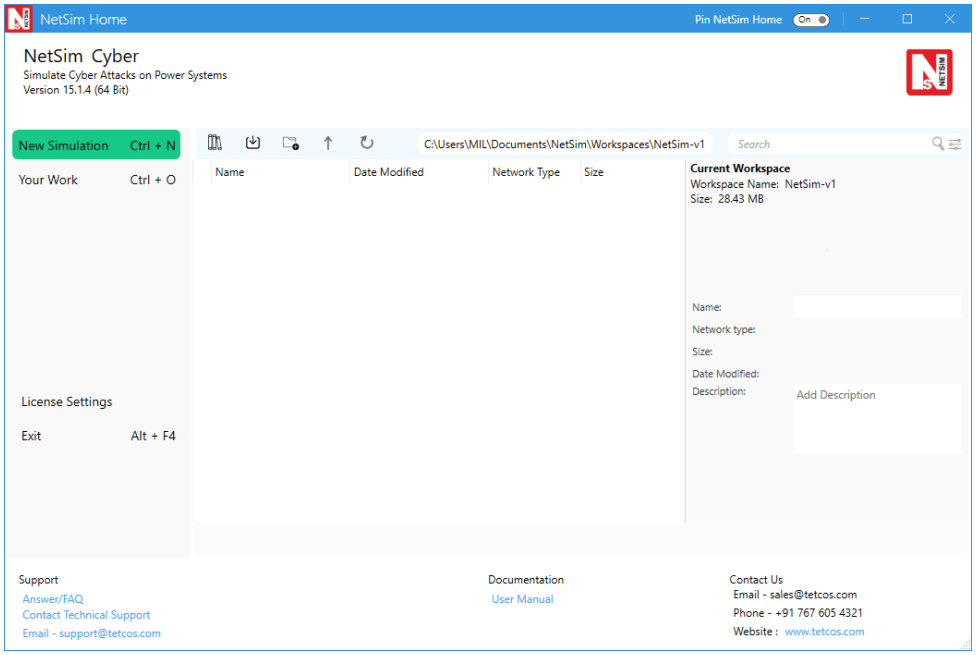


Figure 7-2: *NetSim Cyber home screen showing the workspace and new simulation controls.*

7.3.2 Select the Deployment Type

In the Cyber Environment Settings window, select the deployment type for the experiment.

- **Option A:** Select this option when the protocol traffic originates from or terminates on the same system where NetSim Cyber is installed.
- **Option B:** Select this option when protocol traffic passes through a system where NetSim Cyber is installed between external source and destination devices.

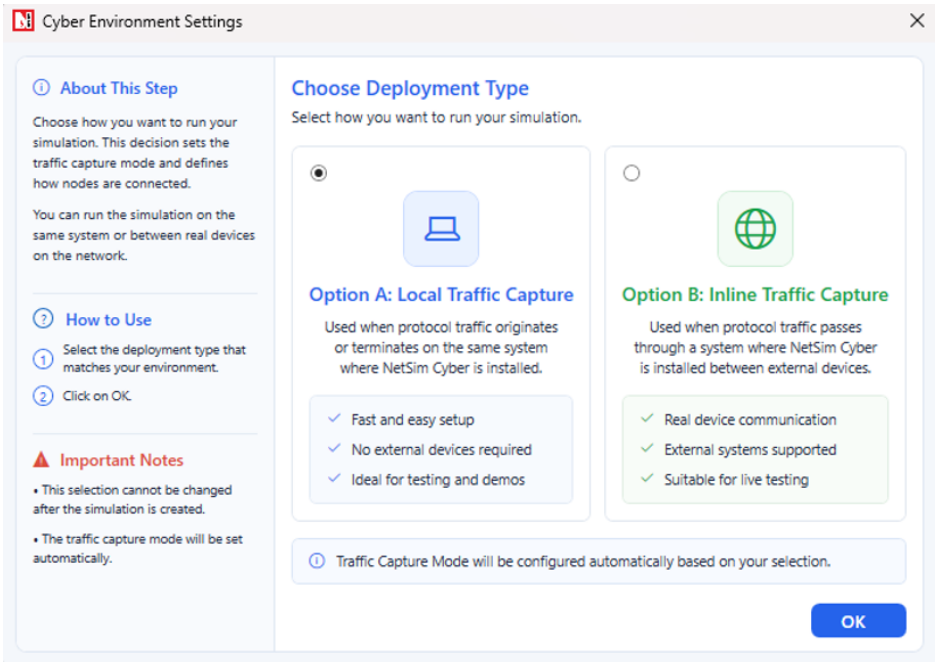


Figure 7-3: *Cyber Environment Settings window for selecting Local Traffic Capture or Inline Traffic Capture.*

7.3.3 Build the Scenario

On the Grid, add the source and destination nodes and place a Threat Agent between them. The nodes are connected automatically when they are added to the scenario. Verify the scenario layout before proceeding to the next configuration step.

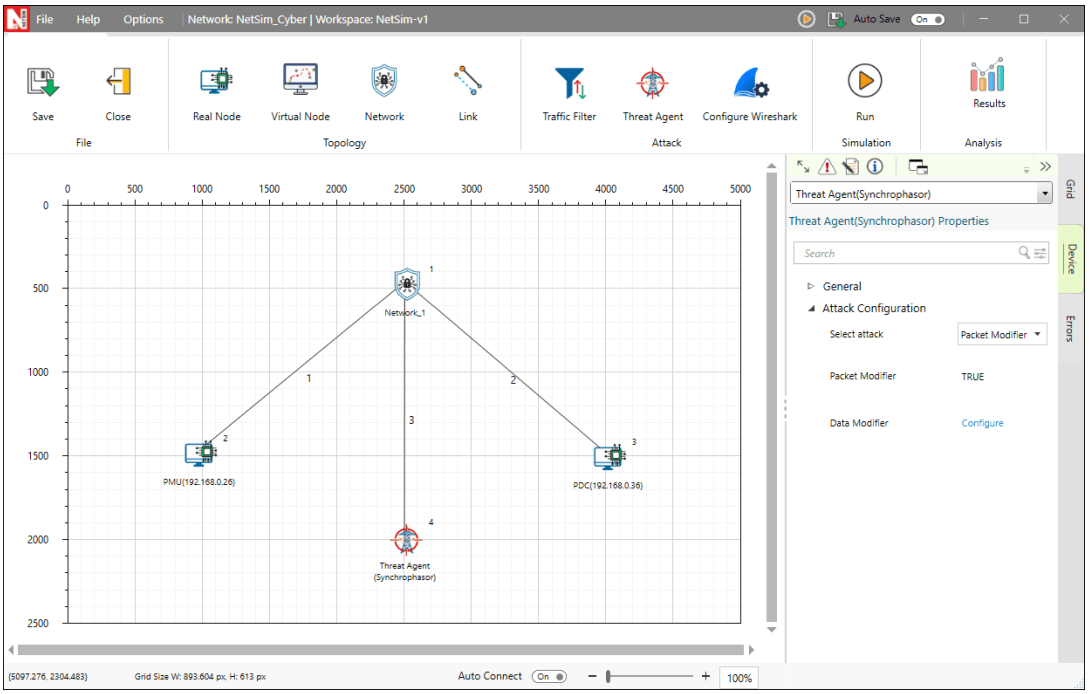


Figure 7-4: *NetSim Cyber scenario canvas showing source and destination real nodes connected through the network path with a Threat Agent placed for protocol attack execution.*

7.3.4 Configure the Application Traffic Filter

Configure the source node, destination node, protocol, and port so the Threat Agent applies the attack only to the intended traffic stream. Typical ports include Modbus TCP 502, DNP3 20000, and the configured synchrophasor PMU/PDC TCP port such as 4712 or 1410.

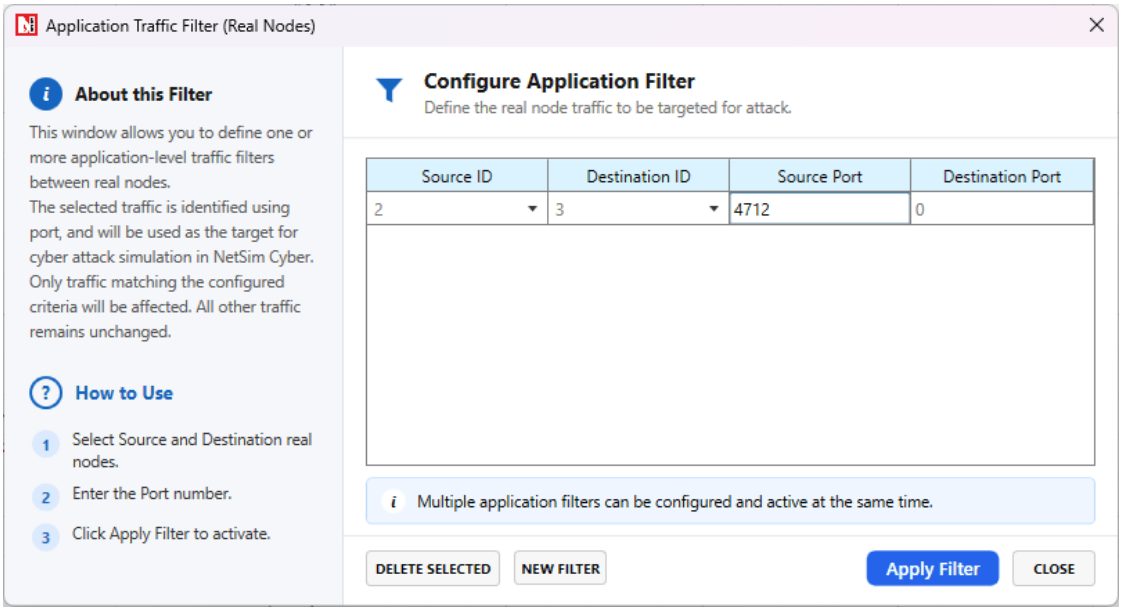


Figure 7-5: *Application Traffic Filter window used to map source node, destination node, and protocol port.*

7.3.5 Select the Attack Protocol

In the Select Attack Protocol window, choose the built-in attacker that matches the traffic generated by the experiment:

Modbus: Modbus TCP traffic

Synchrophasor: IEEE C37.118 traffic, with the required 2005 or 2011 profile

DNP3: DNP3 master/outstation traffic

Routed GOOSE: IEC 61850 GOOSE traffic routed through NetSim Cyber

If the required behavior is not covered by a built-in attacker, import a compatible custom Python attacker.

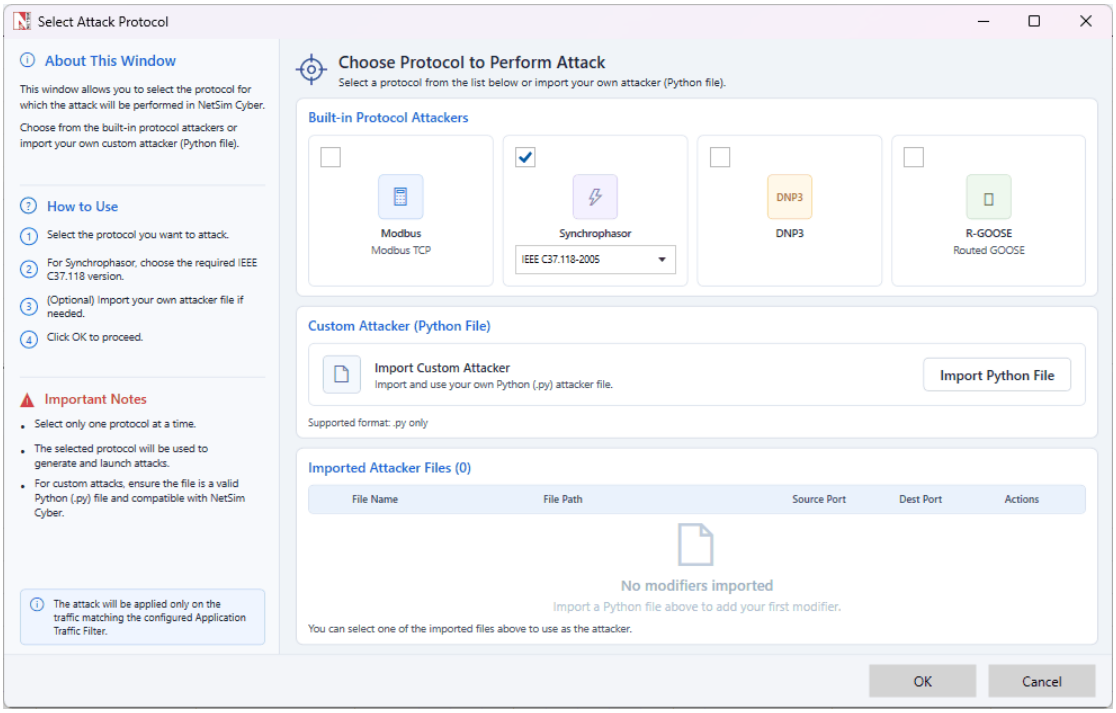


Figure 7-6: *Select Attack Protocol window showing built-in protocol attackers and custom attacker import.*

7.3.6 Run the Simulation

Click Run Simulation. NetSim Cyber starts the configured scenario, opens the required console or attacker windows, and waits for the selected live traffic stream. Start or restart the external simulator traffic only after the NetSim Cyber experiment is ready.

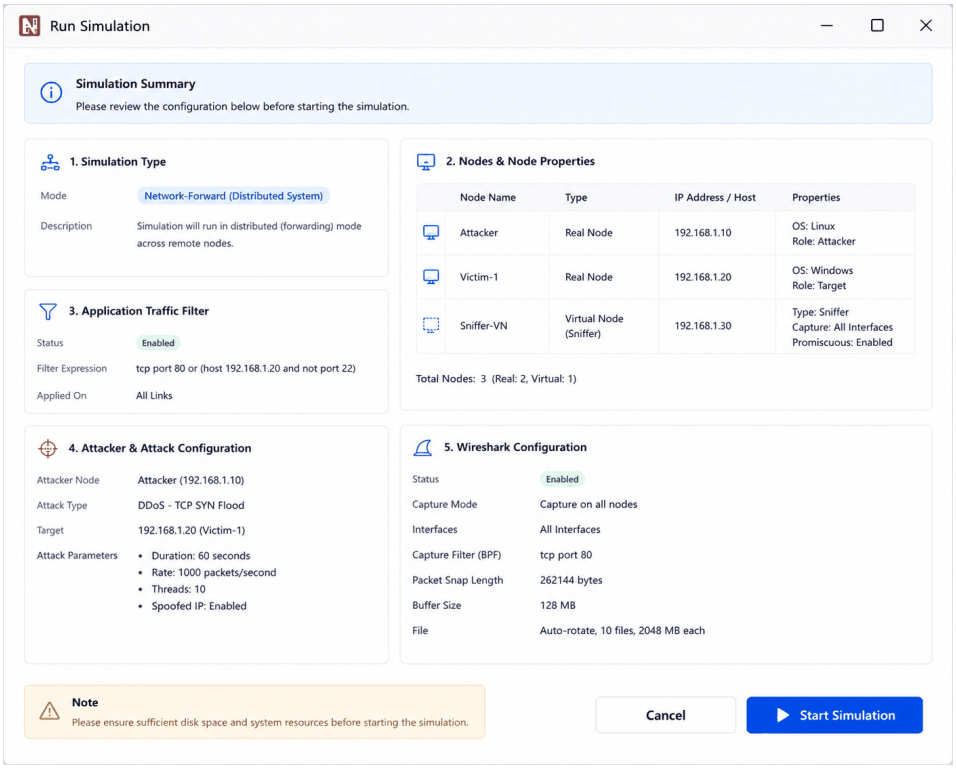


Figure 7-7: *Run Simulation summary window showing node mapping, application filter, and attacker configuration.*

7.3.7 Configure and Launch the Attack

When the attacker UI opens, select the attack type, enter the required parameters, and click Start Attack. Use Modify to update parameters while the attack is active, or Stop Attack to restore normal traffic forwarding.

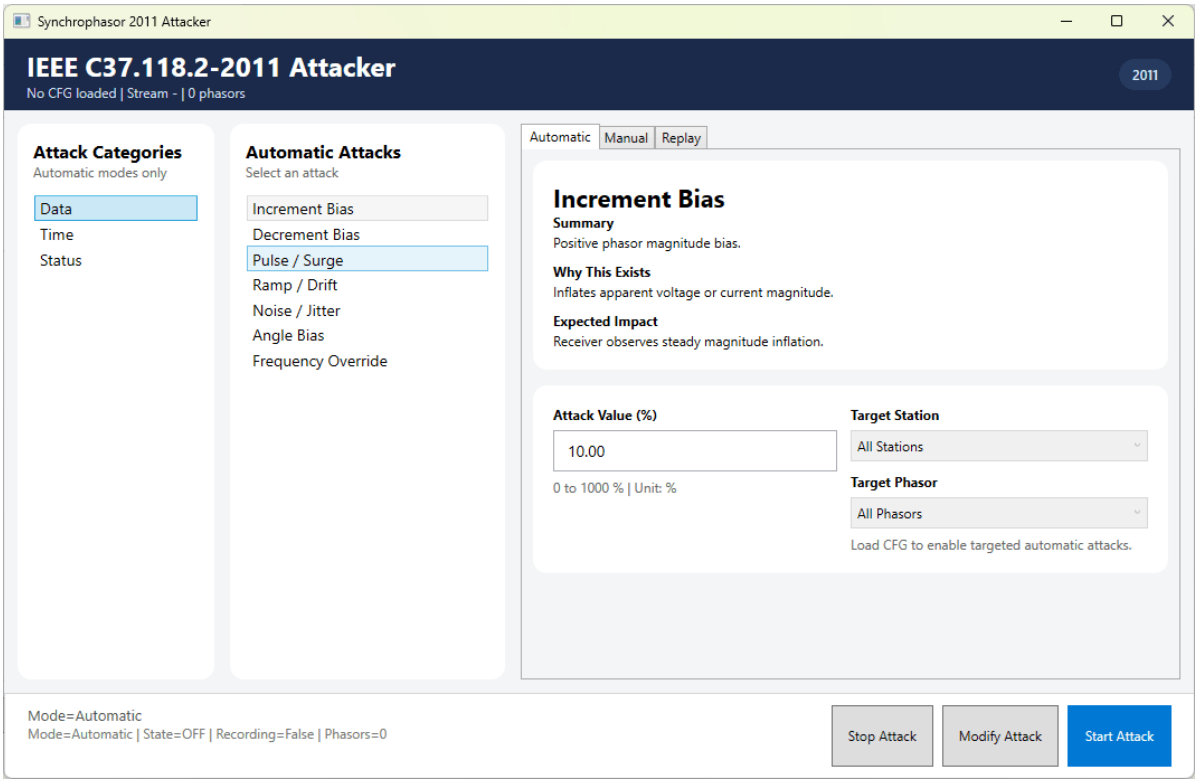


Figure 7-8: *IEEE C37.118.2-2011 attacker UI with automatic, manual, replay, and attack-control options.*

Note: Each protocol has its own attacker UI and parameter set. For Synchrophasor experiments, start NetSim Cyber and keep the attacker ready before starting PMU-PDC traffic, because the first CFG frame is required to load the stream and phasor mapping.

8 Using the Protocol-Specific Attacker UIs

8.1 Purpose and Scope

This section explains how to operate the protocol-specific attacker interfaces supplied with NetSim Cyber for IEEE C37.118 Synchrophasor, Modbus TCP, IEEE 1815/DNP3, and IEC 60870-5-104. It covers the available attack workspaces, target discovery, attack activation, and safe shutdown.

The procedures assume that the correct simulator is running, protocol traffic passes through NetSim Cyber, and the attacker application matches the protocol used by the experiment.

8.2 Synchrophasor Attacker

The IEEE C37.118-2005 and IEEE C37.118.2-2011 attacker windows use the same layout and operating model. Both provide Automatic, Manual, and Replay tabs. The instructions below apply to both profiles.

8.2.1 Configuration Discovery

Station and phasor targeting depends on configuration information observed from the PMU-PDC stream. If the UI reports that no phasor configuration is loaded, generate or request a configuration frame while NetSim Cyber is in the communication path, then reload the available targets.

8.2.2 Automatic Mode

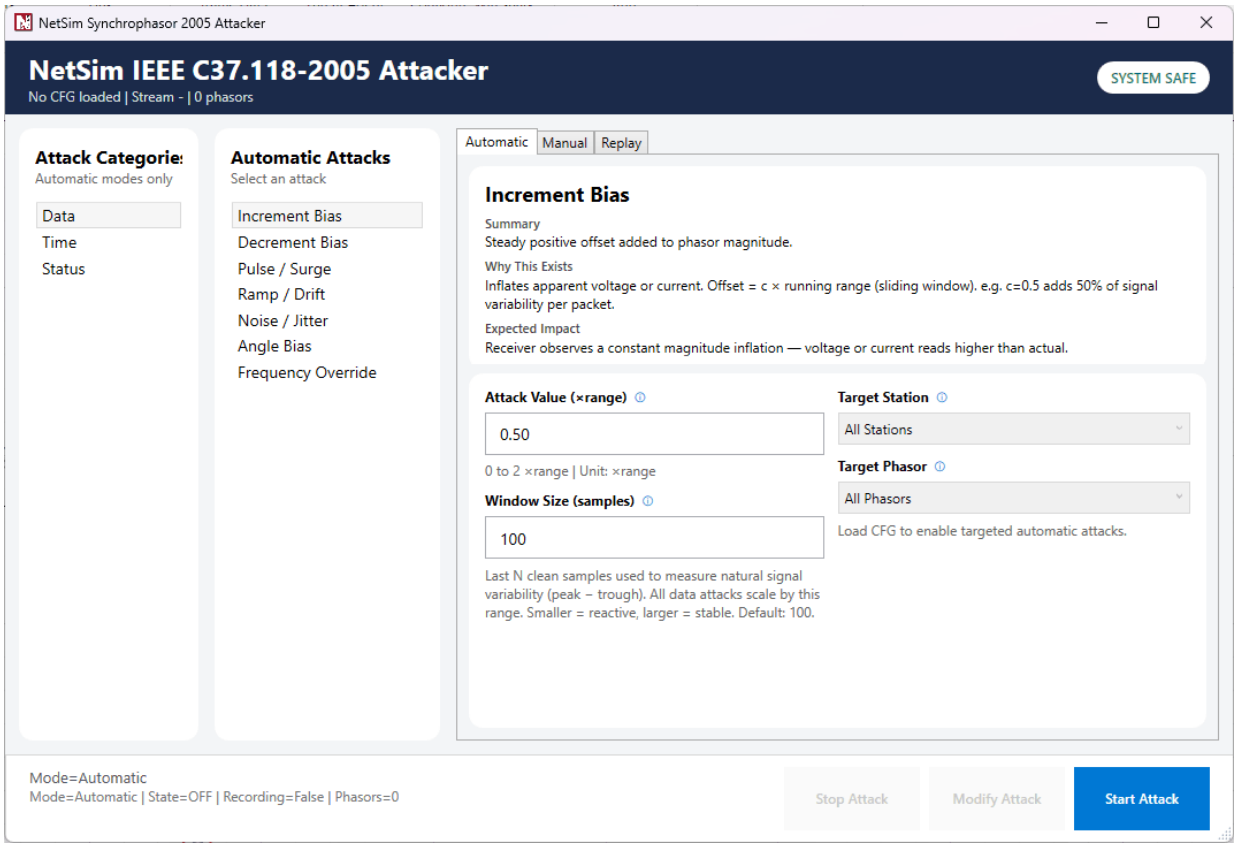


Figure 8-1: *Synchrophasor Automatic tab showing the attack categories, attack list, and controls for Increment Bias.*

Attack	Purpose	Applicable target	Required UI inputs
Increment Bias	Raise selected phasor magnitudes.	Station/phasor	Attack Value, Window Size
Decrement Bias	Lower selected phasor magnitudes.	Station/phasor	Attack Value, Window Size
Pulse / Surge	Apply periodic magnitude surges.	Station/phasor	Attack Value
Ramp / Drift	Change magnitudes progressively.	Station/phasor	Attack Value, Direction, Window Size, optional Max Offset

Attack	Purpose	Applicable target	Required UI inputs
Noise / Jitter	Add random measurement variation.	Station/phasor	Noise Type, Attack Value, probability controls
Angle Bias	Offset the selected phasor angle.	Station/phasor	Attack Value in degrees
Frequency Override	Replace the frequency field.	Selected stream	Attack Value in Hz
Time Sync (SOC)	Shift the whole-second timestamp.	Selected stream	Attack Value in seconds
Fine Time Shift	Shift the sub-second timestamp.	Selected stream	Attack Value in microseconds
STAT Sync Error	Set the synchronization-error status.	Selected stream	Bit Value
STAT Data Modified	Set the data-modified status.	Selected stream	Bit Value
STAT Config Changed	Set the configuration-changed status.	Selected stream	Bit Value
Time Quality Corruption (2011 only)	Change the FRACSEC time-quality value.	Selected 2011 stream	Attack Value from 0 to 15

Packet Attack Probability is available for eligible automatic attacks. Target Station and Target Phasor become usable after the configuration frame has been observed.

Select a category and attack.

Choose the target station and phasor when the selected attack supports targeting.

Set the attack value and any attack-specific controls.

For noise attacks, select Proportional Range, Gaussian, Uniform, or Spike behavior.

Set packet attack probability, window size, ramp direction, or maximum offset when those controls apply.

Review Config Preview and select Start Attack.

Use Modify Attack to update an active configuration or Stop Attack to disable it.

Copyright TETCOS LLP

8.2.3 Manual Mode

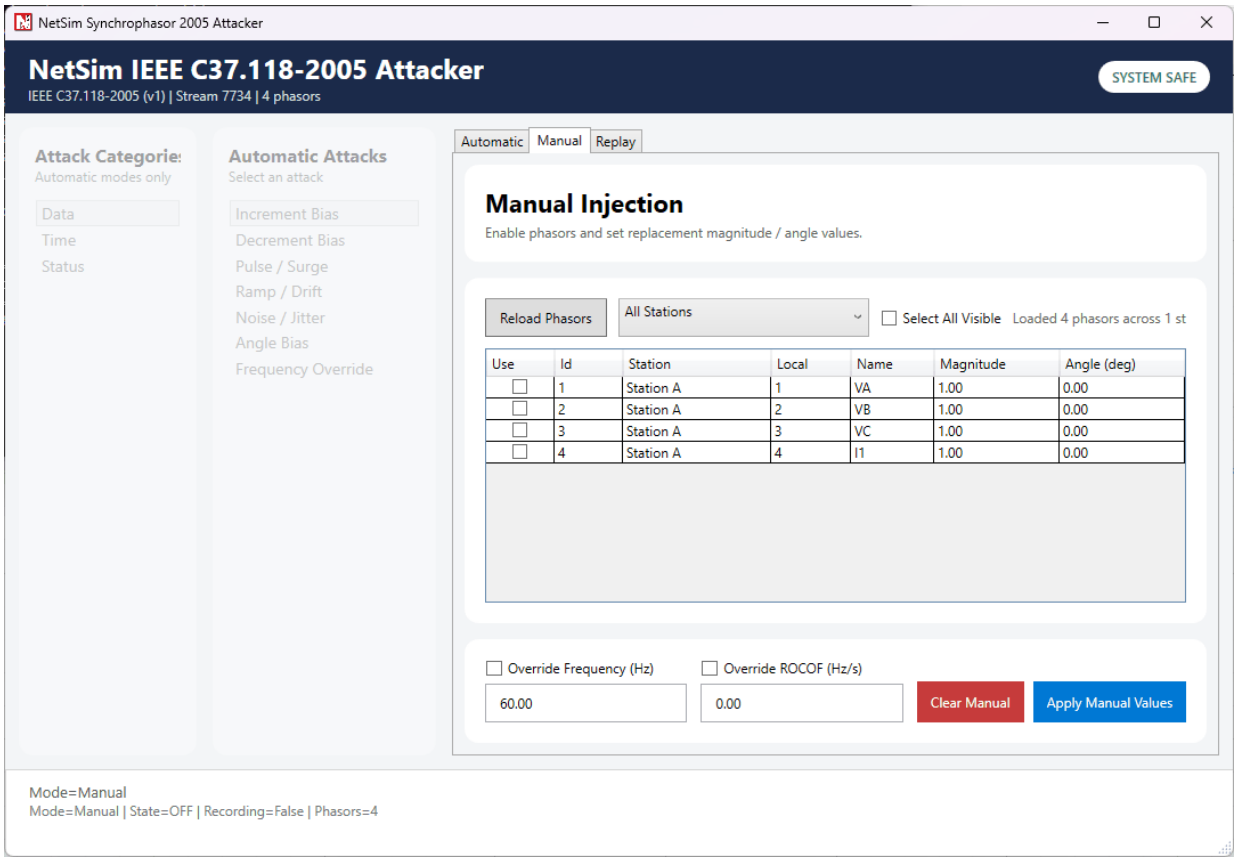


Figure 8-2: *Synchrophasor Manual tab with the phasor list loaded from the configuration frame.*

Select Reload Phasors if the target list is empty or configuration data has changed.

Enable the required phasor rows.

Enter replacement magnitude and angle values.

Optionally enter Frequency Override and ROCOF Override values.

Select Apply Manual Values.

Select Clear Manual when the test is complete.

8.2.4 Replay Mode

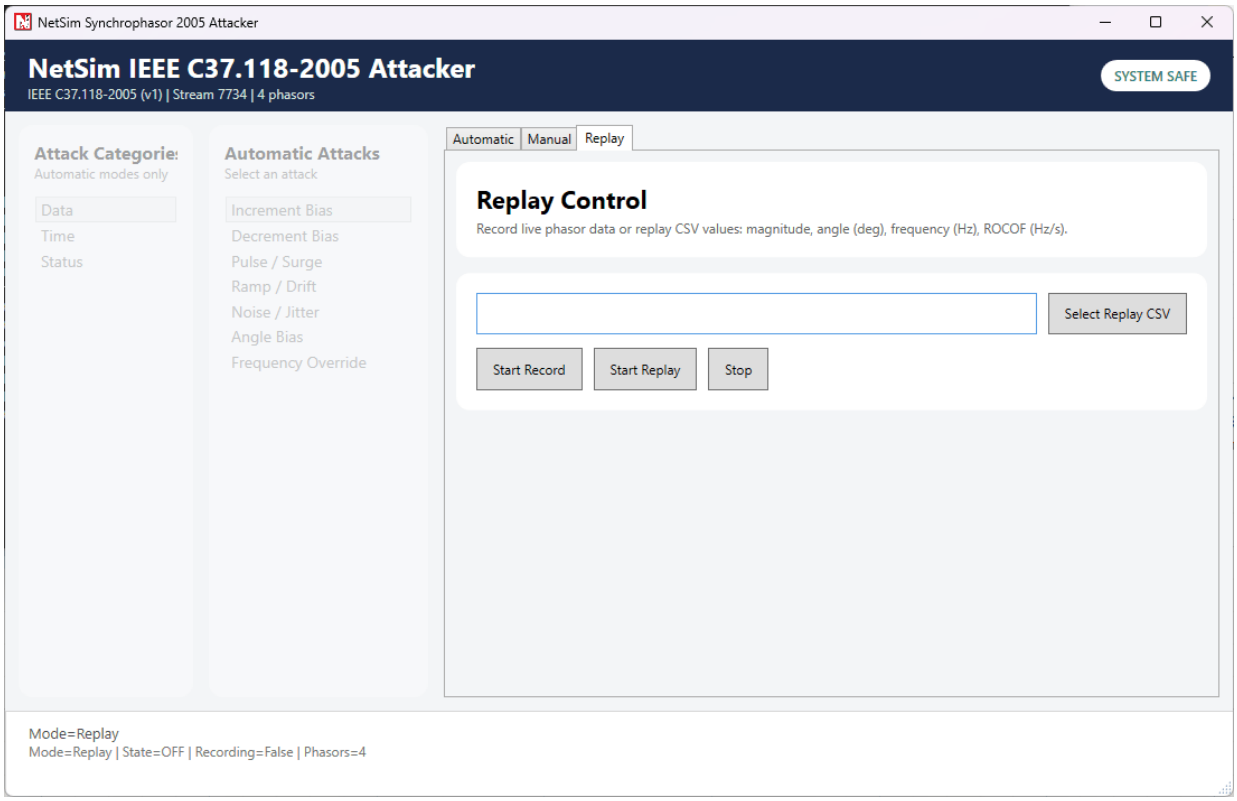


Figure 8-3: *Synchrophasor Replay tab with recording, CSV selection, replay, and stop controls.*

Replay mode records live phasor values or replays values from a CSV file. Replay data can include magnitude, angle in degrees, frequency, and ROCOF.

Select the replay CSV path when using a prepared dataset, or start recording live values.

Allow enough normal data to be recorded for the intended test.

Start replay and monitor the Replay Preview and runtime status.

Select Stop when the scenario is complete.

8.2.5 Profile Difference

The 2005 and 2011 attacker UIs use the same workflow. The 2011 profile additionally provides Time Quality Corruption in the Time category.

8.3 Modbus TCP Attacker

The Modbus TCP attacker modifies selected slave responses after supported master requests. It provides Automatic, Manual, and Replay tabs.

8.3.1 Automatic Mode

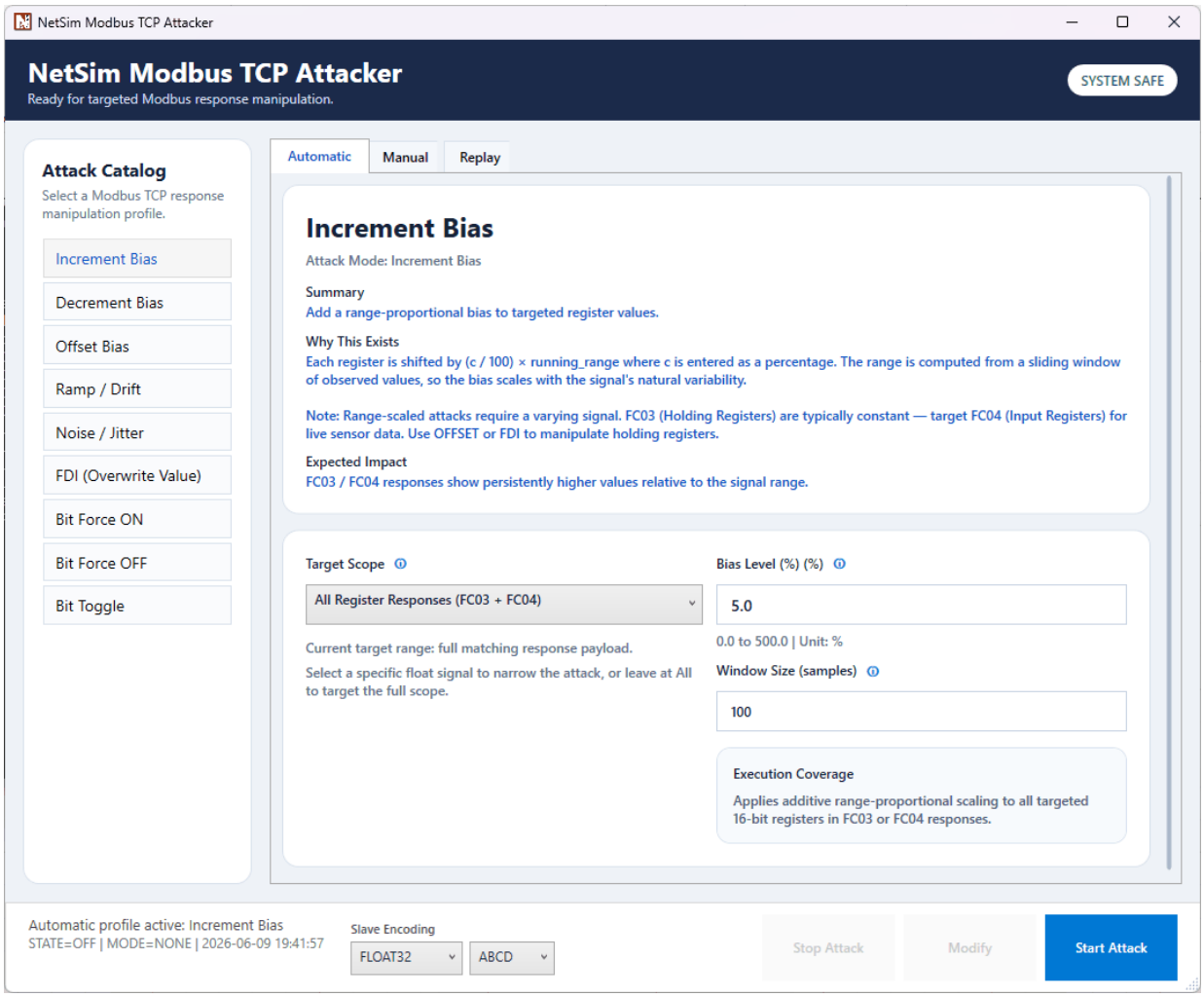


Figure 8-4: *Modbus TCP Automatic tab showing Increment Bias and its target and parameter controls.*

Attack	Purpose	Applicable target	Required UI inputs
Increment Bias	Raise values relative to their observed range.	FC03/FC04 registers	Bias Level, Window Size
Decrement Bias	Lower values relative to their observed range.	FC03/FC04 registers	Bias Level, Window Size
Offset Bias	Add a fixed signed value.	FC03/FC04 registers	Offset Value
Ramp / Drift	Change values progressively.	FC03/FC04 registers	Drift Rate, Direction, Window Size, optional Max Offset
Noise / Jitter	Add configurable random variation.	FC03/FC04 registers	Noise Type, noise value, probability controls

Attack	Purpose	Applicable target	Required UI inputs
FDI (Overwrite Value)	Replace values with a fixed number.	FC03/FC04 registers	Injected Value, encoding, byte order
Bit Force ON	Force selected bits to logic 1.	FC01/FC02 bits	Target scope/signal
Bit Force OFF	Force selected bits to logic 0.	FC01/FC02 bits	Target scope/signal
Bit Toggle	Invert selected bits across responses.	FC01/FC02 bits	Target scope/signal

Select an attack from the Attack Catalog.

Choose the target scope and, when available, a discovered target signal.

Enter the attack value and attack-specific parameters.

For register attacks, select FLOAT32 or UINT16 encoding and the required byte order: ABCD, CDAB, BADC, or DCBA.

Review Config Preview and select Start Attack.

Use Apply Changes to update an active attack or Stop Attack to disable it.

Copyright TETCOS LLP

8.3.2 Manual Mode

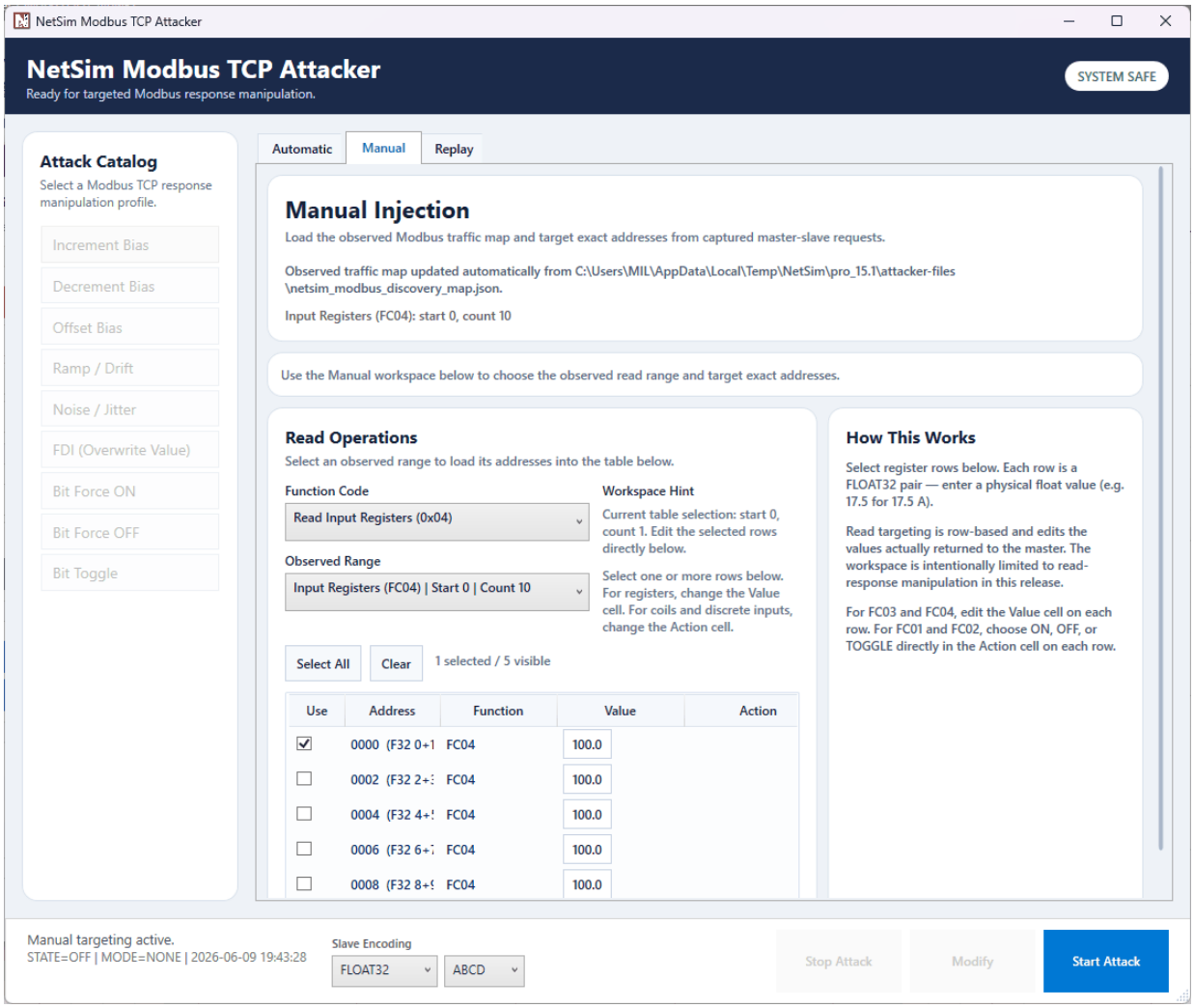


Figure 8-5: *Modbus TCP Manual tab with an observed FC04 address range and editable register rows.*

Manual mode targets addresses discovered from valid Modbus read traffic. It supports FC01, FC02, FC03, and FC04 response modification.

For FC03 and FC04, enable the required rows and enter replacement register values.

For FC01 and FC02, enable the required rows and choose ON, OFF, or TOGGLE.

Use Select All or Clear to manage the visible address rows.

If no observed range is available, first generate normal master-slave read traffic.

8.3.3 Replay Mode

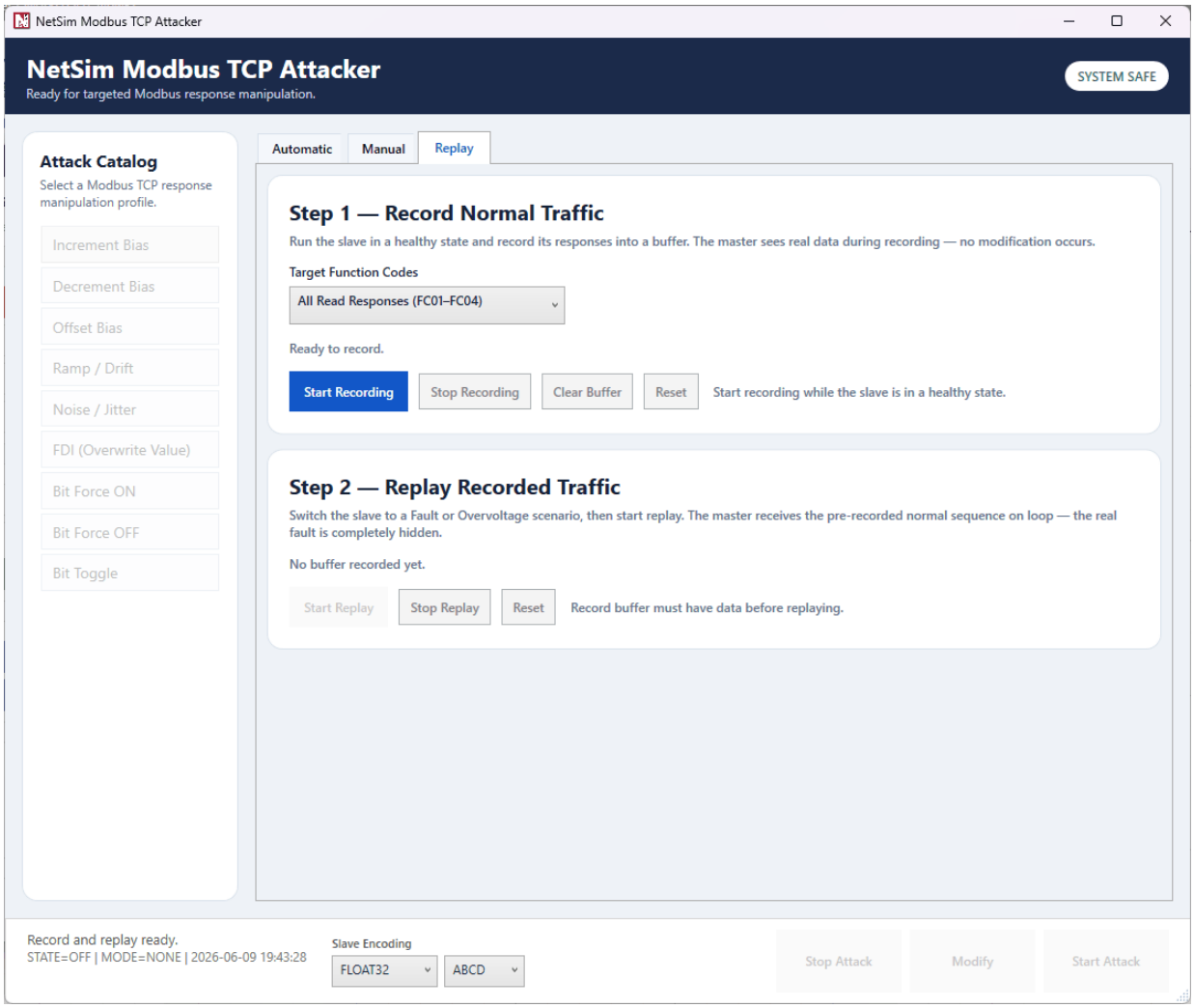


Figure 8-6: *Modbus TCP Replay tab showing the recording and replay workflow.*

Replay mode records supported Modbus read responses and replays the recorded sequence.

Choose all supported read responses or a specific function code: FC01, FC02, FC03, or FC04.

Select Start Recording while the required Modbus traffic is active.

Stop recording after sufficient responses have been captured.

Select Start Replay to use the recorded response sequence.

Select Stop Replay or Reset when the scenario is complete.

8.4 Troubleshooting

Symptom	Likely cause	Operator action
No Modbus address range	No supported read traffic observed	Run normal FC01-FC04 traffic, then reopen or refresh Manual mode.

Symptom	Likely cause	Operator action
No phasor targets	CFG information not observed	Generate or request CFG while NetSim Cyber is in the PMU-PDC path, then reload phasors.
Attack has no visible effect	Wrong target, encoding, profile, or inactive configuration	Check target selection, preview, encoding/profile, and runtime status.
Unexpected values remain	A previous mode is still active	Stop the attack or replay, clear manual values, and confirm inactive status.

9 Attack Library

The NetSim Cyber Attack Library provides a set of configurable cyber-attack models for power-system communication protocols. These attacks are used to study how manipulated, delayed, replayed, or dropped communication traffic can affect monitoring, protection, control, and situational-awareness applications.

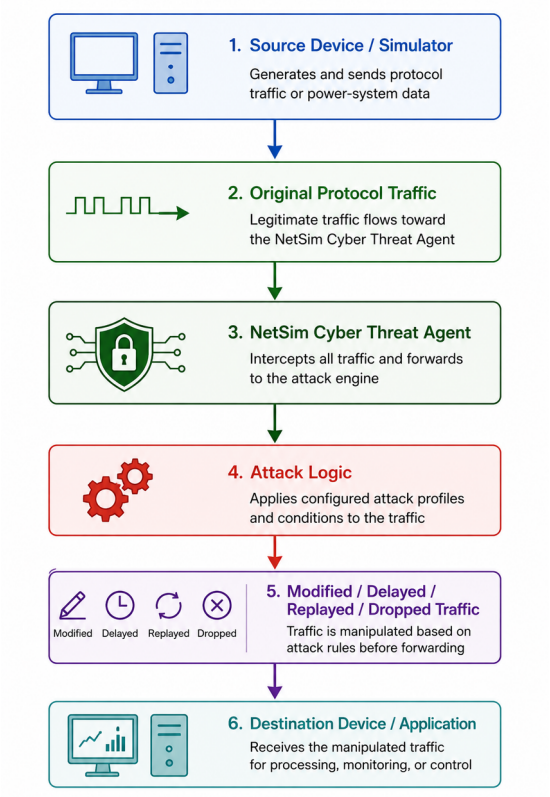


Figure 9-1: *NetSim Cyber attack processing flow*

The attacks described in this section are applied by the NetSim Cyber Threat Agent. The Threat Agent intercepts selected protocol traffic, applies the configured attack logic, and forwards the resulting packet stream to the destination device or application.

The Attack Library supports both signal-level attacks and network/protocol-level attacks.

Signal-level attacks modify measurement values such as voltage, current, frequency, or analog data. Network/protocol-level attacks affect packet timing, packet delivery, protocol fields, or payload contents.

Attack Type	Attack Category	Typical Target	Example Protocols
Noise Injection	Signal-level attack	Voltage, current, analog values	IEEE C37.118, Modbus, DNP3
Increment / Decrement	Signal-level attack	Measurement values, register values	IEEE C37.118, Modbus, DNP3, IEC-104
Ramp Attack	Signal-level attack	Gradually changing measurements	IEEE C37.118, Modbus, DNP3
Pulse / Surge Attack	Signal-level attack	Short-duration measurement spikes	IEEE C37.118, Modbus, DNP3
Frequency and ROCOF Attack	Synchrophasor-specific attack	Frequency and ROCOF fields	IEEE C37.118
Time Synchronization Attack	Timing/data-integrity attack	Timestamp fields	IEEE C37.118, IEC-104, DNP3
Replay Attack	Network/protocol attack	Previously valid packets	GOOSE, DNP3, Modbus, C37.118
Fixed Delay Attack	Network-level attack	Packet delivery time	All supported protocols
PER-Based Drop Attack	Availability / reliability attack	Packet delivery probability	All supported protocols
Packet Manipulation	Protocol-aware payload attack	Protocol-specific fields	All supported protocols
Speed / Bandwidth Suppression	Network performance attack	Available throughput	All supported protocols

9.1 Attack Categories

NetSim Cyber attacks can be grouped into the following categories.

Category	Cyber Action	Possible Power-System Impact
Data integrity attack	Modify measurement or status data	Incorrect monitoring, false alarms, wrong operator interpretation
Timing attack	Delay or shift time information	Event misalignment, delayed control response, synchronization errors
Availability attack	Drop or block traffic	Loss of visibility, communication failure, missing data
Replay attack	Reuse old valid traffic	False event indication or stale measurements
Protocol-field manipulation	Modify selected protocol fields	Protocol-specific impact on SCADA, PDC, IED, or HMI behavior

Copyright TETCOS LLP

9.2 Attack Types

9.2.1 Noise Injection

A noise injection attack adds random variation to a measurement value while keeping the communication stream active. This type of attack is used to simulate low-amplitude measurement corruption, sensor noise, communication-based false data injection, or stealthy manipulation that does not immediately appear as a large disturbance.

Noise injection can be applied to values such as:

- Voltage magnitude
- Current magnitude
- Analog measurements
- Register values
- SCADA telemetry values
- PMU phasor measurements

Attack Logic

The noise amplitude is computed from the recent signal range using a sliding window of the last N clean samples.

$$a = p\,(running\_max - running\_min)$$

where:

- p is the configured noise level.
- $running\_max$ and $running\_min$ are computed from the recent clean samples.

Noise is then generated uniformly within the range:

$$noise \sim \mathcal{U}\left(-\frac{a}{2}, \frac{a}{2}\right)$$

The generated noise is added to the current signal sample.

Typical Use Cases

- Testing measurement quality monitoring
- Evaluating anomaly detection algorithms
- Studying false data injection under low-amplitude conditions
- Generating noisy datasets for AI/ML models
- Testing SCADA or PDC visualization response

Copyright TETCOS LLP

Parameters

Attack Value / Noise Level: noise amplitude as a percentage of the running signal range. In the Threat Agent catalog, the range is 0 to 200% of range, with a default value of 5%.

Window Size: number of recent clean samples used to compute running_min, running_max, and running_range.

Target Station / Target Phasor: optional selectors used to apply the attack only to a specific station or phasor.

9.2.2 Ramp Attack

A ramp attack gradually changes a measurement value over time. Unlike an increment attack, which creates a sudden step change, a ramp attack introduces a slowly increasing or decreasing bias.

Ramp attacks are important because they may resemble real power-system drift, gradual load change, sensor calibration error, or slow voltage variation. This makes them more difficult to detect using simple threshold-based methods.

Attack Logic

A ramp attack injects a linearly increasing offset into the signal starting from the time the attack is enabled.

$$offset(t) = r\,(t - t_{start})$$

The attacked output is computed as:

$$output(t) = signal(t) + offset(t)$$

where:

- r is the configured ramp rate.
- t_{start} is the attack start time.

The recent range is computed from a sliding window of the most recent clean samples. For each incoming sample:

1. Append the clean sample to the buffer.
2. Remove the oldest sample if the buffer exceeds the configured window size.
3. Compute:
    - $running\_min$
    - $running\_max$
    - $running\_range = running\_max - running\_min$

Copyright TETCOS LLP

The per-sample offset increment is calculated as:

$$\delta_{sample} = r \cdot running\_range \cdot \Delta t$$

where Δt is the polling interval in seconds.

The offset is accumulated and applied to the signal. If a maximum offset is configured, the accumulated offset is limited to that value.

Attack Parameters

Attack Value / Ramp Rate: ramp rate in percent of running range per second. In the Threat Agent catalog, the range is 0 to 200% of range/s, with a default value of 2% of range/s.

Window Size: number of recent clean samples used to compute the running range.

Max Offset: optional cap that limits the accumulated ramp offset.

Target Station / Target Phasor: optional selectors used to apply the attack only to a specific station or phasor.

9.2.3 Increment/Decrement Attack

- An increment/decrement attack applies a fixed positive or negative offset to selected measurement values. This creates a step change in the received data while the communication stream remains active.
- An increment attack increases the target value. A decrement attack decreases the target value.
- This attack is useful for studying how monitoring systems respond to sudden measurement bias.

Attack Logic

An increment attack applies a fixed offset to every sample while the attack is active.

The offset is computed from the recent signal range:

$$offset = c \cdot running\_range$$

The attacked output is computed as:

$$output(t) = signal(t) + offset$$

where:

- c is the configured increment magnitude.

The recent range is computed using a sliding window of the most recent clean samples. For each incoming sample:

1. Append the clean sample to the buffer.

2. Remove the oldest sample if the buffer exceeds the configured window size.
3. Compute:
    - $running\_min$
    - $running\_max$
    - $running\_range = running\_max - running\_min$

The computed offset is then applied to the current sample.

Interpretation

An increment/decrement attack appears as a sudden step change in the received measurement. For example:

- Voltage magnitude may appear suddenly higher or lower.
- Current magnitude may show a false load change.
- A Modbus holding register may display an incorrect process value.
- A DNP3 analog input may indicate an abnormal field measurement.

Typical Use Cases

- Testing threshold-based alarms
- Studying false load or voltage changes
- Evaluating SCADA display integrity
- Testing PMU/PDC anomaly detection
- Simulating biased sensor readings

Attack Parameters

- Attack Value: increment or decrement coefficient c. In the Threat Agent catalog, the range is 0 to 2 x running range, with a default value of 0.5 x running range.
- Window Size: number of recent clean samples used to compute the running range.
- Target Station / Target Phasor: optional selectors used to apply the attack only to a specific station or phasor.

9.2.4 Pulse / Surge Attack

A pulse or surge attack periodically multiplies selected phasor magnitude values for a short duration. This attack is used to simulate intermittent spikes, sudden measurement surges, transient disturbances, or repeated false events without continuously corrupting every packet.

Pulse / surge can be applied to values such as:

- Voltage magnitude

- Current magnitude
- Synchrophasor magnitude fields
- SCADA telemetry values
- Analog measurement values
- Register values that represent process measurements

Attack Logic

- The attack is active only during the configured pulse window. During the active window, the selected magnitude is multiplied by the configured surge multiplier.
- output(t) = m * signal(t)
- where:
- m is the configured pulse or surge multiplier.
- The active pulse window is selected from the packet cycle.
- Outside the pulse window, the packet is forwarded without this magnitude multiplication.

Typical Use Cases

- Testing spike detection and transient-event handling
- Evaluating PMU/PDC visualization response to repeated surges
- Studying false disturbance or false fault indications
- Generating intermittent corrupted measurements for AI/ML datasets
- Testing alarm debounce and event filtering logic

Parameters

- Attack Value: pulse multiplier m. In the Threat Agent catalog, the range is 0 to 100 x, with a default value of 5 x.
- Pulse Interval: packet interval after which the pulse cycle repeats.
- Pulse Duration: number of packets for which the surge remains active in each cycle.
- Target Station / Target Phasor: optional selectors used to apply the attack only to a specific station or phasor.

9.2.5 Frequency / ROCOF Attack

A frequency / ROCOF attack modifies the frequency-related fields in synchrophasor data frames. This attack is used to simulate off-nominal system frequency, false frequency excursions, or incorrect rate-of-change-of-frequency indications while the rest of the frame continues to appear valid.

Frequency / ROCOF manipulation can be applied to values such as:

- Frequency field in IEEE C37.118 data frames
- ROCOF field in IEEE C37.118 data frames
- PMU frequency telemetry
- PDC frequency display values
- Frequency-related monitoring and protection inputs

Attack Logic

- The frequency attack overwrites the frequency field with the configured target frequency.
- frequency_out = configured_frequency
- If ROCOF override is configured, the ROCOF field is overwritten with the configured target ROCOF value.
- rocof_out = configured_rocof
- For floating-point frame formats, the configured values are written directly. For integer frame formats, frequency is encoded as the difference from nominal frequency and ROCOF is encoded using the protocol scaling.

Typical Use Cases

- Testing frequency-limit alarms
- Evaluating ROCOF-based disturbance detection
- Studying false frequency excursion scenarios
- Testing synchrophasor analytics and event classifiers
- Generating frequency-corrupted datasets for AI/ML models

Parameters

- Attack Value: configured frequency. In the Threat Agent catalog, the Frequency Override range is 40 to 70 Hz, with a default value of 59 Hz.
- ROCOF Override: optional configured ROCOF value when ROCOF manipulation is enabled through the attack configuration.
- Target Station: optional selector used to apply the attack only to a specific station.
- Frame Format: determines whether frequency and ROCOF are written as floating-point values or protocol-scaled integer values.

9.2.6 Time Synchronization Attack

A time synchronization attack modifies the timestamp fields of synchrophasor data frames while leaving the measurement stream active. This attack is used to simulate clock skew, wrong event ordering, timing-quality degradation, or misalignment between PMU streams.

Time synchronization manipulation can be applied to values such as:

- SOC timestamp field
- FRACSEC sub-second timestamp field
- FRACSEC time-quality bits
- PMU event timing
- PDC time-alignment logic

Attack Logic

- A whole-second time shift modifies the SOC field by the configured number of seconds.
- SOC_out = SOC_in + delta_seconds
- A fine time shift modifies only the lower 24-bit FRACSEC counter while preserving the upper time-quality and leap-second flags.
- FRACSEC_out = FRACSEC_in + delta_counts
- For IEEE C37.118.2-2011 streams, the time-quality bits can also be overwritten to indicate degraded or faulty synchronization.

Typical Use Cases

- Testing PDC time-alignment behavior
- Evaluating event ordering and timestamp validation
- Studying clock-skew attacks on PMU streams
- Testing bad-time-quality handling
- Generating time-shifted synchrophasor datasets for AI/ML models

Parameters

- Time Sync (SOC) Attack Value: whole-second shift. In the Threat Agent catalog, the range is -3600 to +3600 s, with a default value of 0 s.
- Fine Time Shift Attack Value: sub-second FRACSEC shift. In the Threat Agent catalog, the range is -1,000,000 to +1,000,000 microseconds, with a default value of 1000 microseconds.
- Time Quality: IEEE C37.118.2-2011 time-quality value. In the Threat Agent catalog, 0 indicates locked time and 15 indicates a fault.

9.2.7 Replay Attack

A replay attack uses previously captured protocol data at a later time. In the current NetSim Cyber attacker implementation, the Threat Agent intercepts a matching live packet, replaces its protocol body with a previously captured body, and forwards the modified packet. It does not inject an additional packet into the traffic stream. Therefore, replay does not increase the packet count. The captured body and the live packet body must have the same length.

Copyright TETCOS LLP

Replay attacks are relevant for protocols where message freshness, timestamp validation, or sequence validation is important.

Common replay targets include:

- IEC 61850 GOOSE event messages
- IEEE C37.118 synchrophasor data frames
- DNP3 responses or events
- Modbus responses
- IEC 60870-5-104 telemetry messages

Replay Attack Logic

Let:

$P(t)$ = matching live packet received at time t
$B(t)$ = protocol body of the live packet
B_r = previously captured protocol body

During replay, the Threat Agent replaces $B(t)$ with B_r and forwards the modified live packet:

$$P'(t) = P(t) \text{ with } B(t) \text{ replaced by } B_r$$

Replay is applied only when the captured body and the live body have the same length:

$$|B_r| = |B(t)|$$

The Threat Agent forwards one modified packet for each intercepted live packet. It does not create an additional replay packet.

Possible Impact

A replay attack can cause the destination application to receive stale but valid-looking information. This may result in repeated measurements, false status indication, incorrect event interpretation, or loss of message freshness.

Protocol-Specific Replay Controls

Replay capture and control differ by protocol.

- **Modbus:** `RECORD` stores supported Modbus responses in a buffer organised by function code. `REPLAY` uses a recorded body to modify matching live responses.
- **Synchrophasor:** Recording is a flag used together with `REPLAY` and is not a separate mode. Replay can use an in-memory recording buffer or a CSV replay source.
- **R-GOOSE:** R-GOOSE has no separate record mode. When `REPLAY` starts, the attacker captures the first matching R-GOOSE frame. It then uses the captured body to overwrite later matching frames of the same length. Changing the attacker mode clears the replay buffer.

Copyright TETCOS LLP

R-GOOSE Replay Considerations

R-GOOSE replay is controlled through the `MODE` keyword in `attack_config_rgoose.txt`.

To replay a trip body, `REPLAY` must be enabled while the trip frame is on the wire. The first matching frame captured after `REPLAY` starts must contain the required trip body. The mode must remain unchanged while later matching frames are overwritten because changing the mode clears the replay buffer.

Pure replay and field adjustment must be treated as separate cases. The R-GOOSE `MODE` keywords that modify freshness-related protocol fields are:

- `STNUMRESET`: sets `stNum` to 1.
- `STNUMSPOOF`: sets `stNum` to a specified value.
- `SQNUMCORR`: adds a delta to `sqNum`.
- `TALLZERO`: sets `timeAllowedtoLive` to zero.
- `TALLINFLATE`: increases `timeAllowedtoLive`.
- `TALLSET`: sets `timeAllowedtoLive` to a specified value.

A subscriber may reject pure replay when the captured state number, sequence number, or time-allowed-to-live value is stale. Tests using the field-adjustment modes must therefore be identified separately from pure `REPLAY` mode.

9.2.8 Fixed Delay Attack

A fixed delay attack intentionally delays selected packets by a constant configured time before forwarding them to the destination. The packet contents are not necessarily changed, but the destination receives the data later than expected.

This attack is useful for evaluating the sensitivity of power-system communication applications to latency. It is especially relevant for time-sensitive systems such as Synchrophasor monitoring, IEC 61850 event messaging, SCADA polling, and telecontrol communication.

Fixed Delay Attack Logic

Let:

P(t) = packet received by the Threat Agent at time t
D = configured fixed delay

The packet is forwarded at:

$$t_{forward} = t + D$$

where D remains constant for the duration of the attack unless the user changes the attack parameter.

Interpretation

A fixed delay attack creates a constant additional latency in the communication path. The destination still receives packets, but every affected packet arrives later than it would under normal conditions.

For example:

- A PMU data frame may reach the PDC late.
- A GOOSE or R-GOOSE event may be delayed.
- A SCADA response may appear slow.
- A control-center application may receive stale telemetry.
- A monitoring dashboard may show lagging data.

Typical Use Cases

- Testing latency tolerance of PDCs and SCADA applications.
- Studying delayed event reporting.
- Evaluating time-sensitive monitoring and control applications.
- Testing alarm or timeout behavior.
- Simulating network congestion-like latency.

9.2.9 PER-Based Drop Attack

A PER-based drop attack drops packets according to a configured **Packet Error Rate**, or **PER**. Instead of dropping every packet or every fixed Nth packet, the Threat Agent probabilistically drops packets based on the configured PER value.

This attack is used to emulate unreliable communication links, packet loss, wireless errors, congestion loss, or degraded network conditions.

PER-Based Drop Logic

Let:

PER = configured packet error rate
r = random value uniformly generated between 0 and 1

For each selected packet:

If r < PER:
Drop the packet
Else:
Forward the packet

For example:

PER = 0.05

means approximately 5% of the selected packets are dropped over time.

Interpretation

A PER-based drop attack creates communication loss based on probability. The destination may observe missing measurements, gaps in time-series data, communication retries, reduced data quality, or protocol timeouts depending on the protocol and application behavior.

For example:

- A PDC may show missing PMU samples.
- A SCADA master may miss responses and retry.
- A Modbus master may experience read failures.
- A DNP3 master may observe missing or delayed telemetry.
- A GOOSE subscriber may miss selected event messages.

Typical Use Cases

- Testing application tolerance to packet loss.
- Studying data gaps in PMU/PDC systems.
- Evaluating SCADA retry behaviour.
- Simulating lossy communication links.
- Generating packet-loss datasets for detection algorithms.

Attack Parameters

Packet Error Rate (%) – Specifies the percentage of packets that will be intentionally corrupted during transmission. Higher values increase the number of affected packets and the likelihood of communication errors.

9.2.10 Speed / Bandwidth Suppression Attack

A speed or bandwidth suppression attack reduces the effective communication rate available to the selected traffic stream. Instead of completely blocking packets, the Threat Agent limits how quickly traffic is forwarded.

This attack emulates constrained bandwidth, throttling, congestion, or intentional degradation of network performance.

Speed Suppression Logic

Let:

$B_{configured}$ = configured bandwidth limit
B_{normal} = normal available bandwidth

During the attack, the effective forwarding rate is limited to:

$$B_{effective} = B_{configured}$$

where:

$$B_{configured} < B_{normal}$$

If incoming traffic exceeds the configured bandwidth limit, packets may be queued, delayed, or dropped depending on the implementation and buffer behavior.

Interpretation

Speed suppression does not necessarily modify packet contents. Instead, it degrades communication performance by reducing throughput. This may cause increased latency, buffering, delayed delivery, application timeouts, or packet loss when buffers overflow.

For example:

- PMU data may arrive late or in bursts.
- SCADA polling cycles may become slower.
- DNP3 telemetry responses may be delayed.
- Modbus read/write operations may take longer.
- IEC-104 data transfer may show reduced responsiveness.

Typical Use Cases

- Testing communication performance under bandwidth limitation.
- Simulating congested or low-speed links.
- Evaluating protocol behaviour under constrained throughput.
- Studying delayed monitoring and control response.
- Testing application timeout and buffering behaviour.

9.2.11 DoS Attack

A DoS attack degrades the selected communication path while keeping the Threat Agent active in the traffic flow. This type of attack is used to simulate reduced network availability, constrained link performance, added latency, or unreliable packet delivery.

DoS can be applied to communication paths carrying traffic such as:

- SCADA polling traffic
- PMU to PDC synchrophasor streams
- Modbus TCP communication
- DNP3 communication
- IEC 60870-5-104 communication
- Custom TCP/IP application traffic

Attack Logic

- The DoS option uses the configured speed, delay, and packet error rate parameters to degrade the selected traffic path.
- If a fixed delay is configured, the packet forwarding time is: $t_{forward} = t + D$, where D is the configured fixed delay and t is the time at which the packet reaches the Threat Agent.

- If packet error rate is configured, each selected packet is dropped based on the configured probability: drop = (r < PER), where r is a random value between 0 and 1 and PER is the configured packet error rate.
- If speed is configured, the effective forwarding rate is limited to the configured value.

Typical Use Cases

- Testing application behavior under reduced link speed.
- Studying delayed telemetry delivery.
- Evaluating timeout and retry behavior.
- Simulating unreliable communication links.
- Testing SCADA, PMU/PDC, or custom TCP/IP communication during degraded network conditions.

Parameters

- Speed (Mbps): Configures the effective link speed or bandwidth limit.
- Fixed Delay (ms): Adds fixed delay to packets passing through the Threat Agent.
- Packet Error Rate (%): Configures packet error or drop probability.

9.2.12 Virtual SYN Flood Attack

A Virtual SYN Flood attack emulates TCP SYN flood behavior without generating real SYN packets on the network. This type of attack is used to study TCP connection pressure, timeout behavior, and availability effects in a controlled way.

Virtual SYN Flood can be applied to TCP-based experiments such as:

- TCP client-server communication
- SCADA over TCP/IP
- Modbus TCP communication
- DNP3 over TCP communication
- IEC 60870-5-104 communication
- Custom TCP/IP application traffic

Attack Logic

- The Threat Agent creates repeated virtual TCP connection attempts based on the configured frequency. These virtual attempts are used to model connection pressure without sending actual SYN flood traffic onto the network.
- The virtual event interval is controlled by the configured event rate. TCP behavior during the virtual attack is influenced by the configured ACK handling, initial RTO, TCP timeout, and maximum TCP connection values.
- If frequency is configured as events per second, the virtual event interval is: delta_t = 1 / frequency.

Copyright TETCOS LLP

Typical Use Cases

- Studying TCP connection pressure without real flood traffic.
- Testing TCP timeout behavior.
- Evaluating maximum connection handling.
- Comparing virtual TCP pressure with real SYN flood behavior.
- Testing availability behavior in a safer lab configuration.

Parameters

- Enable Virtual SynFlood: Enables or disables the virtual SYN flood attack.
- Frequency: Controls how often virtual SYN events are generated.
- Ignore TCP ack: Controls whether TCP ACK handling is ignored by the virtual attack model.
- Initial RTO (s): Sets the initial retransmission timeout value used for TCP behavior modeling.
- TCP Timeout (s): Sets the TCP timeout value used by the model.
- Max TCP connections: Sets the maximum number of TCP connections allowed in the model.

9.2.13 Real SYN Flood Attack

A Real SYN Flood attack generates real TCP SYN traffic toward a configured server. This type of attack is used to study how TCP servers and applications behave when repeated connection attempts are directed toward a target endpoint.

Real SYN Flood can be applied to TCP-based applications such as:

- TCP server applications
- Modbus TCP servers
- DNP3 TCP outstations or masters
- IEC 60870-5-104 endpoints
- Custom TCP/IP applications
- Test traffic generators such as iPerf

Attack Logic

- The Threat Agent sends repeated TCP SYN packets to the configured server address and server port. The configured frequency controls how often SYN packets are generated.
- If frequency is configured as packets per second, the SYN generation interval is: delta_t = 1 / frequency.
- The attack uses the configured client and server endpoint values to form the TCP connection attempt.

Typical Use Cases

- Testing TCP server behavior under repeated connection attempts.
- Studying connection backlog or availability effects.
- Observing application behavior when new TCP sessions become slow or unreliable.
- Comparing real SYN traffic with Virtual SYN Flood behavior.
- Generating controlled TCP availability test traffic in a lab.

Parameters

- Enable Real SynFlood: Enables or disables real SYN flood packet generation.
- Frequency: Controls how often SYN packets are generated.
- Server Address: IP address of the target server.
- Server Port: TCP port of the target server.
- Client Address: Source or client IP address used for the attack traffic.
- Client Port: Source or client TCP port used for the attack traffic.

9.2.14 RST Flood Attack

An RST Flood attack sends TCP reset packets using the configured client and server endpoint values. This type of attack is used to study TCP session interruption and application reconnection behavior.

RST Flood can be applied to TCP-based communication such as:

- Long-running TCP sessions
- SCADA communication over TCP/IP
- Modbus TCP communication
- DNP3 over TCP communication
- IEC 60870-5-104 communication
- Custom TCP/IP application sessions

Attack Logic

- The Threat Agent sends TCP reset packets for the configured endpoint pair. A TCP reset packet requests immediate termination of a TCP connection when the receiving endpoint accepts it as valid for the active session.
- The attack uses the configured endpoint tuple consisting of client address, client port, server address, and server port.
- If the reset packet matches an active TCP flow and is accepted by the endpoint, the session may be terminated. If the endpoint does not accept the reset packet as valid, the packet may be ignored.

Copyright TETCOS LLP

Typical Use Cases

- Studying TCP session interruption.
- Testing application reconnection behavior.
- Observing how monitoring or control applications respond to reset events.
- Testing resilience of long-running TCP communication.
- Evaluating logs and alarms for unexpected TCP session resets.

Parameters

- Enable RST Flood: Enables or disables repeated TCP reset packet generation.
- Server Address: Destination server IP address.
- Server Port: Destination server TCP port.
- Client Address: Source or client IP address used in the reset packet.
- Client Port: Source or client TCP port used in the reset packet.

9.3 Custom Payload Modifier API

NetSim Cyber allows users to provide custom packet-modification logic through payload_modifier_custom.py. This file is intended for customer-defined attacks when the built-in protocol attackers do not cover the required behavior. NetSim calls the custom function for every packet that matches the Application Traffic Filter and passes through the configured Threat Agent.

9.3.1 Purpose

Use the custom payload modifier when a user needs to test protocol-specific logic, field replacement, byte-level corruption, packet drop, or additional packet delay without developing a full built-in attacker UI. The custom script operates on the application-layer payload. The NetSim Application Traffic Filter should be used to select the required protocol or port before packets reach the custom modifier.

9.3.2 API Entry Point

The customer should implement attack logic inside the following function:

```
NETSIM_PACKET_API_VERSION = 2

def modify_payload(ipv4, ipv6, tcp, udp, payload, iopath, context=None):
```

NETSIM_PACKET_API_VERSION must be the exact integer 2 to enable the unified seven-argument callback. NetSim validates during initialization that the configured function accepts all seven positional arguments. An invalid marker or marker/signature mismatch causes Packet Modifier initialization to fail. Scripts without this marker do not receive context (src, dst -> socket address) as an argument.

9.3.3 Function Parameters

Parameter	Description	Common Fields / Use
ipv4	Network-layer packet handle supplied by NetSim. It may be an internal NetSim object.	Usually ignored for simple payload-only attacks.
ipv6	Network-layer packet handle supplied by NetSim. It may be an internal NetSim object.	Usually ignored for simple payload-only attacks.
tcp	Transport-layer packet handle supplied by NetSim. It may be an internal NetSim object.	Use the NetSim Application Traffic Filter to select TCP traffic.
udp	Transport-layer packet handle supplied by NetSim. It may be an internal NetSim object.	Use the NetSim Application Traffic Filter to select UDP traffic.
payload	Writable application-layer byte buffer.	Read using payload[offset], write using payload[offset] = value, and check length using len(payload).
iopath	NetSim-provided simulation I/O path.	Use for logs, configuration files, temporary files, or external attack parameters. Simple payload-only attacks can ignore it.
context	Packet context dictionary supplied to marker-v2 scripts.	Read context.get("src_socket", "") and context.get("dst_socket", ""). Values use ip:port for IPv4, [ip]:port for IPv6, or an IP-only value when no TCP/UDP port is available.

9.3.4 Required Return Rule

A marker-v2 function should return a dictionary containing modified, drop, and extra_delay_ms. modified and drop are Boolean values. extra_delay_ms is a non-negative delay in milliseconds. Omitted keys use their no-action defaults. Existing six-argument scripts may continue to return True, False, or None. NetSim also compares payload bytes before and after each callback, so an actual payload change is treated as modified even when the reported modified value is False. Setting drop to True terminates processing for that packet; NetSim records the packet as dropped and does not reinject it.

NetSim validates all verdict values. Invalid verdict types cause that callback result to fail open without applying its reported actions. extra_delay_ms must be a finite numeric value excluding Boolean values. Negative or non-finite values are replaced with 0, and values above

60000 milliseconds are capped at 60000 milliseconds (1 min). If a callback raises an exception or returns a malformed verdict, NetSim restores that callback's pre-call payload snapshot, ignores that callback's verdict, and preserves results from earlier successful callbacks.

```
return {
"modified": modified,
"drop": drop,
"extra_delay_ms": extra_delay_ms,
}
```

9.3.5 Safe Editing Guidelines

Always check that payload is not None and len(payload) is sufficient before reading or writing an offset.

The payload and packet handles are valid only during the callback. Do not retain them after the function returns, and overwrite only existing payload bytes without resizing the buffer.

Use protocol documentation or Wireshark to identify the correct byte offset before modifying a field.

Use the NetSim Application Traffic Filter to select the target protocol or port before packets reach the custom modifier.

Use big-endian or little-endian packing based on the protocol field format.

Keep callback processing synchronous, short, and non-blocking. Do not wait for user input, network responses, or long-running external work.

On an exception, write a concise diagnostic to stderr and re-raise the exception so NetSim can restore the payload snapshot and continue with fail-open behavior.

9.3.6 Example Patterns

```
import struct
import sys

NETSIM_PACKET_API_VERSION = 2

def modify_payload(ipv4, ipv6, tcp, udp, payload, iopath, context=None):
context = context or {}
modified = False
drop = False
extra_delay_ms = 0.0

try:
if payload is not None and len(payload) > 0:
mv = memoryview(payload)
if not mv.readonly:
# Pattern 1: modify a byte after confirming the offset.
# if len(mv) > OFFSET:
# mv[OFFSET] = NEW_BYTE_VALUE
# modified = True
```

```
# Pattern 2: overwrite a two-byte big-endian field.
# if len(mv) >= OFFSET + 2:
# struct.pack_into(">H", mv, OFFSET, NEW_VALUE)
# modified = True

# Pattern 3: packet-type-specific drop or delay.
# if tcp is not None:
# drop = True
# if udp is not None:
# extra_delay_ms = 100.0

# Pattern 4: endpoint-specific action.
# src_socket = context.get("src_socket", "")
# dst_socket = context.get("dst_socket", "")
except Exception as exc:
print(f"Custom payload modifier error: {exc}", file=sys.stderr)
raise

return {
"modified": modified,
"drop": drop,
"extra_delay_ms": extra_delay_ms,
}
```

9.3.7 Where This Fits in the Workflow

In the NetSim Cyber GUI, the user selects Import Custom Attacker in the Select Attack Protocol window and browses to a compatible Python payload modifier. For a marker-v2 script, NetSim passes ipv4, ipv6, tcp, udp, payload, iopath, and context to the custom function. The custom modifier is then applied by the Threat Agent to the traffic selected by the Application Traffic Filter during simulation.

The extra_delay_ms verdict belongs to the Packet Modifier callback (modify_payload()). It does not replace or combine with the separate DoS Custom Delay callback (get_delay()), because Packet Modifier and DoS are separate attack selections.

10 Featured Examples

The following end-to-end examples show how NetSim Cyber can be used to observe and modify live power-system communication. Each walkthrough combines a representative plant or data source, a receiving application or controller, and a Threat Agent placed in the communication path.

The IEEE C37.118 example focuses on synchrophasor traffic between a PMU and a PDC. The IEEE 9-Bus integration example shows how measurement and control traffic from a MATLAB/Simulink power-system model can be exercised over TCP, UDP, and Modbus TCP.

Copyright TETCOS LLP

10.1 IEEE C37.118 Synchrophasor Protocol

This section demonstrates the IEEE C37.118-2005 Synchrophasor workflow using the built-in PMU simulator, built-in PDC simulator or OpenPDC destination, and a NetSim Cyber Threat Agent placed in the communication path. NetSim Cyber also supports IEEE C37.118.2-2011 using the corresponding 2011 PMU/PDC simulators and attacker profile; however, the step-by-step walkthrough and results shown in this section use the 2005 profile.

10.1.1 IEEE C37.118 Standard Background

IEEE C37.118 defines communication and measurement requirements for synchrophasor systems used in wide-area monitoring and control. A PMU publishes time-synchronized phasor measurements, and a PDC receives, aligns, validates, and visualizes the stream.

The featured experiment uses IEEE C37.118-2005. NetSim Cyber also supports IEEE C37.118.2-2011, where the measurement and communication requirements are separated more clearly. For both profiles, the first configuration frame is critical because it describes the stream layout, phasor names, data fields, scaling information, and PMU/PDC identifiers required by the attacker UI.

Data frames carry the live phasor, frequency, ROCOF, analog, and digital measurements.

Configuration frames describe how the data frames must be parsed.

Header frames carry human-readable station or stream information.

Command frames are used by the PDC to control PMU data transmission.

10.1.2 Supported Synchrophasor Profiles in NetSim Cyber

Profile	Source Simulator	Destination	Attack Support
IEEE C37.118-2005	PMU-2005.pyw	PDC-2005.pyw or OpenPDC	2005 Synchrophasor attacker
IEEE C37.118.2-2011	PMU-2011.pyw	PDC-2011.pyw or OpenPDC	2011 Synchrophasor attacker

The remaining procedure uses the IEEE C37.118-2005 row from this table. For 2011 validation, select the 2011 PMU, 2011 PDC or OpenPDC configuration, and the 2011 Synchrophasor attacker profile.

10.1.3 OpenPDC Reference

OpenPDC setup steps are not repeated in this example. Refer to the official OpenPDC documentation and installer package when OpenPDC is used as the destination PDC.

OpenPDC project and documentation: `https://github.com/GridProtectionAlliance/openPDC`

OpenPDC Windows installer package: https://github.com/GridProtectionAlliance/openPDC/releases/download/v2.9.148/Synchrophasor.Installs.zip

After extraction, run openPDCSetup.msi to install OpenPDC Manager.

10.1.4 NetSim Cyber Architecture

The featured setup uses a distributed three-system arrangement. System 1 generates PMU traffic, System 2 runs NetSim Cyber and the Threat Agent, and System 3 receives the stream using a PDC or OpenPDC. NetSim Cyber bridges the live traffic path and applies protocol-aware payload manipulation or timing attacks.

System 1: PMU source simulator generating IEEE C37.118 traffic.

System 2: NetSim Cyber host with Real Nodes, network path, application filter, and Threat Agent.

System 3: PDC destination using the built-in PDC viewer or OpenPDC.

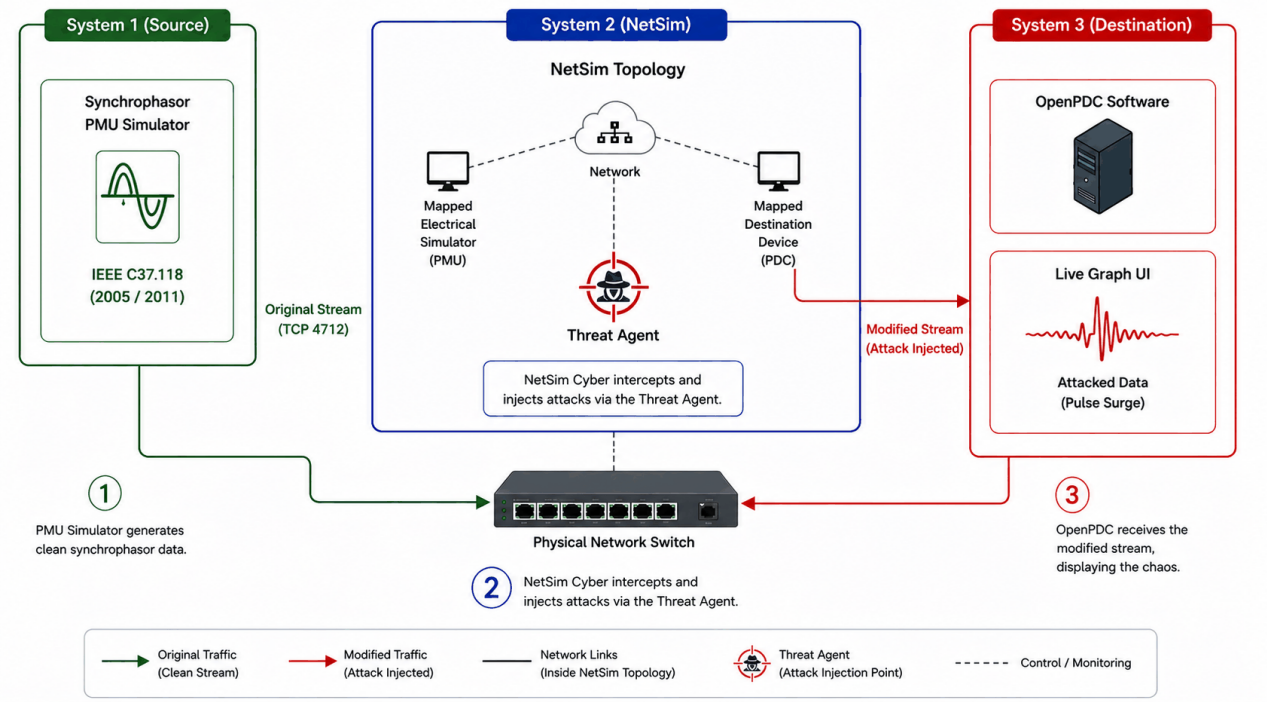


Figure 10-1: *Distributed Synchrophasor experiment architecture with PMU source, NetSim Cyber Threat Agent, and PDC destination.*

10.1.5 Important Synchrophasor Run Order

For Synchrophasor experiments, NetSim Cyber must be running and the Synchrophasor attacker must be ready before PMU-PDC traffic starts. The attacker learns the stream layout from the first configuration frame. If the PMU and PDC complete their initial handshake before NetSim Cyber is ready, the phasor list may not load correctly.

10.1.6 Demonstrated Attack Modes in This Example

In this example, we show Increment Bias, Decrement Bias, Ramp/Drift, Noise/Jitter, Frequency Override, and Manual Injection. These modes are selected because their effect can be observed directly at the PDC destination without adding extra setup steps.

10.1.7 Experimental Procedure

The sequence is important for Synchrophasor experiments because the attacker must capture the initial configuration frame before PMU data transmission begins.

10.1.7.1 Prepare the Distributed Systems

Step 1: Connect all systems to the same LAN and verify reachability using ping.

Step 2: Launch NetSim Cyber on System 2.

Step 3: Run NetSimCyberClient.exe as Administrator on both the source and destination systems when using the distributed workflow.

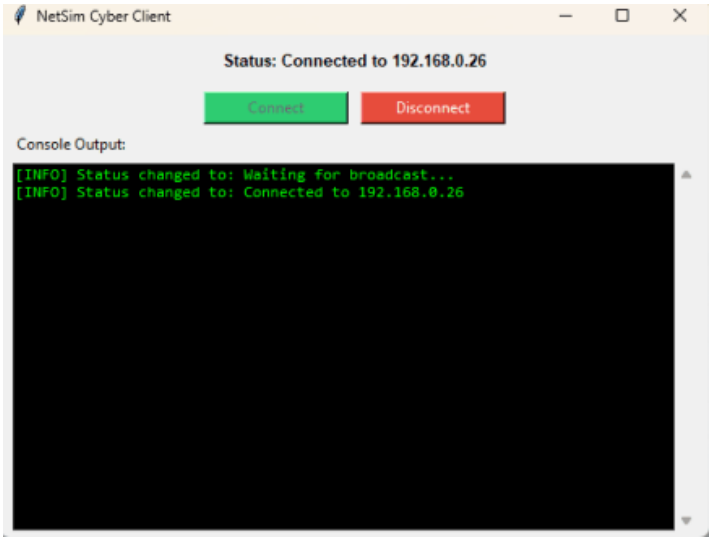


Figure 10-2: *NetSimCyberClient.exe connected to the NetSim Cyber host. Run this client on both the source and destination systems so NetSim Cyber can configure the communication gateway automatically.*

Note: If required, the same communication path can also be prepared manually by configuring static routes on the endpoint systems so traffic is forwarded through the NetSim Cyber system.

Step 4: Confirm that the source and destination Real Nodes appear in the NetSim Cyber topology.

Copyright TETCOS LLP

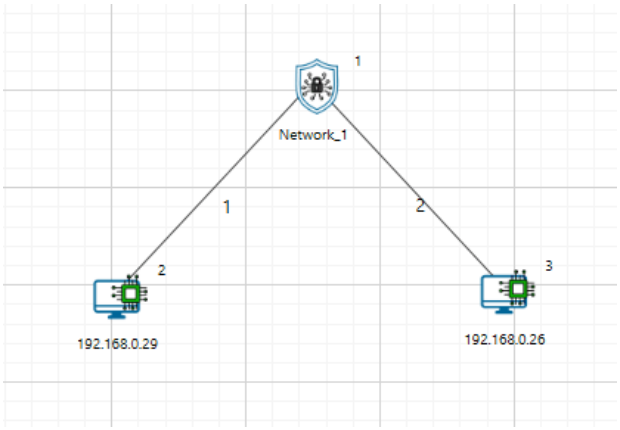


Figure 10-3: *NetSim Cyber topology showing the PMU source and PDC destination Real Nodes connected through the network path.*

10.1.7.2 Configure NetSim Cyber

Step 1: Place the Threat Agent in the communication path between the source and destination.

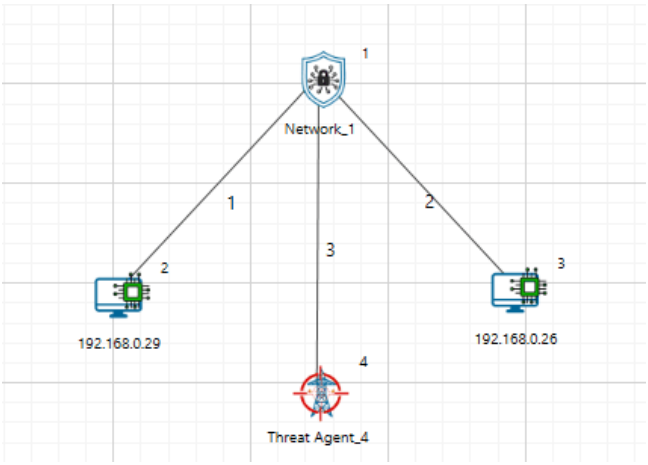


Figure 10-4: *Threat Agent placed in the communication path between the PMU source and PDC destination.*

Step 2: Configure the Application Traffic Filter from the top ribbon by selecting the PMU source, PDC destination, and TCP port used by the Synchrophasor traffic.

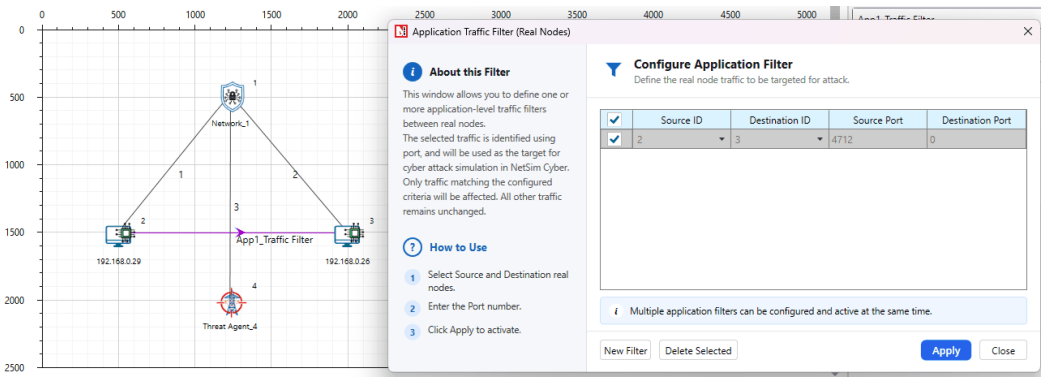


Figure 10-5: *Application Traffic Filter configured for the PMU-to-PDC traffic stream and protocol TCP port.*

Step 3: Select the Synchrophasor attacker and choose the IEEE C37.118-2005 profile for this example.

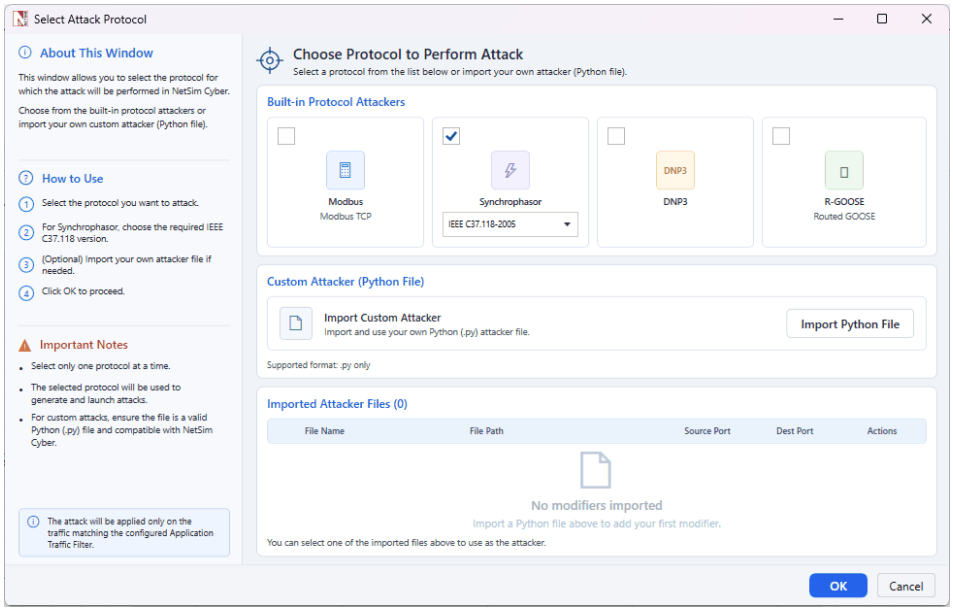


Figure 10-6: *Built-in Synchrophasor attacker selection. The 2005 profile is used for this walkthrough.*

Step 4: Run the NetSim Cyber simulation and keep the attacker dashboard ready.



Figure 10-7: *IEEE C37.118-2005 attacker dashboard showing automatic, manual, and replay attack modes.*

10.1.7.3 Start PMU-PDC Traffic and Apply Attack

Important: keep NetSim Cyber and the IEEE C37.118-2005 attacker dashboard ready before PMU-PDC communication starts, because the attacker must capture the first configuration frame.

Step 1: Open and configure the IEEE C37.118-2005 PMU simulator, but do not start PMU

traffic yet.

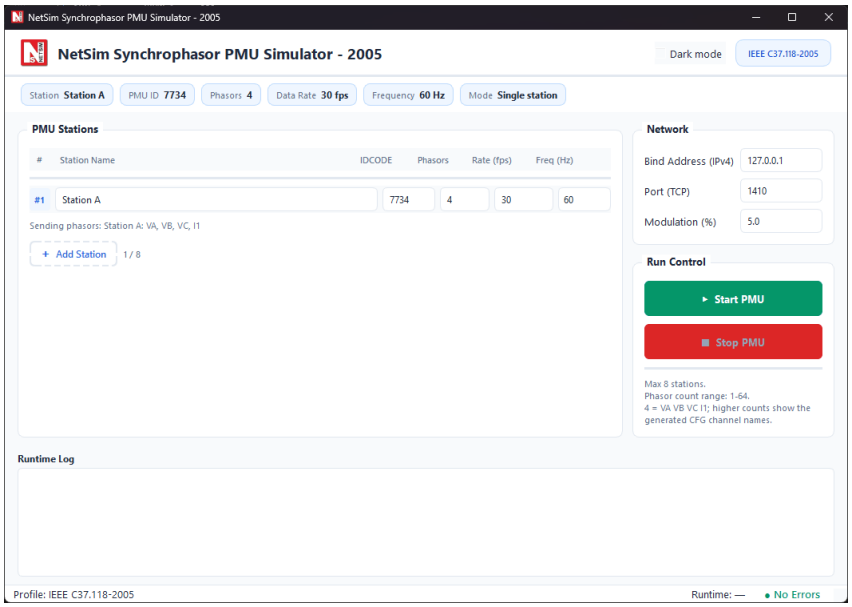


Figure 10-8: *PMU simulator configuration screen showing station, PMU ID, port, and run controls. For this walkthrough, use the IEEE C37.118-2005 PMU simulator.*

Step 2: Start the PMU simulator only after the NetSim Cyber simulation and attacker dashboard are ready, then connect the built-in PDC simulator or OpenPDC Manager to the PMU stream.

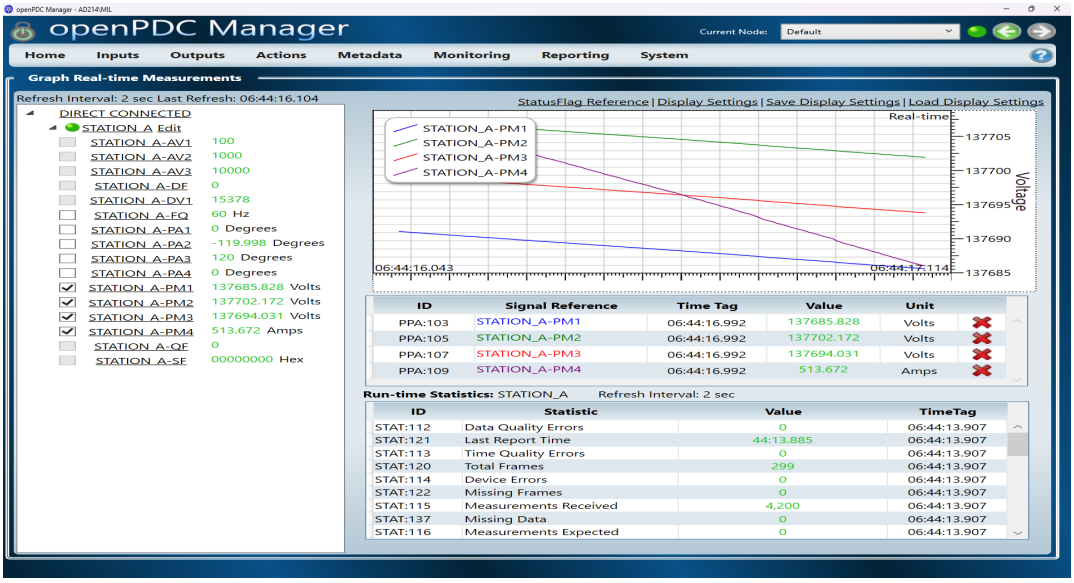


Figure 10-9: *PDC or OpenPDC destination receiving and visualizing the live Synchrophasor measurement stream from the PMU.*

Step 3: Verify that live measurements are displayed at the PDC and that the attacker UI has loaded the stream and phasor mapping from the first configuration frame.

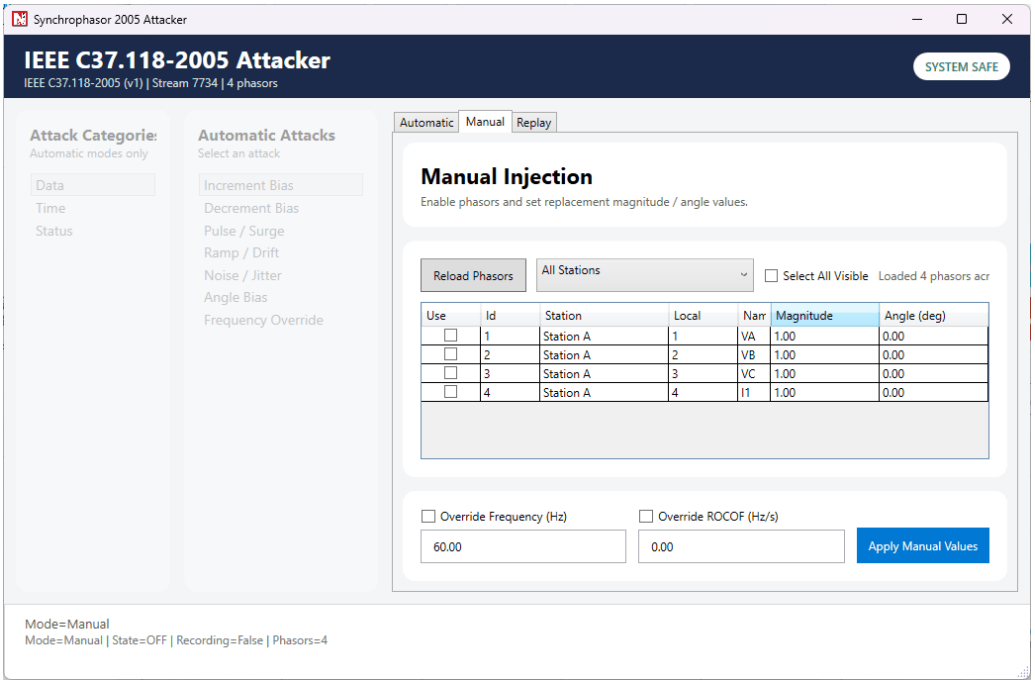


Figure 10-10: *Manual injection view used after the phasor list is loaded from the first configuration frame.*

Step 4: Apply the selected automatic attack or manual injection and observe the changed values at the PDC destination.

10.1.8 Results and Analysis

This section compares the baseline PDC measurements with the measurements observed after applying selected IEEE C37.118-2005 attacks. Each result figure shows the effect visible at the PDC or in the attacker interface while the PMU-PDC communication stream remains active.

10.1.8.1 Baseline Operation

During normal operation, the PDC receives a stable Synchrophasor stream from the PMU simulator. The baseline view is used as the reference for comparing the attack results.

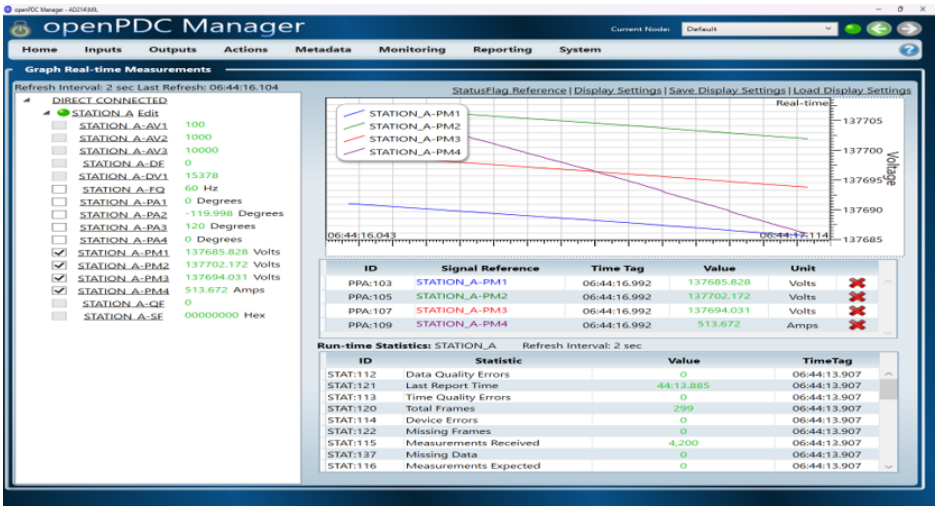


Figure 10-11: *Baseline PDC measurements before attack activation.*

10.1.8.2 Increment Bias Attack

The Increment Bias attack adds a positive offset to selected phasor magnitude values. In the result, the selected PMU measurements increase at the PDC while the Synchrophasor stream remains connected.

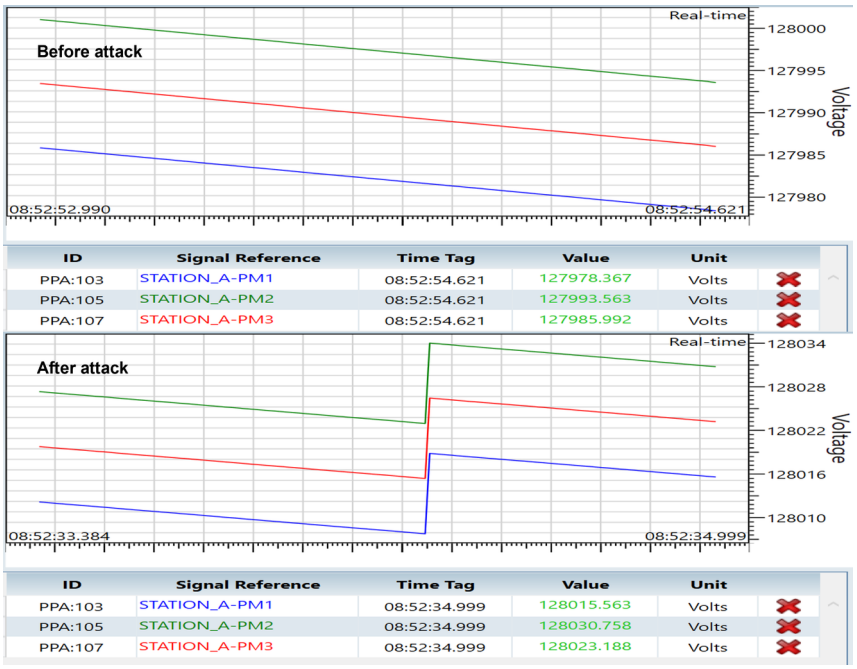


Figure 10-12: *PDC measurements before and after applying the Increment Bias attack.*

10.1.8.3 Decrement Bias Attack

The Decrement Bias attack subtracts an offset from selected phasor magnitude values. In the result, the selected PMU measurements decrease at the PDC without interrupting the communication stream.

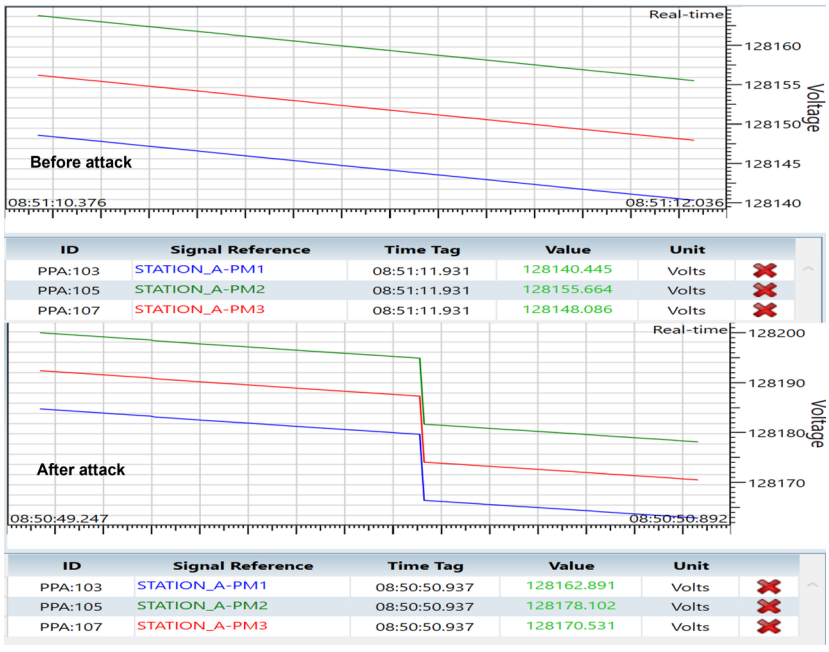


Figure 10-13: *PDC measurements before and after applying the Decrement Bias attack.*

10.1.8.4 Ramp / Drift Attack

The Ramp / Drift attack gradually changes selected phasor magnitude values over time. The result shows a progressive change in the plotted measurements, resembling slow sensor drift or load variation while the protocol stream remains valid.

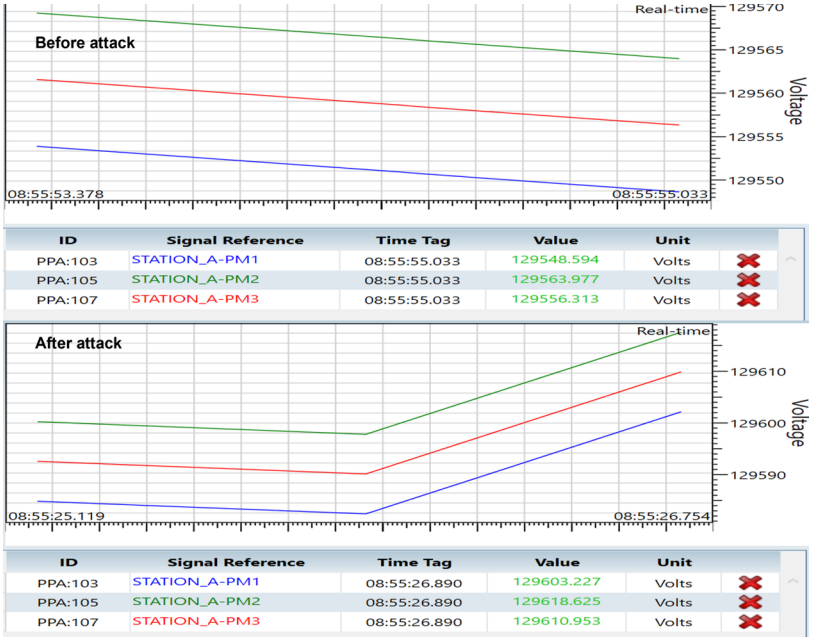


Figure 10-14: *PDC measurements before and after applying the Ramp / Drift attack.*

10.1.8.5 Noise / Jitter Attack

The Noise / Jitter attack adds random variation to selected phasor magnitude values while keeping the communication stream active. The result shows unstable measurement traces at the PDC, which can be used to evaluate measurement-quality monitoring and anomaly-detection logic.



Figure 10-15: *PDC measurements before and after applying the Noise / Jitter attack.*

10.1.8.6 Frequency Override Attack

The Frequency Override attack replaces the frequency value in the Synchrophasor data frame. The result shows the PDC frequency measurement changing from the baseline value to the injected value while the stream remains connected.

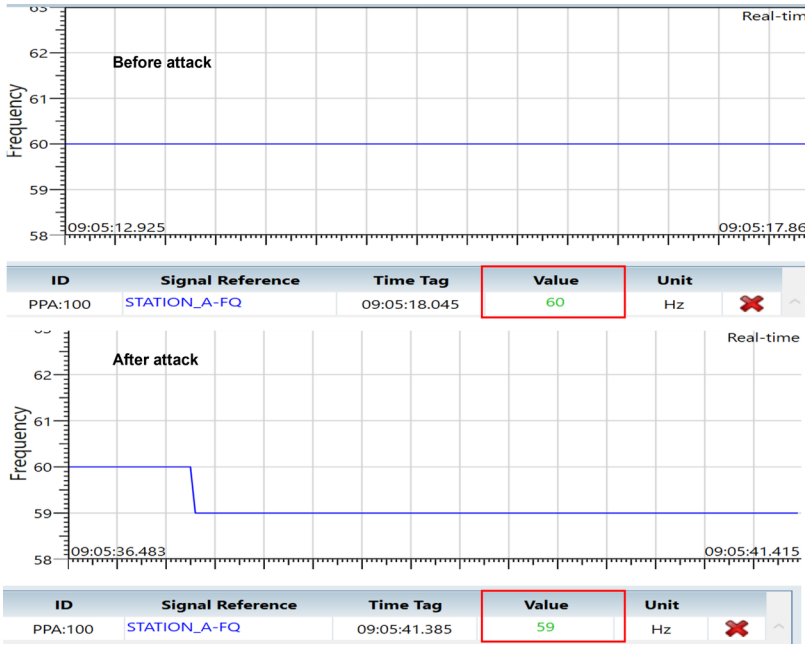


Figure 10-16: *PDC frequency measurement before and after applying the Frequency Override attack.*

10.1.8.7 Manual Injection

Manual Injection allows selected phasor magnitude, angle, frequency, or ROCOF values to be entered directly after the phasor list is loaded from the first configuration frame. The result shows user-defined measurement values appearing at the PDC for the selected fields.

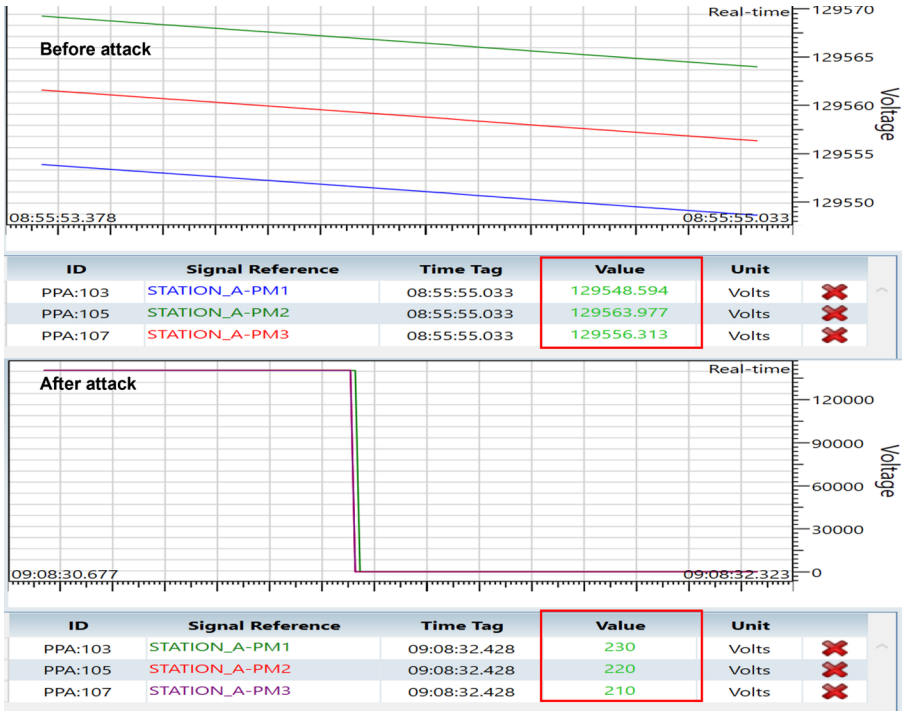


Figure 10-17: *PDC measurements before and after manual injection of selected Synchrophasor values.*

10.1.9 Key Observations

- NetSim Cyber can intercept and modify live IEEE C37.118 traffic between PMU and PDC endpoints.
- This walkthrough uses the IEEE C37.118-2005 profile; the 2011 profile is supported as a separate variant.
- The first configuration frame must be captured so the attacker UI can load the stream layout and phasor list.
- The example focuses on increment, decrement, ramp/drift, noise/jitter, frequency override, and manual injection effects at the PDC destination.
- OpenPDC can be used as the PDC destination, but detailed OpenPDC setup is referenced externally instead of repeated in this example.

10.2 IEEE 9-Bus Integration

This example demonstrates TCP, UDP, and Modbus TCP communication between an IEEE 9-Bus MATLAB/Simulink plant, a supervisory controller, and NetSim Cyber.

10.2.1 Concept and Project Objective

The IEEE 9-bus model represents the electrical plant in MATLAB/Simulink. The example monitors the Bus 6 voltage and the Bus 6 circuit-breaker state. A MATLAB supervisory controller receives these measurements and returns a binary command to the breaker.

NetSim Cyber is inserted in the measurement communication path. In this document, a baseline run means that the NetSim Cyber attacker is not started, so the supervisory controller receives

the unmodified voltage produced by Simulink. During an attack run, the configured NetSim Cyber attacker changes the Bus 6 voltage before the supervisory controller processes it. The supervisory controller therefore acts on the received network value rather than the original plant value.

Threat model. The Threat Agent is positioned as a man-in-the-middle on the Simulink-to-supervisory-controller measurement channel. It changes selected application-payload values while leaving the surrounding network exchange and unaffected fields usable. Detection is difficult when packets remain syntactically valid and arrive with plausible addressing, sequence and timing information; measurement plausibility and cross-signal checks are therefore important.

Binary breaker convention. Command 0 closes the breaker and permits current flow through the connected circuit. Command 1 opens the breaker and disconnects that circuit.

10.2.2 Study objectives

- Compare normal closed-loop control with configured measurement manipulation in the network path.
- Observe the voltage received by the supervisory controller and the breaker command returned to Simulink.
- Inspect original and modified packets in NetSim Cyber and Wireshark.
- Run the same use case with TCP, UDP and Modbus TCP communication.

10.2.3 System Architecture and Control Flow

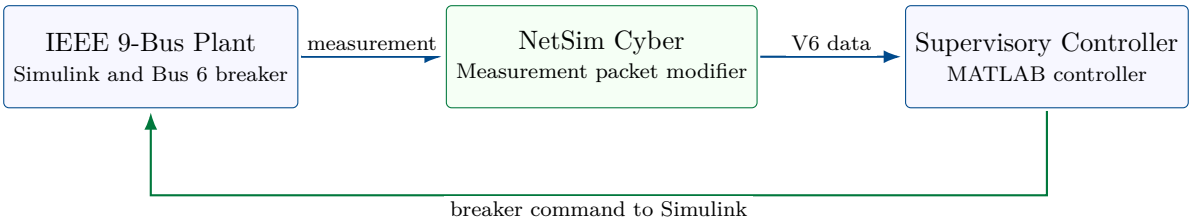


Figure 10-18: *Closed-loop architecture used by the three communication examples.*

10.2.3.1 Roles of the components

Component	Role	Data handled
IEEE 9-bus Simulink plant	Represents the electrical network and Bus 6 breaker.	Bus 6 voltage, breaker state and returned command
NetSim Cyber	Models the communication path and executes the selected attacker.	Custom frames on port 5001 or Modbus responses from port 502
Supervisory controller	Applies voltage-limit logic and creates a breaker command.	Received Bus 6 voltage and returned command
Wireshark	Decodes PRE, MOD and POST packet captures.	TCP, UDP or Modbus TCP packets

10.2.3.2 Source and destination

For TCP and UDP, starting the supervisory controller first does not make it the measurement source. It opens or listens on the required port and waits, while Simulink creates the Bus 6 measurement frame. In the Modbus example, the controller initiates an FC04 request and Simulink returns the requested plant measurement.

Path	Source	Destination	Purpose
Measurement	Simulink	Supervisory controller, through NetSim Cyber	Deliver Bus 6 voltage and breaker state; for Modbus this is the FC04 response from source port 502
Command	Supervisory controller	Simulink	Apply the selected breaker command; for Modbus this is an FC05 request to destination port 502

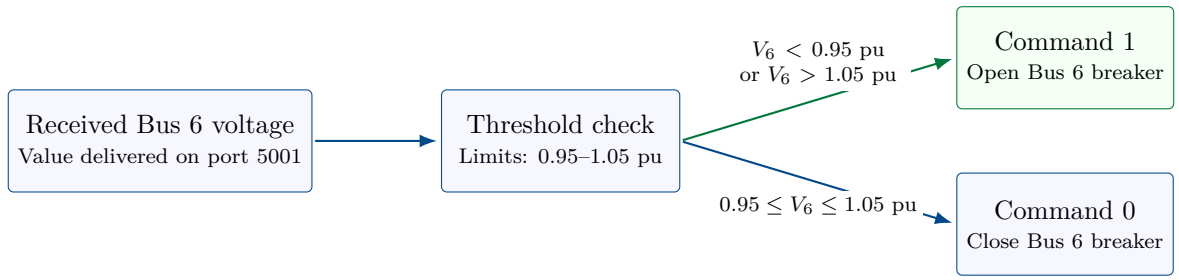


Figure 10-19: *Example supervisory-controller decision logic.*

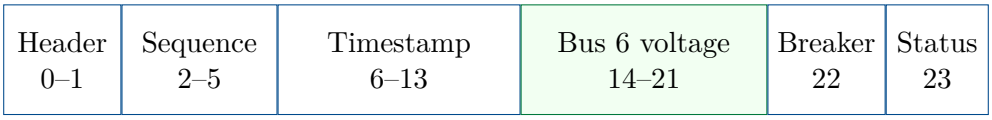
Example thresholds. The 0.95 pu and 1.05 pu limits are experiment settings for this example. They are not protection settings for an operating power system.

10.2.4 Data and Command Formats

The TCP and UDP variants use the same custom application payloads. Multi-byte fields are encoded in little-endian order. UDP carries each frame in a datagram, while TCP carries the same fields in an ordered byte stream. The Modbus TCP example uses FC04 registers, FC02 discrete inputs and an FC05 coil instead of these custom frames.

Copyright TETCOS LLP

Measurement frame – 24 bytes



Command frame – 18 bytes



Figure 10-20: *Measurement and command payload layouts. Green fields are used by the packet-modification and breaker-control logic.*

10.2.4.1 24-byte measurement frame

Bytes	Field	Type	Size	Description
0–1	Header	uint16	2 bytes	Measurement-frame identifier
2–5	Sequence	uint32	4 bytes	Monotonic frame number
6–13	Timestamp	double	8 bytes	Simulink simulation time in seconds
14–21	Bus 6 voltage	double	8 bytes	Voltage magnitude (pu)
22	Breaker state	uint8	1 byte	0 closed; 1 open
23	Status	uint8	1 byte	Frame status

10.2.4.2 18-byte command frame

Bytes	Field	Type	Size	Description
0–1	Header	uint16	2 bytes	Command-frame identifier
2–5	Sequence	uint32	4 bytes	Command sequence number
6–13	Timestamp	double	8 bytes	Controller time associated with the decision
14–15	Target ID	uint16	2 bytes	Bus 6 breaker target
16	Command	uint8	1 byte	0 closed; 1 open
17	Status	uint8	1 byte	Command status

Per unit. pu means per unit. A value of 1.00 pu represents the selected nominal voltage base; 0.95 pu represents 95 percent of that base.

10.2.5 Simulink Integration

10.2.5.1 Measurement interface

The measurement subsystem samples the Bus 6 voltage and breaker state, reads the simulation timestamp, packs the values into a 24-byte frame, and passes the frame to the UDP Send or

TCP/IP Send block. The sample period is 0.05 s.

10.2.5.2 Command interface

The command subsystem receives an 18-byte frame, extracts the command byte, validates its data type, and applies the value to the Bus 6 breaker. A hold block preserves the last received command between 50 ms packets. A fallback command is used before the first valid command arrives.

10.2.5.3 Observable signals

- Controller plot: Bus 6 voltage received after the network path.
- Controller plot: breaker command sent back to Simulink.
- Simulink scope: selected breaker command.
- Simulink results: bus voltages, generator terminal voltages and rotor angles.

10.2.6 TCP Example

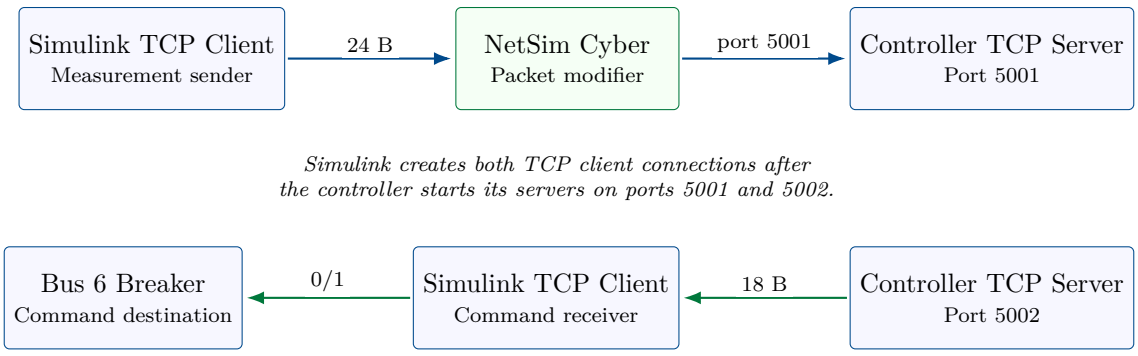
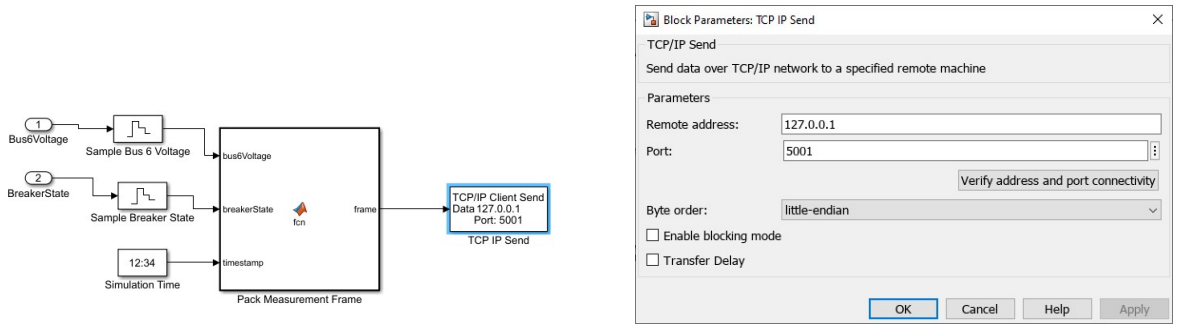


Figure 10-21: *TCP measurement and command connections with both client-server paths shown.*

10.2.6.1 TCP measurement connection



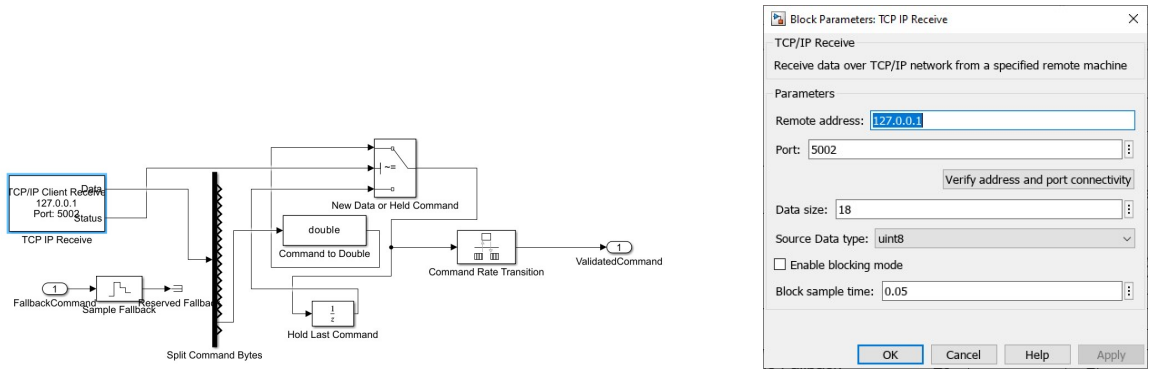
(a) Simulink TCP measurement subsystem.

(b) TCP/IP Send: controller server 127.0.0.1:5001.

Figure 10-22: *TCP measurement interface and block parameters.*

The supervisory controller is started first and creates a TCP server on port 5001. When the Simulink model starts, its TCP/IP Send block connects as a client and transmits the 24-byte measurement stream.

10.2.6.2 TCP command connection



(a) Simulink TCP command subsystem. *(b) TCP/IP Receive: controller server 127.0.0.1:5002.*

Figure 10-23: *TCP command interface and block parameters.*

The supervisory controller also creates a TCP server on port 5002. The Simulink TCP/IP Receive block connects as a client and receives the 18-byte command stream. The operating system assigns the TCP client's ephemeral source port; port 5002 is the fixed command-server port.

Connection	TCP client	TCP server	Frame
Measurement	Simulink TCP/IP Send	Controller port 5001	24 bytes
Command	Simulink TCP/IP Receive	Controller port 5002	18 bytes

TCP framing. TCP is a byte stream. The receiver reconstructs application frames using the fixed 24-byte and 18-byte lengths. Packet captures may show a frame split across TCP segments or multiple frames in one segment.

10.2.7 UDP Example

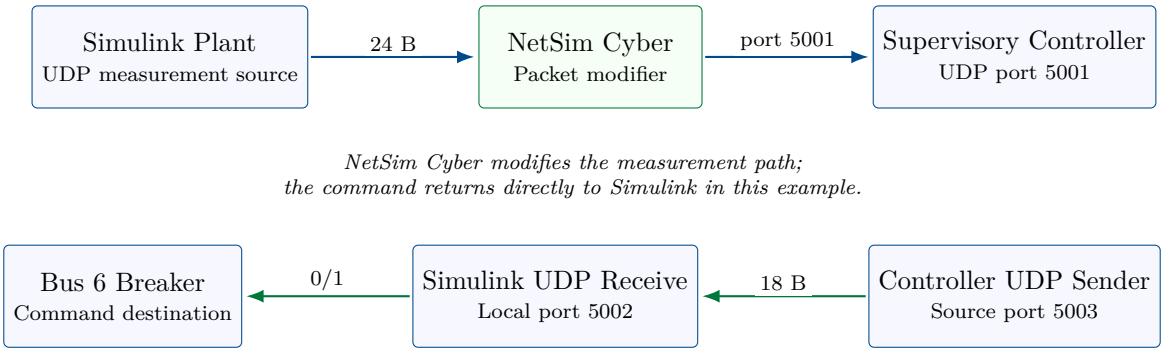
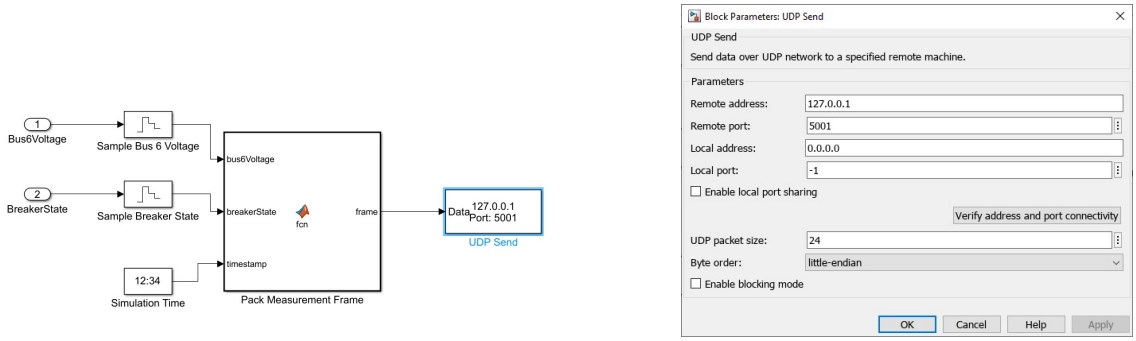


Figure 10-24: *UDP measurement and command paths with complete source and destination endpoints.*

10.2.7.1 UDP measurement path

Simulink creates one 24-byte UDP datagram per measurement sample. The destination port is 5001. NetSim Cyber filters this traffic and may change the Bus 6 voltage field before the datagram reaches the controller.

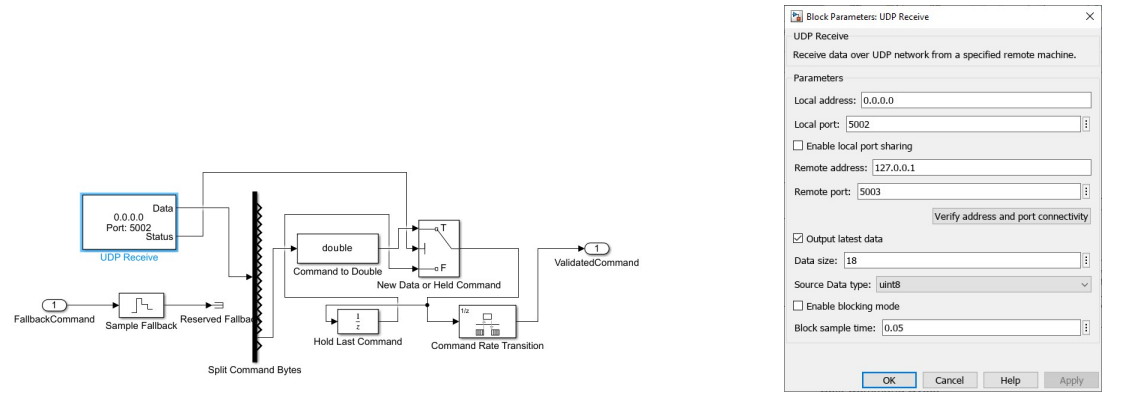


(a) Simulink UDP measurement subsystem.

(b) UDP Send: destination 127.0.0.1:5001 and 24-byte payload.

Figure 10-25: *UDP measurement interface and block parameters.*

10.2.7.2 UDP command path



(a) Simulink UDP command subsystem.

(b) UDP Receive: local port 5002 and controller source port 5003.

Figure 10-26: *UDP command interface and block parameters.*

Direction	Source port	Destination port	Payload
Measurement	Ephemeral/local setting	5001	24 bytes
Command	5003	5002	18 bytes

UDP behavior. UDP does not establish a connection. Each measurement or command is one datagram, so loss, delay, duplication and reordering can be studied at the network layer.

10.2.8 NetSim Cyber Configuration for TCP and UDP

The same NetSim Cyber topology and packet-modifier procedure are used for both examples. The application filter selects measurement traffic sent from the Simulink-side Real Node to the controller-side Real Node on destination port 5001.

10.2.8.1 Create the Real Node topology

On the Grid, add the source and destination nodes and place a Threat Agent between them. The nodes are connected automatically when they are added to the scenario. Verify the scenario

layout before proceeding to the next configuration step.

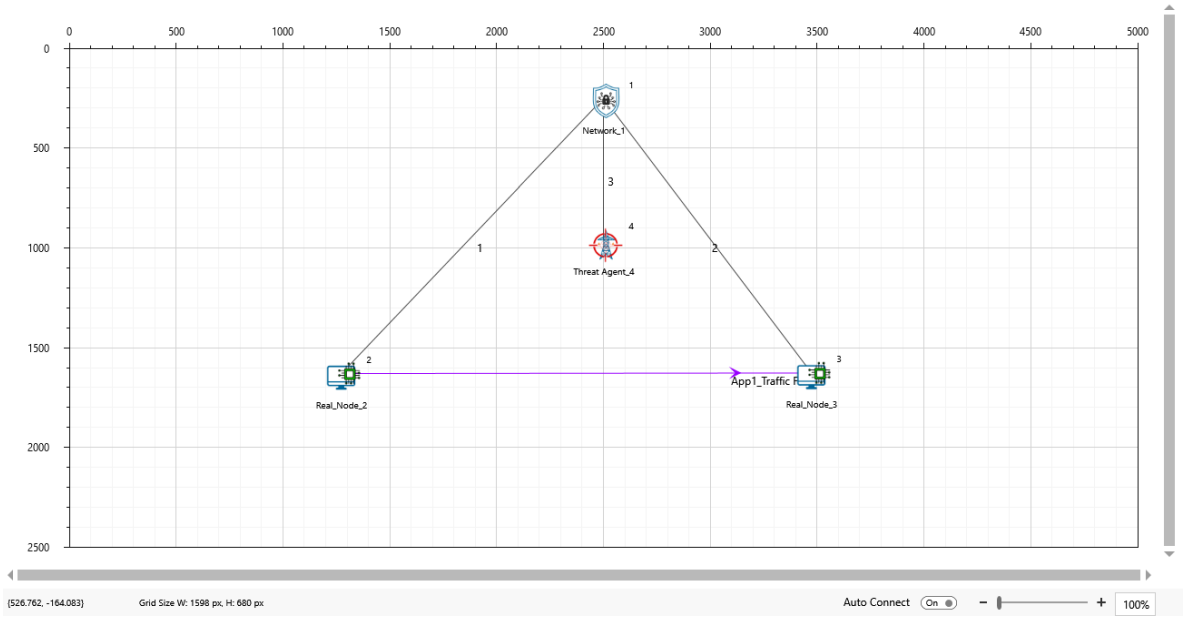


Figure 10-27: *NetSim Cyber topology with two Real Nodes, one network and one Threat Agent.*

10.2.8.2 Configure the application filter

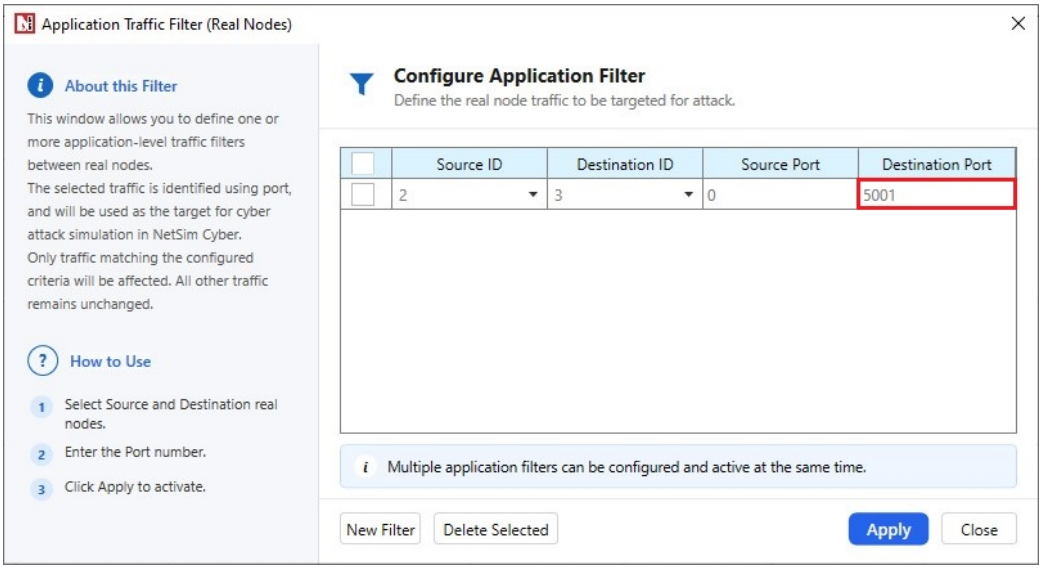
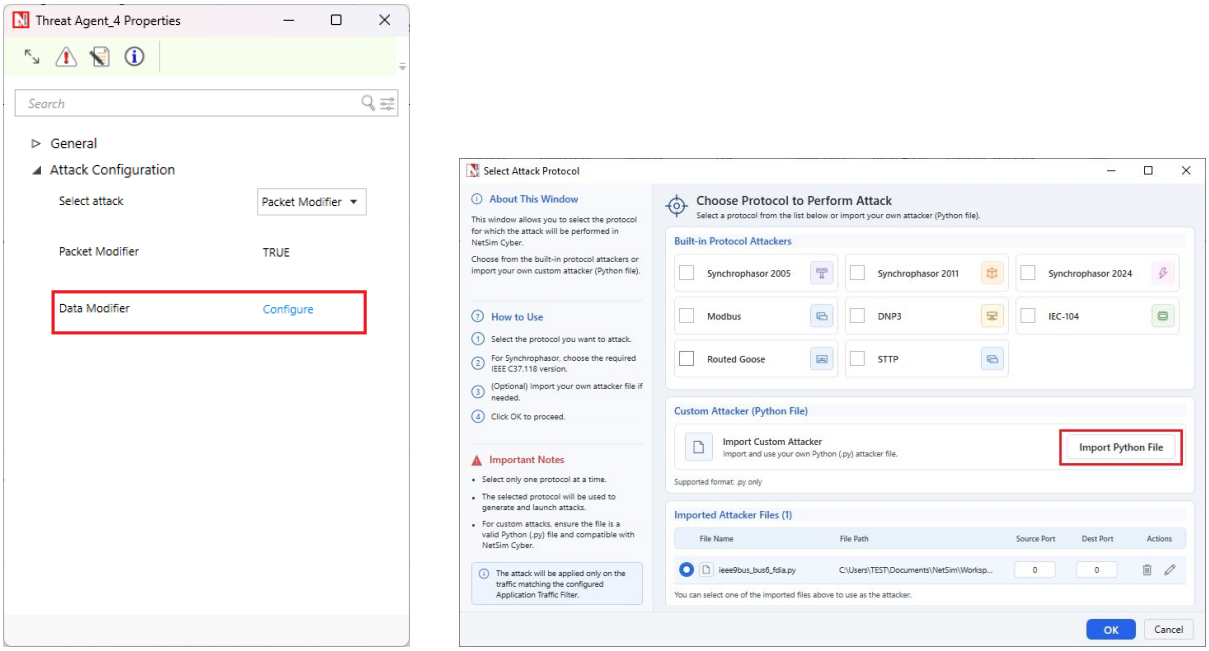


Figure 10-28: *Application filter for source node 2, destination node 3 and destination port 5001.*

Choose the Simulink-side Real Node as Source ID and the controller-side Real Node as Destination ID. Set Destination Port to 5001. Source Port may remain 0 when the sending endpoint uses an automatically selected port.

10.2.8.3 Select Packet Modifier

1. Choose **Packet Modifier** in the Threat Agent properties.
2. Select **Configure** under Data Modifier and import `ieee9bus_bus6_fdia.py`.
3. Match destination port 5001 so the modifier processes the Bus 6 measurement path.



(a) Threat Agent attack type set to Packet Modifier.

(b) Custom Python attacker selected for Bus 6 FDIA.

Figure 10-29: *Threat Agent and custom payload-modifier configuration.*

TCP attacker file location. The packaged TCP attacker script is available in:

```
C:\Program Files\NetSim\Cyber_v15_1\Docs\
Sample_Configuration\Cyber_IEEE 9-Bus_Example\
IEEE 9-Bus with TCP\NetSimAttackerFiles
```

Select `ieee9bus_bus6_fdia.py` from this folder. If a different NetSim edition or version is installed, use the corresponding NetSim installation directory.

The example modifier reads the little-endian double at measurement bytes 14–21 and applies a voltage bias.

FDIA rule. Attacked voltage = original Bus 6 voltage − 0.07 pu. Only the voltage field is changed; the breaker-state byte in the same measurement frame is not overwritten.

10.2.8.4 Enable packet captures

Enable PRE, Modified Packets and POST captures. Enable Dropped Packets capture when testing a drop scenario. Select an output directory with enough space for the expected traffic volume.

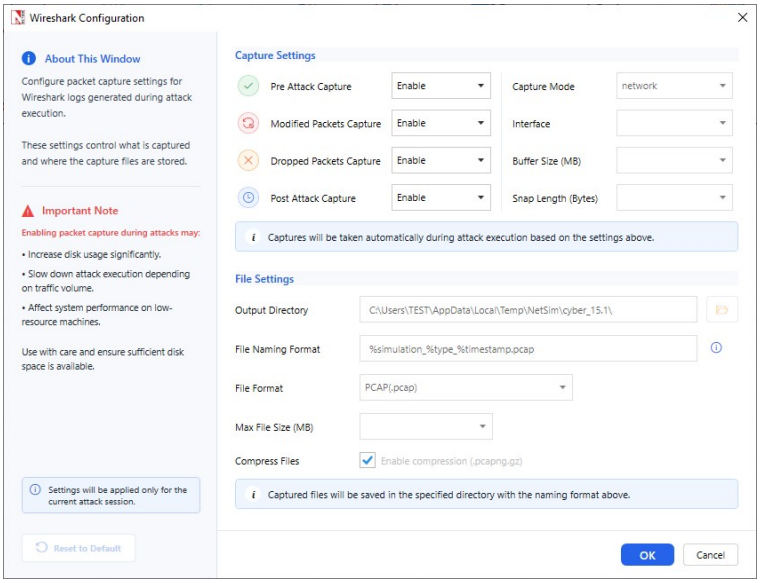


Figure 10-30: *Wireshark capture settings for PRE, MOD, DROP and POST files.*

10.2.8.5 Run and open the results

After the run, use the Results view to confirm the application filter, packet counts and generated capture files. Open `PRE.pcap` and `MOD.pcap` in Wireshark to compare the original and attacked Bus 6 voltage.

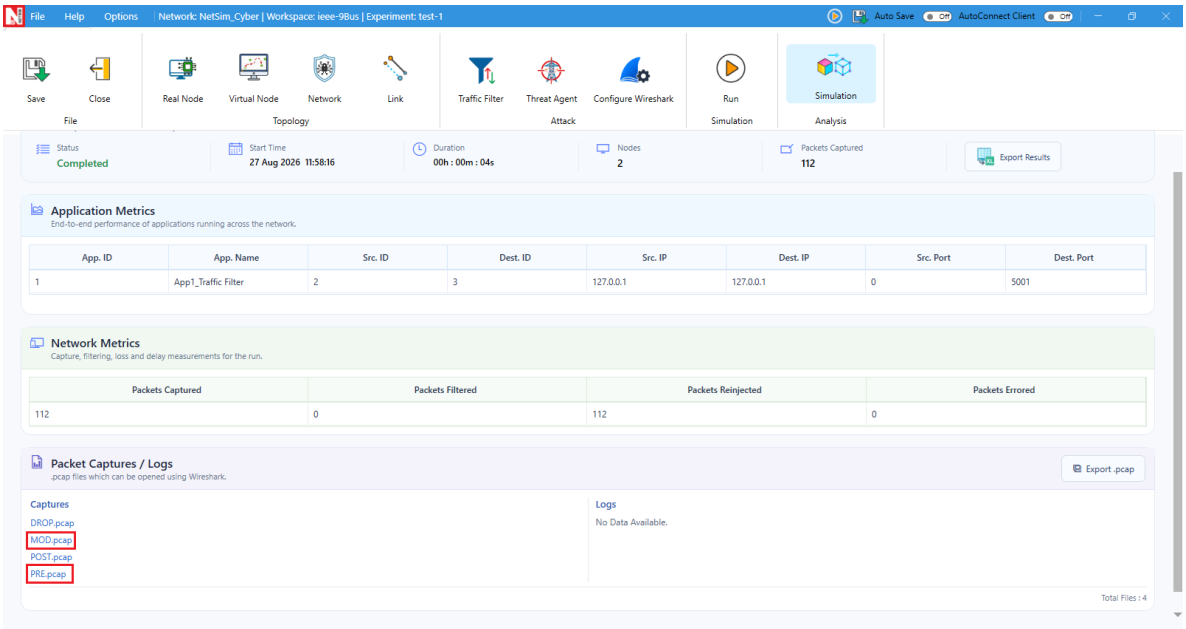


Figure 10-31: *NetSim Cyber results with application metrics and PRE, MOD and POST packet captures.*

10.2.9 TCP and UDP Baseline and Attack Results

Each protocol is run twice with the same Simulink stop time, sample period and supervisory-controller limits. For the baseline, the same communication path is used but the NetSim Cyber attacker is not started, so no payload modification is applied. For the attack run, the configured attacker is started. The plots show the voltage processed by the supervisory controller and the command returned to Simulink.

10.2.9.1 TCP results

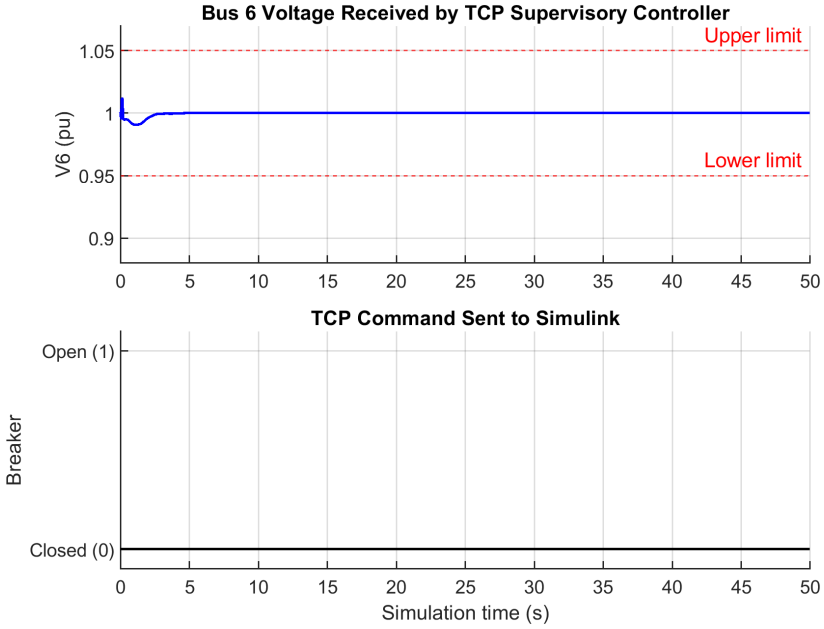


Figure 10-32: *TCP baseline: received Bus 6 voltage remains inside the example limits and the breaker command remains closed (0).*

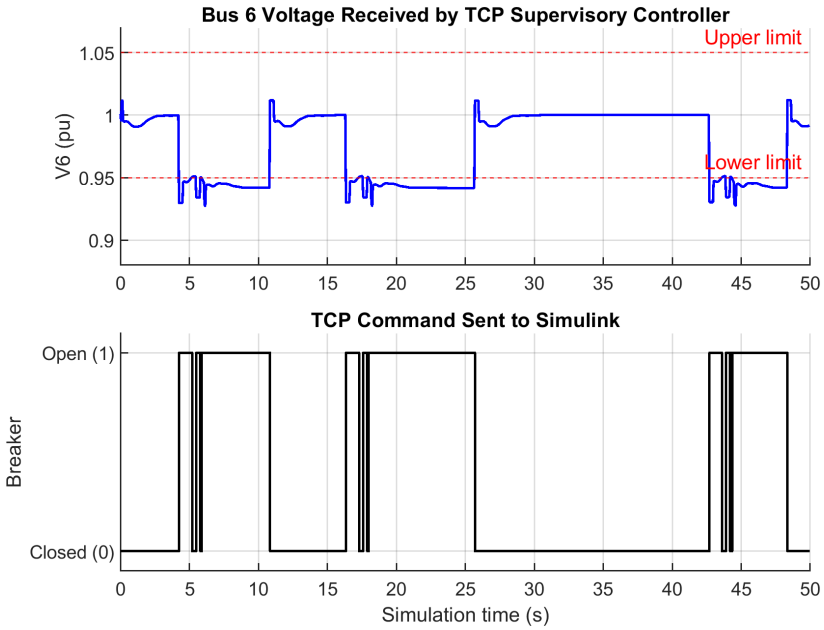


Figure 10-33: *TCP with NetSim Cyber: the received voltage falls below 0.95 pu and the controller sends open command (1).*

10.2.9.2 UDP results

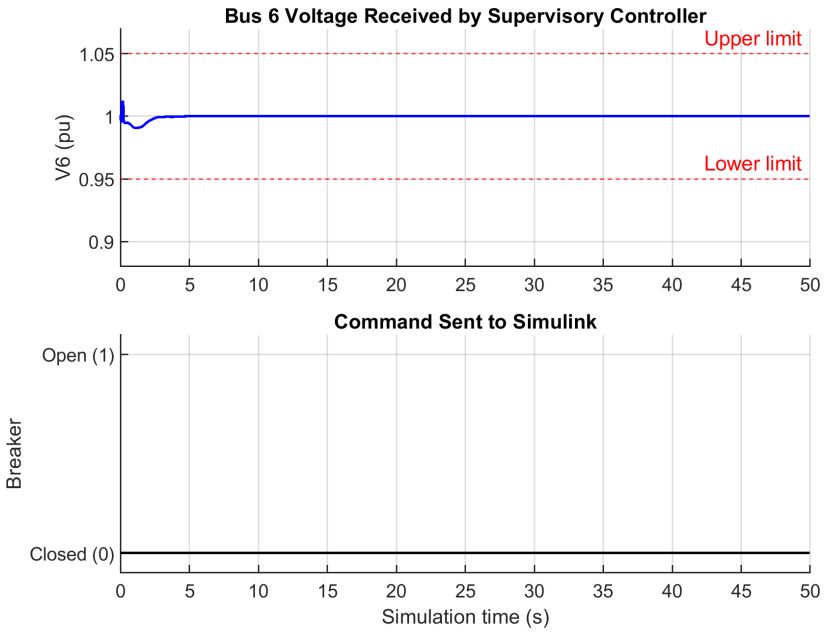


Figure 10-34: *UDP baseline: received Bus 6 voltage remains inside the example limits and the breaker command remains closed (0).*

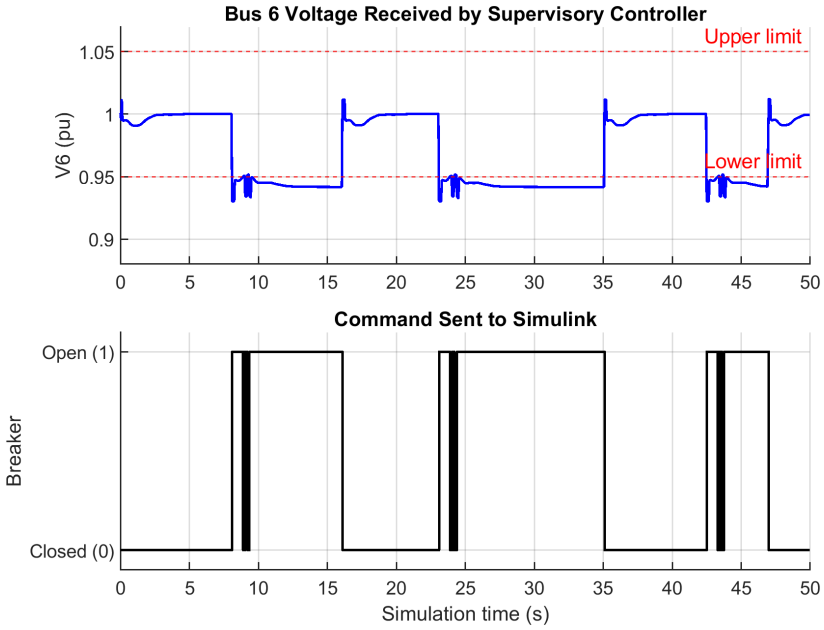


Figure 10-35: *UDP with NetSim Cyber: the biased voltage crosses the lower limit and the controller issues open command (1) during the affected intervals.*

10.2.9.3 Interpretation

Observation	Without attack	With NetSim Cyber FDIA
Voltage used by controller	Original Bus 6 voltage	Original voltage minus configured bias
Threshold decision	Normally stays within 0.95–1.05 pu	May cross the 0.95 pu lower limit
Breaker command	0: closed	1: open while received V6 is below the limit
Physical model response	Normal Bus 6 and generator behavior	Breaker operation can change later electrical samples

10.2.10 Modbus TCP Example

The Modbus TCP example retains the IEEE 9-bus plant, Bus 6 measurement and voltage-limit logic used in the TCP and UDP examples. The communication representation changes from custom frames to Modbus functions. Simulink hosts a Modbus TCP server on port 502, and the MATLAB supervisory controller operates as the Modbus client. FC04 requests and FC05 commands travel from the supervisory controller to Simulink with destination port 502; FC04 measurement responses travel from Simulink to the supervisory controller with source port 502 and pass through the NetSim Cyber response filter.

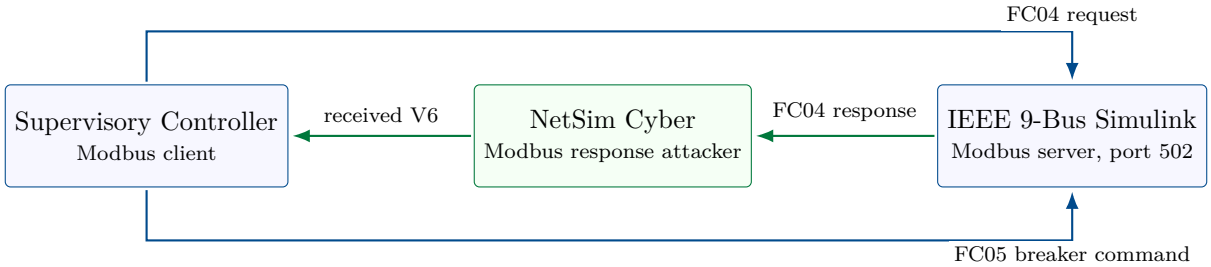


Figure 10-36: *Modbus TCP measurement request, attacked response and breaker-command path.*

10.2.10.1 Simulink Modbus interface

The `ModbusLevel2Server` block is implemented by `modbusPlantServer.m`. It publishes the Bus 6 voltage and simulation timestamp, reports the breaker state, receives coil writes and forwards the validated command to the Bus 6 breaker. The command value is held between Modbus requests.

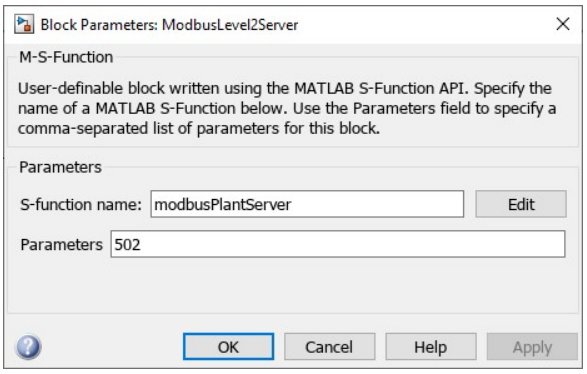


Figure 10-37: *Level-2 MATLAB S-Function configured with* `modbusPlantServer` *and TCP port 502.*

10.2.10.2 Register and coil mapping

Modbus operation	Address	Encoding	Purpose
FC04 input registers	0–1	FLOAT32, ABCD	Bus 6 voltage in pu
FC04 input registers	2–3	FLOAT32, ABCD	Simulink timestamp in seconds
FC02 discrete input	0	Boolean	Bus 6 breaker state
FC05 single coil	0	OFF/ON	Breaker command: 0 closed, 1 open

The controller opens the breaker when the received voltage is below 0.95 pu or above 1.05 pu. Values inside that range produce command 0 and keep the breaker closed.

10.2.10.3 Baseline result without NetSim Cyber

During the baseline run, the FC04 voltage received by the supervisory controller remains near 1.0 pu. It stays inside the configured limits, so the FC05 command remains 0.

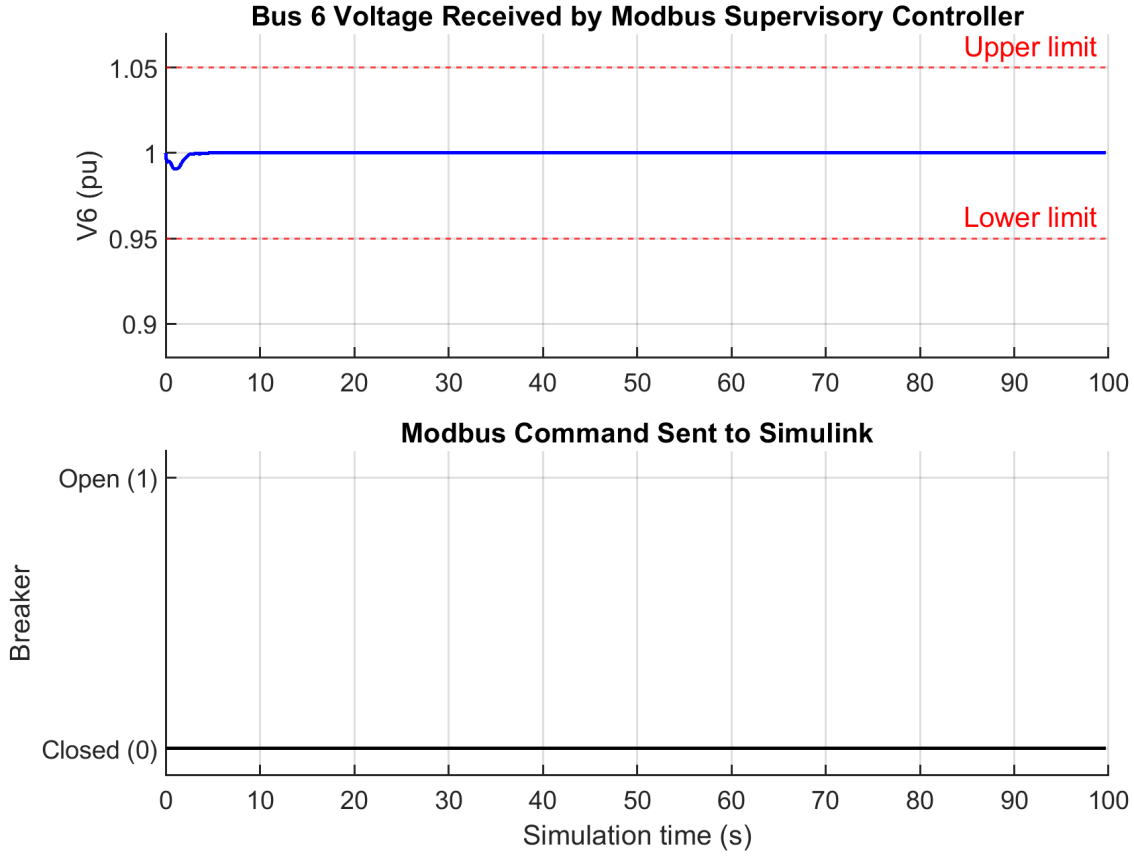


Figure 10-38: *Modbus TCP baseline: received Bus 6 voltage stays within the limits and the breaker remains closed.*

10.2.10.4 NetSim Cyber Modbus configuration

Figures 10-39–10-42 show the configuration sequence used for the Modbus example.

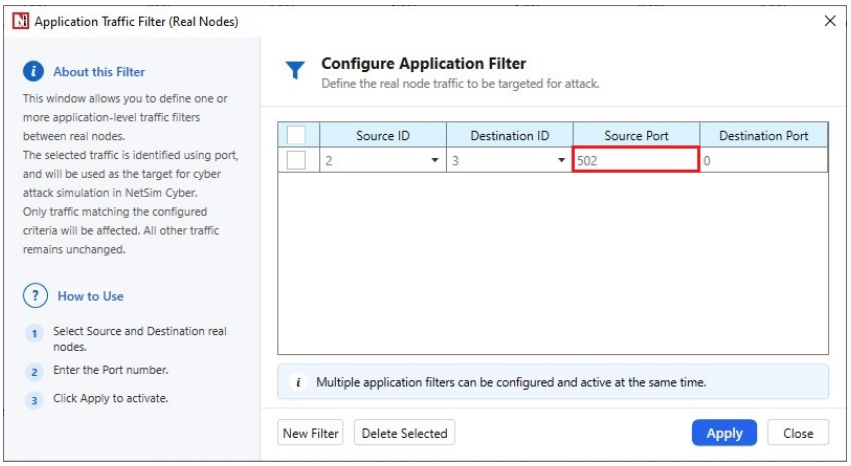


Figure 10-39: *MODBUS-1: application filter for Modbus response traffic from source port 502.*

The application filter selects the Real Node direction carrying responses from the Simulink Modbus server. Source port 502 identifies the server response; destination port 0 permits the controller's selected client port.

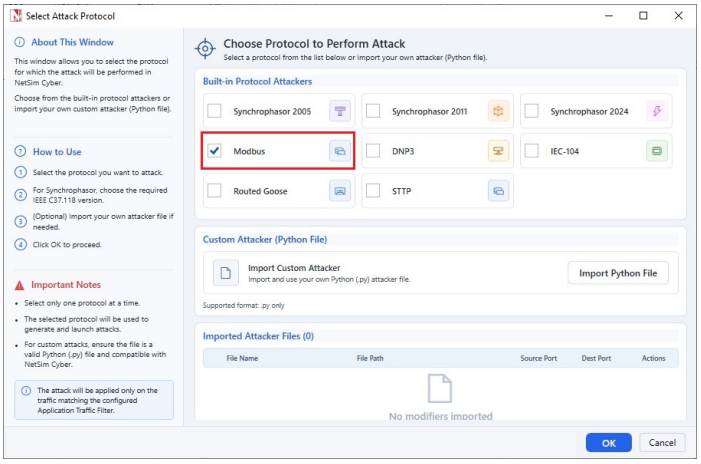


Figure 10-40: *MODBUS-2: selection of the built-in Modbus protocol attacker.*

Select **Modbus** under Built-in Protocol Attackers. A custom Python payload modifier is not required for this example because NetSim Cyber parses Modbus register responses.

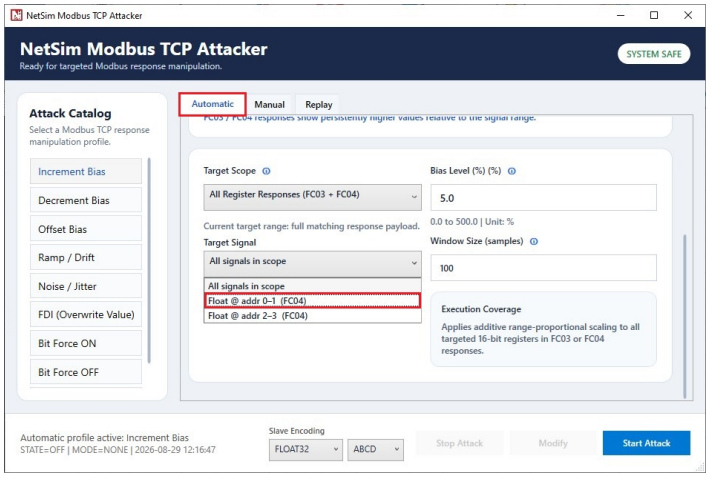


Figure 10-41: *MODBUS-3: selection of the Bus 6 voltage at FC04 addresses 0–1.*

After NetSim observes FC04 traffic, the Target Signal list shows two FLOAT32 values. Select **Float @ addr 0–1 (FC04)** to attack Bus 6 voltage. The value at addresses 2–3 is the simulation timestamp and is not selected for the voltage test.

The Automatic tab supports increment bias, decrement bias, offset bias, ramp/drift, noise/jitter and value overwrite. The illustrated setup applies range-scaled Gaussian Noise/Jitter to addresses 0–1. Keep the slave encoding at **FLOAT32** and the byte order at **ABCD**.

The Noise/Jitter magnitude is derived from the sliding-window range of the observed signal. Bus 6 voltage is nearly constant around 1.0 pu after startup, so a steady window produces a small range. Startup transients and variation inside the attack window enlarge that range and therefore the applied noise. The limit crossings in the result depend on the observed window and configured noise settings; they are not a fixed per-unit offset.

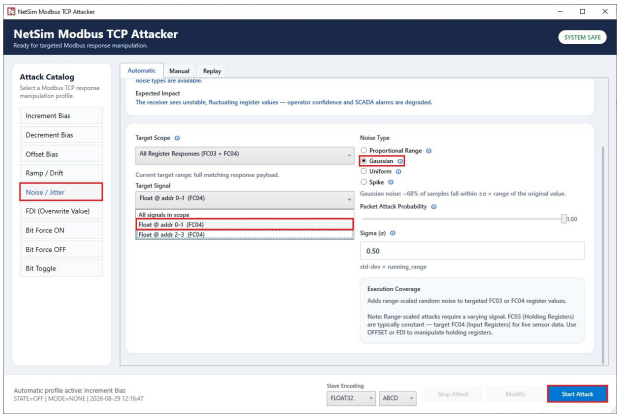


Figure 10-42: *MODBUS-4: Gaussian Noise/Jitter applied to the Bus 6 voltage signal with FLOAT32 ABCD encoding.*

10.2.10.5 Attack result with NetSim Cyber

During the attack intervals, the voltage received by the supervisory controller fluctuates and crosses the 0.95 pu and 1.05 pu limits. Each crossing produces FC05 command 1. When the received voltage returns inside the range, the supervisory controller writes command 0.

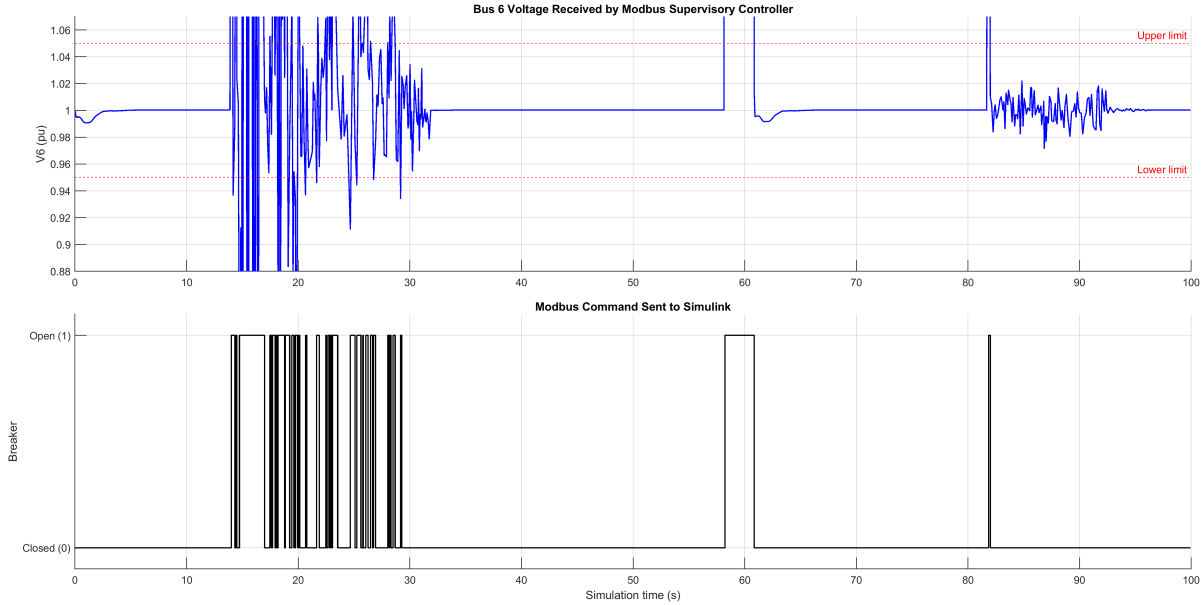


Figure 10-43: *Modbus TCP with NetSim Cyber: modified FC04 voltage responses cause corresponding Bus 6 breaker commands.*

Result interpretation. The upper plot is the network value received by the supervisory controller. The lower plot is the FC05 command returned to Simulink. The plant-side Bus 6 voltage scope shows the physical model value and does not reproduce the false network value.

10.2.10.6 Quantitative summary of the illustrated runs

Protocol	Applied manipulation	First visible trip	Observed command-1 activity
TCP	Fixed subtractive bias of 0.07 pu	Approximately 4.4 s	Three primary open intervals, with brief threshold recrossings
UDP	Fixed subtractive bias of 0.07 pu	Approximately 8.0 s	Three primary open intervals, with brief threshold recrossings
Modbus TCP	Range-scaled Gaussian Noise/Jitter on FC04 addresses 0–1	Approximately 14.0 s	Repeated trips from about 14–30 s, followed by shorter events near 59 s and 82 s

The times and interval counts above are read from the plotted example runs. They summarize the displayed evidence and are not protection-performance measurements.

10.2.11 TCP and UDP Packet Verification

This section verifies both communication representations. Section 10.1 compares the custom TCP and UDP measurement frames. Section 10.2 compares decoded Modbus FC04 responses and confirms that the Bus 6 voltage registers change while the timestamp registers remain unchanged.

10.2.11.1 Compare PRE and MOD packets

Wireshark shows the payload as raw data until the matching IEEE 9-bus Lua dissector is loaded. Use `ieee9bus_udp.lua` for the UDP example and `ieee9bus_tcp.lua` for the TCP example. The Lua file adds field names for the frame identifier, sequence, simulation time, Bus 6 voltage, breaker state and status.

Create the plugin folder and add the Lua dissector.

1. In Wireshark, select **Help** → **About Wireshark**, as shown in Figure 10-44.
2. Open the **Folders** tab and find the row named **Personal Lua Plugins**, as shown in Figure 10-45. The displayed path is specific to the signed-in Windows account.
3. The usual Windows location is `%APPDATA%\Wireshark\plugins`. For example, the screenshot shows `C:\Users\TEST\AppData\Roaming\Wireshark\plugins`.
4. Open File Explorer, enter `%APPDATA%\Wireshark` in the address bar and press Enter.
5. If a folder named `plugins` is not present, create it: right-click inside the `Wireshark` folder, select **New** → **Folder**, and name it `plugins`. The final path must end with `Wireshark\plugins`.
6. Copy `ieee9bus_udp.lua`, `ieee9bus_tcp.lua`, or both from the example files into the new `plugins` folder. Do not place these Lua files in the versioned binary-plugin folder such as `plugins\4.4`.
7. Return to Wireshark and select **Analyze** → **Reload Lua Plugins**, or press `Ctrl+Shift+L`. Restart Wireshark if the reload command is unavailable.
8. Open a capture and confirm that the packet-details pane shows **IEEE 9-Bus UDP Measurement Frame** or **IEEE 9-Bus TCP Measurement Frame** instead of only **Data**.

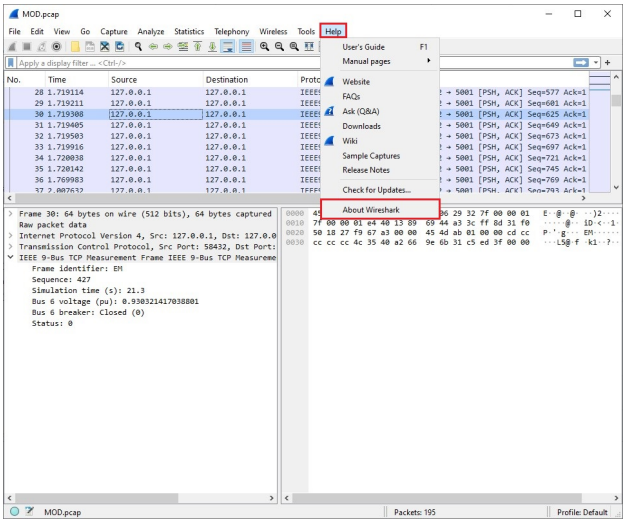


Figure 10-44: *Open Help and select About Wireshark.*

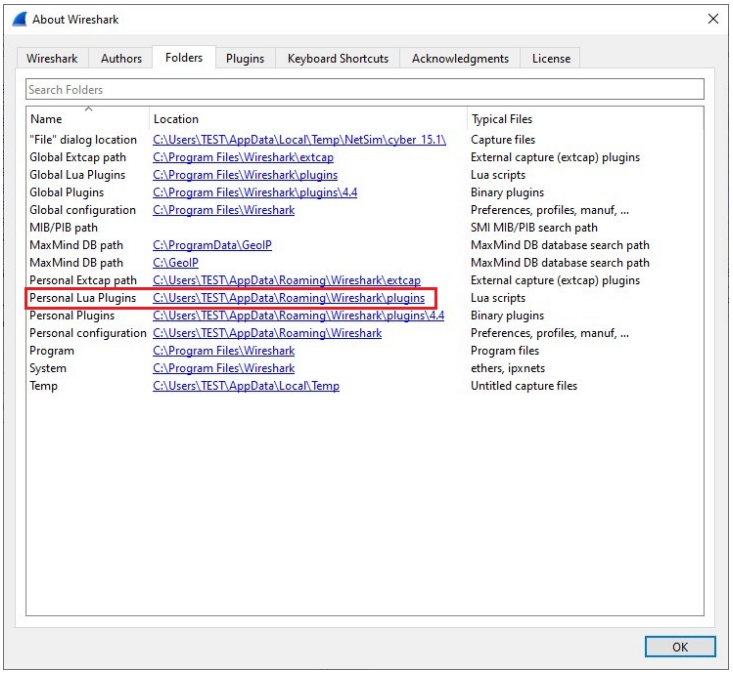
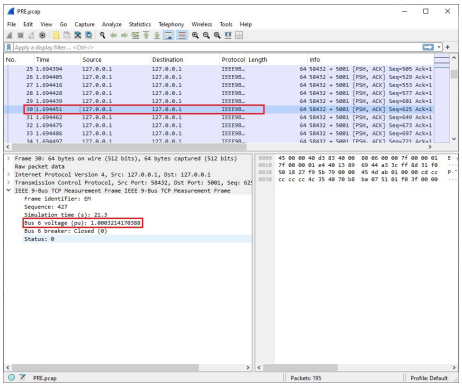


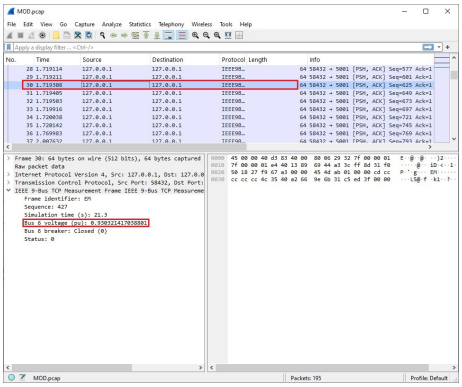
Figure 10-45: *On the Folders tab, use the path shown for Personal Lua Plugins. Create the final* `plugins` *folder when it is missing.*

Compare the captured packets.

1. Open `PRE.pcap` and `MOD.pcap` from the NetSim Cyber Results view.
2. Select the same measurement sequence in both captures.
3. Verify that the MOD voltage equals the PRE voltage minus the configured attack bias.
4. Check a later measurement for the breaker response. The breaker-state byte is not changed inside the same modified measurement frame.



(a) PRE capture: sequence 427 carries the original Bus 6 voltage.



(b) MOD capture: the same sequence carries the voltage after the FDIA bias.

Figure 10-46: *Wireshark decoding of matching TCP measurement frames before and after NetSim Cyber packet modification.*

Field	PRE example	MOD example	Expected difference
Sequence	427	427	No change
Simulation time	21.3 s	21.3 s	No change
Bus 6 voltage	1.000321 pu	0.930321 pu	−0.070000 pu
Breaker state in that frame	Closed (0)	Closed (0)	No direct overwrite

Causal sequence. Measurement on port 5001 → controller threshold check → command on port 5002 → Simulink breaker update → later measurement reports the resulting breaker state and voltage.

10.2.11.2 Modbus FC04 PRE and MOD verification

The Modbus packet captures are decoded by Wireshark's Modbus/TCP dissector. In both selected responses, the source port is 502, the function is FC04 Read Input Registers, and four registers are returned. Registers 0–1 encode the FLOAT32 Bus 6 voltage in ABCD byte order. Registers 2–3 encode the FLOAT32 Simulink timestamp.

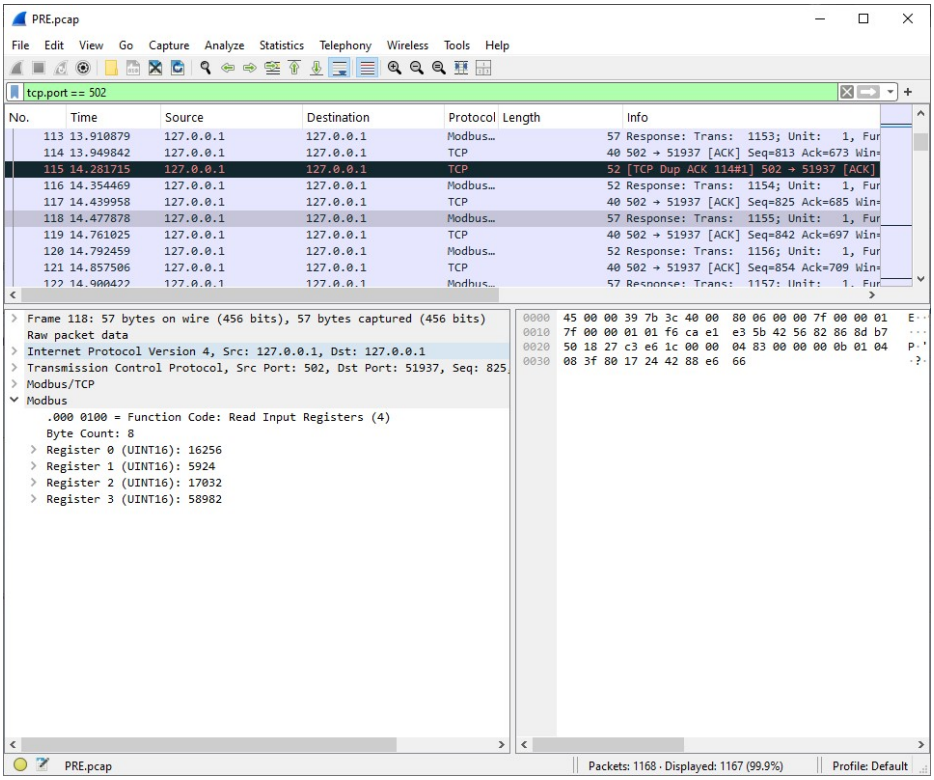


Figure 10-47: *Modbus PRE capture: FC04 response before NetSim Cyber modifies the Bus 6 voltage registers.*

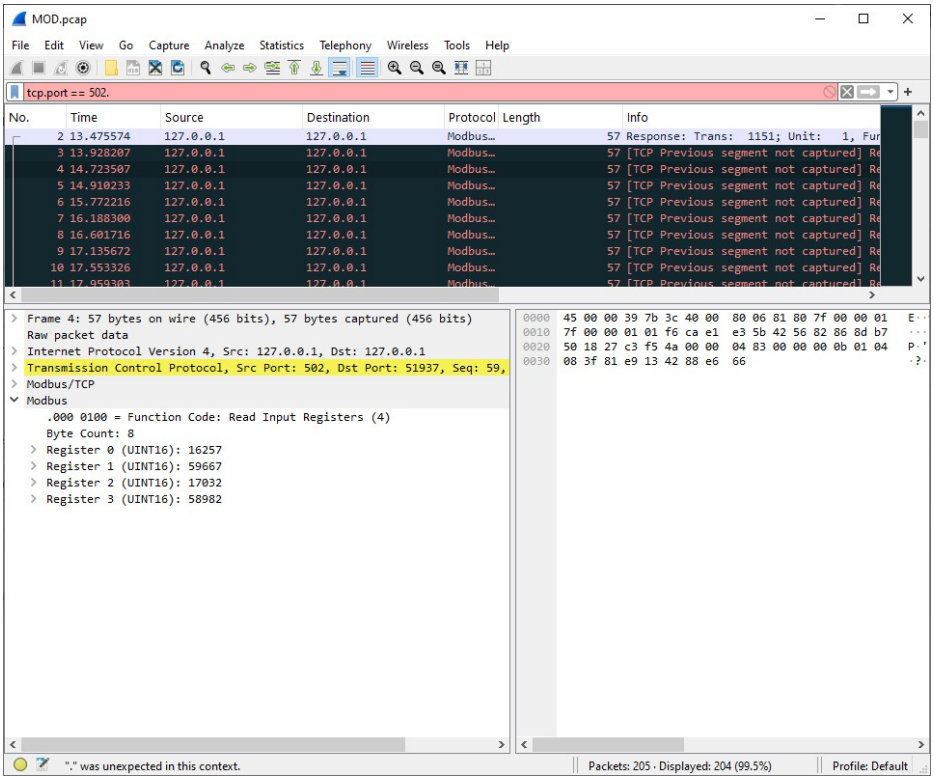


Figure 10-48: *Modbus MOD capture: FC04 response after NetSim Cyber modifies registers 0–1.*

Decoded field	PRE capture	MOD capture	Interpretation
Source port	502	502	Simulink Modbus-server response
Function	FC04	FC04	Read Input Registers
Registers 0–1	16256, 5924	16257, 59667	Targeted Bus 6 voltage words changed
Decoded voltage	1.0007062 pu	1.0149254 pu	Increase of approximately 0.0142192 pu in this packet
Registers 2–3	17032, 58982	17032, 58982	Timestamp words unchanged
Decoded timestamp	68.45 s	68.45 s	Same Simulink sample time

The PRE and MOD files are separate capture streams, so Wireshark packet numbers and TCP sequence information need not match. The matching timestamp and unchanged registers 2–3 identify the same Simulink sample, while the changed registers 0–1 show the Modbus voltage manipulation. Because the configured Noise/Jitter attack is variable, another selected packet may show a different positive or negative voltage change.

10.2.12 Run Procedure

10.2.12.1 Baseline run

1. Choose the TCP, UDP or Modbus TCP folder. Do not mix models or controller scripts between protocol folders.

2. Open the NetSim Cyber experiment and confirm the two Real Nodes, network, Threat Agent and application filter.
3. Leave the NetSim Cyber attacker stopped. Use the same Real Nodes, application filter, Simulink model, sample period and supervisory-controller thresholds as the attack run; no packet payload is modified in this baseline.
4. Start the supervisory controller. For TCP, wait for both Simulink clients to connect. For Modbus TCP, start Simulink after the controller begins waiting for port 502.
5. Run the Simulink model with the selected stop time and a 0.05 s communication period.
6. Save the controller plot and packet capture as the baseline evidence.

10.2.12.2 Attack run

1. For TCP or UDP, enable Packet Modifier and select `ieee9bus_bus6_fdia.py`. For Modbus TCP, select the built-in Modbus attacker.
2. Confirm that the Real Node direction and application filter match the selected protocol. TCP and UDP use measurement port 5001; Modbus responses use source port 502.
3. Enable PRE, MOD and POST captures.
4. Start the supervisory controller, then run Simulink and NetSim Cyber.
5. Keep the same thresholds, stop time and sample period used for the baseline run.
6. Compare received V6, breaker command, PRE/MOD packet values and later Simulink electrical response.

Two-machine setup. Replace 127.0.0.1 with the reachable host addresses in the Simulink blocks, controller settings and NetSim Real Nodes. Allow TCP/UDP ports 5001 and 5002, UDP command source port 5003, and Modbus TCP port 502 through the Windows firewall.

10.2.13 Example Package and Requirements

10.2.13.1 Software

Item	Tested configuration	Purpose
MATLAB/Simulink	MATLAB R2023a	IEEE 9-bus plant and controller execution
Instrument Control Toolbox	R2023a toolbox	TCP/IP client-server and UDP communication APIs
NetSim Cyber	15.1.x; current installed build 15.1.9	Real Node communication, packet modifier and captures
Python runtime used by NetSim Cyber	Installed for custom attacker	Executes the TCP and UDP Bus 6 payload modifier
Wireshark	Version supporting Lua plugins	Decodes PRE, MOD and POST captures
Operating system	Windows 10/11, 64-bit	Host environment for MATLAB, NetSim Cyber and Wireshark

10.2.13.2 File manifest

Keep each protocol example in its own folder so that its model and supervisory-controller script remain paired.

Package area	Required files	Purpose
UDP	`ee_ieee9bus_fdia.slx`, `supervisoryControllerSimulink.m`, `ieee9bus_bus6_fdia.py`, `ieee9bus_udp.lua`	Simulink model, UDP supervisory controller, packet modifier and Wireshark dissector
TCP	`ee_ieee9bus_fdia.slx`, `supervisoryControllerSimulink.m`, `ieee9bus_bus6_fdia.py`, `ieee9bus_tcp.lua`	Simulink model, TCP supervisory controller, packet modifier and Wireshark dissector
Modbus TCP	`ee_ieee9bus_fdia.slx`, `supervisoryControllerSimulink.m`, `modbusPlantServer.m`	Simulink model, Modbus supervisory controller and Level-2 Modbus plant server
NetSim Cyber	Experiment folder containing `Configuration.netsim` and its associated experiment resources	Topology, Real Nodes, Threat Agent, application filters and capture settings
Documentation	This `.tex` source, generated PDF and the `figures` folder	Reproducible document source and referenced images

10.2.14 Scope and Extensions

The packaged examples focus on Bus 6 to keep the communication path, data mapping and controller decision easy to follow. TCP and UDP target the measurement path to destination port 5001. Modbus TCP targets FC04 responses from source port 502.

10.2.14.1 Current scope

- One measured voltage: Bus 6 voltage magnitude.
- One reported device state: Bus 6 breaker state.
- One controlled device: Bus 6 circuit breaker.
- TCP and UDP manipulation: subtract a fixed 0.07 pu bias from the Bus 6 voltage field.
- Modbus TCP manipulation: apply range-scaled Gaussian Noise/Jitter to FC04 addresses 0–1.
- Three communication variants: TCP, UDP and Modbus TCP.

10.2.14.2 Extensions

- Route the command path through a second NetSim Cyber filter and attack the command byte.
- Add delay, loss, duplication, replay and connection-interruption scenarios.
- Apply Modbus increment bias, ramp/drift and noise/jitter to FC04 addresses 0–1.
- Add hysteresis or time qualification to reduce command changes near the voltage limit.
- Extend the measurement frame to additional buses, angles, currents or breaker states.

- Measure network delay, command latency, packet loss and power-system recovery time.

Safety boundary. This example is intended for simulation and cybersecurity studies. Controller thresholds and breaker logic must not be used for field protection or grid operation without a separate engineering design and protection study.

10.2.15 Conclusion

The example links an IEEE 9-bus Simulink plant, a MATLAB supervisory controller and NetSim Cyber in a closed measurement-and-command loop. The TCP and UDP variants exchange custom 24-byte measurement and 18-byte command frames. The Modbus TCP variant exchanges FC04 measurements and an FC05 breaker command through port 502.

All three variants use the same Bus 6 voltage limits and breaker convention. The baseline and attack plots show how a modified network measurement changes the value evaluated by the controller and therefore changes the command returned to Simulink.

10.2.16 References

1. MathWorks, "IEEE 9-Bus System," https://www.mathworks.com/help/sps/ug/ieee-9-bus.html.
2. MathWorks, "Simulink Documentation," https://www.mathworks.com/help/simulink/.
3. MathWorks, Instrument Control Toolbox documentation for UDP and TCP/IP communication.

10.3 Modbus Replay Attack Example

10.3.1 Objective

This example validates the Modbus replay mode supplied with NetSim Cyber by using a varying FC04 input-register signal. The Threat Agent first records valid FC04 response bodies while the Slave is in the Normal scenario. The Overvoltage scenario is then applied, changing the current FC04 voltage from approximately 110 V to 124 V. During replay, the Threat Agent replaces matching live response bodies with the recorded normal bodies, so the Master receives the earlier normal signal while the Slave remains in Overvoltage.

The Threat Agent forwards one modified live packet for each intercepted packet. It does not add a second replay packet. Replay is applied only when the recorded response body and the current response body have the same length.

10.3.2 Saved Example Configuration

The workspace contains the experiment `Modbus_Replay`. It uses a Modbus Master, a Modbus Slave, one NetSim Cyber network, one Threat Agent and a traffic filter between the endpoints.

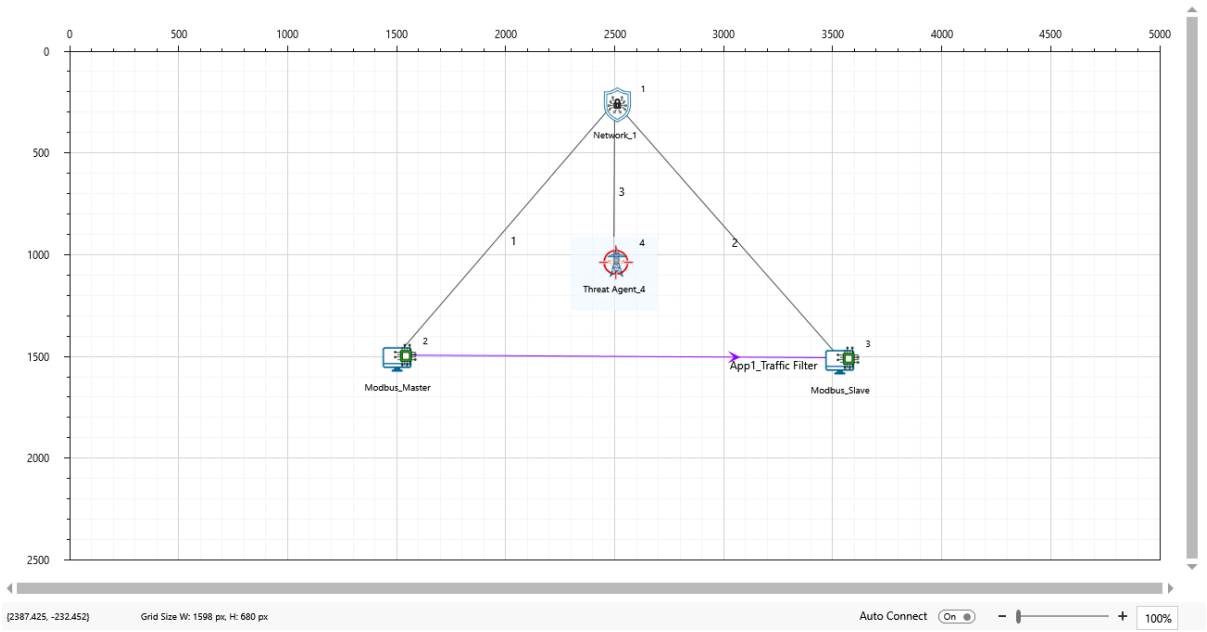


Figure 10-49: *Saved Modbus replay experiment in NetSim Cyber.*

Table 10-1: *Example configuration.*

Item	Configuration
NetSim Cyber version	15.1.12 (64 Bit)
Experiment	`Modbus_Replay`
Modbus endpoint	`127.0.0.1:502`, Unit ID 1
Function code	FC04 – Read Input Registers
Register selection	Start address 0, count 10
Signals	Voltage Phase A, Voltage Phase B, Voltage Phase C, Current and Frequency
Value encoding	FLOAT32, ABCD byte order
Poll interval	500 ms
Replay buffer in this run	19 FC04 responses (9.5-second loop)

The same function code, start address, count, encoding and poll interval are retained during recording and replay. This keeps the response-body length constant and preserves the timing of the recorded sequence.

10.3.3 Open the Saved Experiment

1. Copy the supplied workspace to a location where experiment output can be saved.
2. Open NetSim Cyber and select the supplied workspace.
3. Open the `Modbus_Replay` experiment.
4. Confirm that the Modbus Master, Modbus Slave, network and Threat Agent appear as shown in Figure 1.
5. Enable PRE, MOD and POST packet capture if packet-level verification is required.

Select **Options** on the NetSim Cyber ribbon.

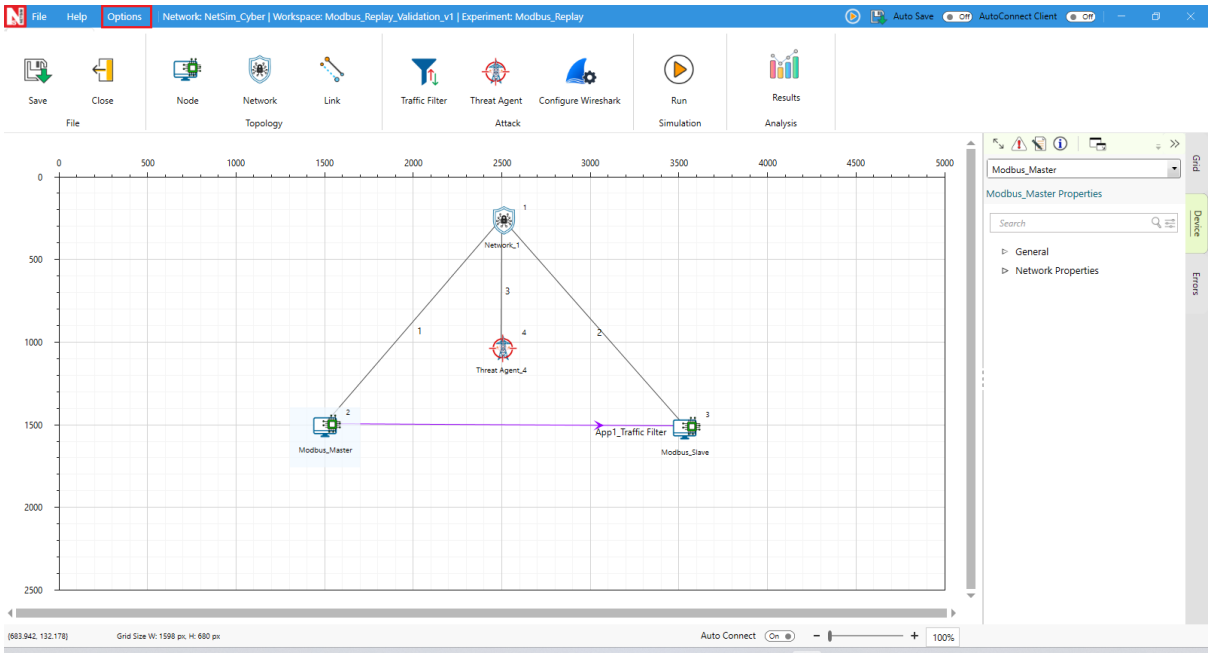


Figure 10-50: *Options menu in the saved Modbus replay experiment.*

Select **Open Simulators Folder** to open the simulator folder for the installed NetSim Cyber version.

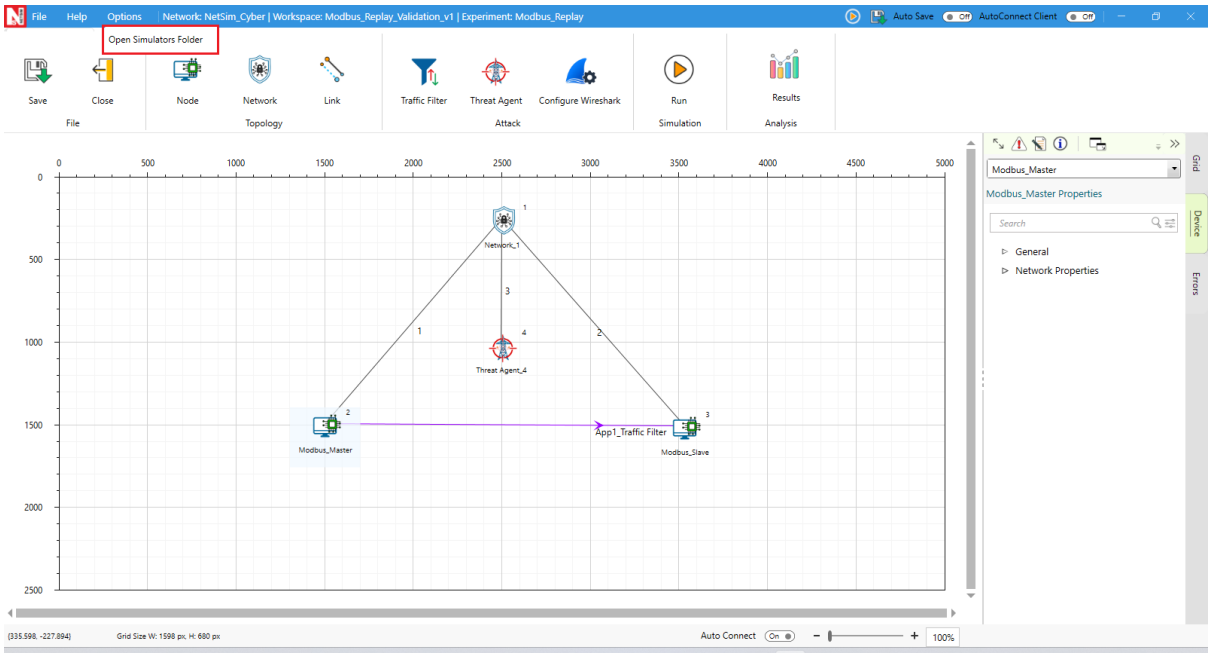


Figure 10-51: *Open Simulators Folder command used to locate the Modbus programs.*

10.3.4 Start Modbus Communication

10.3.4.1 Start the Modbus Slave

In the `Modbus-Simulator` folder, double-click `Modbus-Slave.pyw`. Start the Slave before the Master so that port 502 is listening when the Master connects.

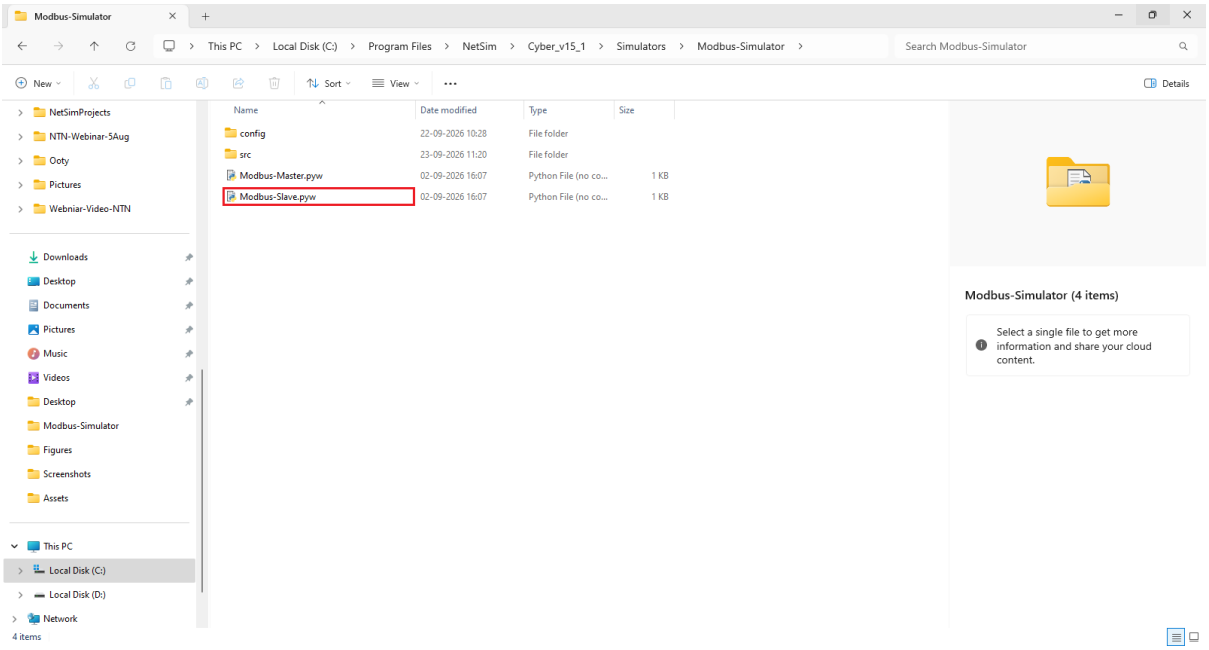


Figure 10-52: *Launch the Modbus TCP Slave Simulator first.*

Use `127.0.0.1`, TCP port 502 and Unit ID 1, then select **Start Server**. Leave the Slave in the Normal scenario during the recording stage. Do not use Signal Override. The FC04 signals change slightly as the simulator runs.

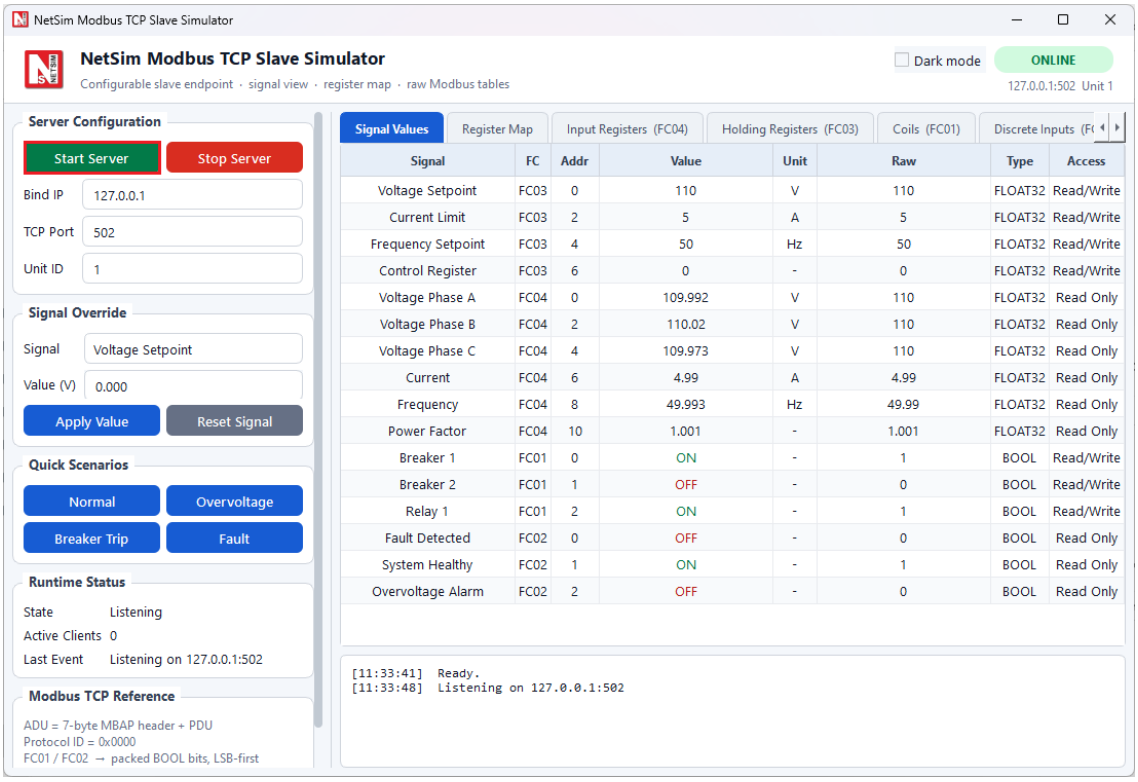


Figure 10-53: *Modbus Slave running with naturally varying FC04 signals before recording.*

10.3.4.2 Run NetSim Cyber and Start the Modbus Master

Return to the `Modbus-Simulator` folder and double-click `Modbus-Master.pyw`.

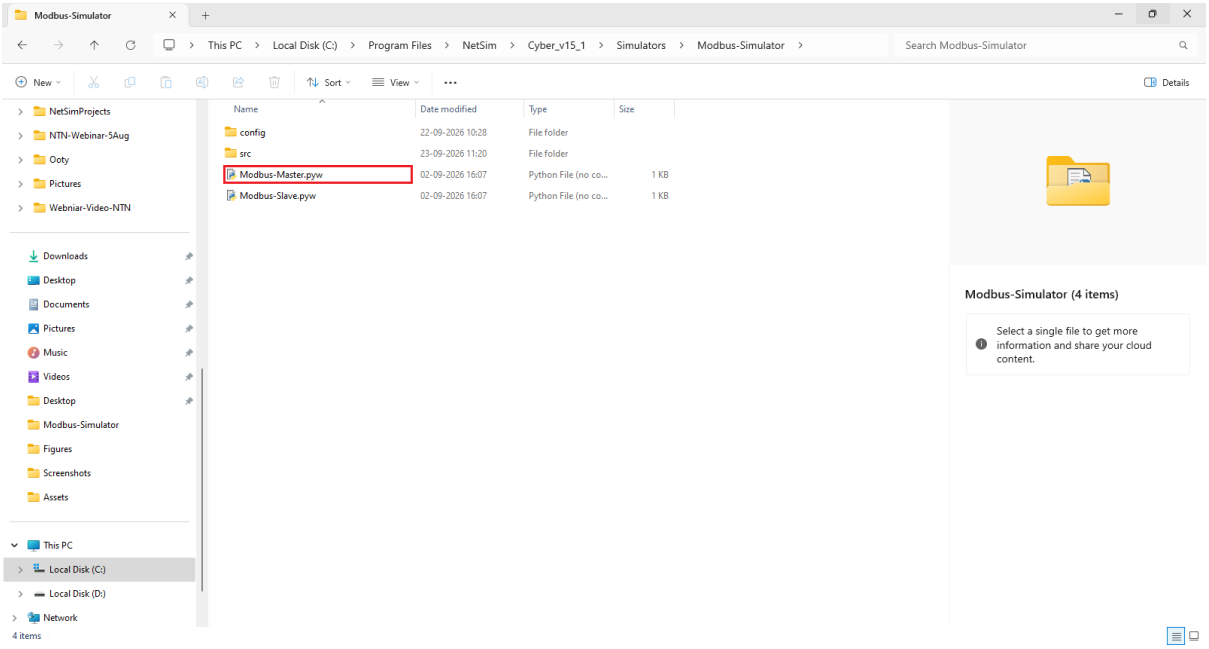


Figure 10-54: *Launch the Modbus TCP Master Simulator after starting the Slave.*

Start the NetSim Cyber experiment. If the Master was connected before the run started, disconnect and reconnect it so that the TCP flow passes through the Threat Agent.

Configure the Master as shown in Table 2. Select **Connect**, followed by **Start Poll**.

Table 10-2: *Modbus Master settings.*

Parameter	Value
IP address	`127.0.0.1`
TCP port	502
Unit ID	1
Function code	FC04 – Read Input Registers
Start address	0
Count	10
Poll rate	500 ms
Display format	FLOAT32 (ABCD)

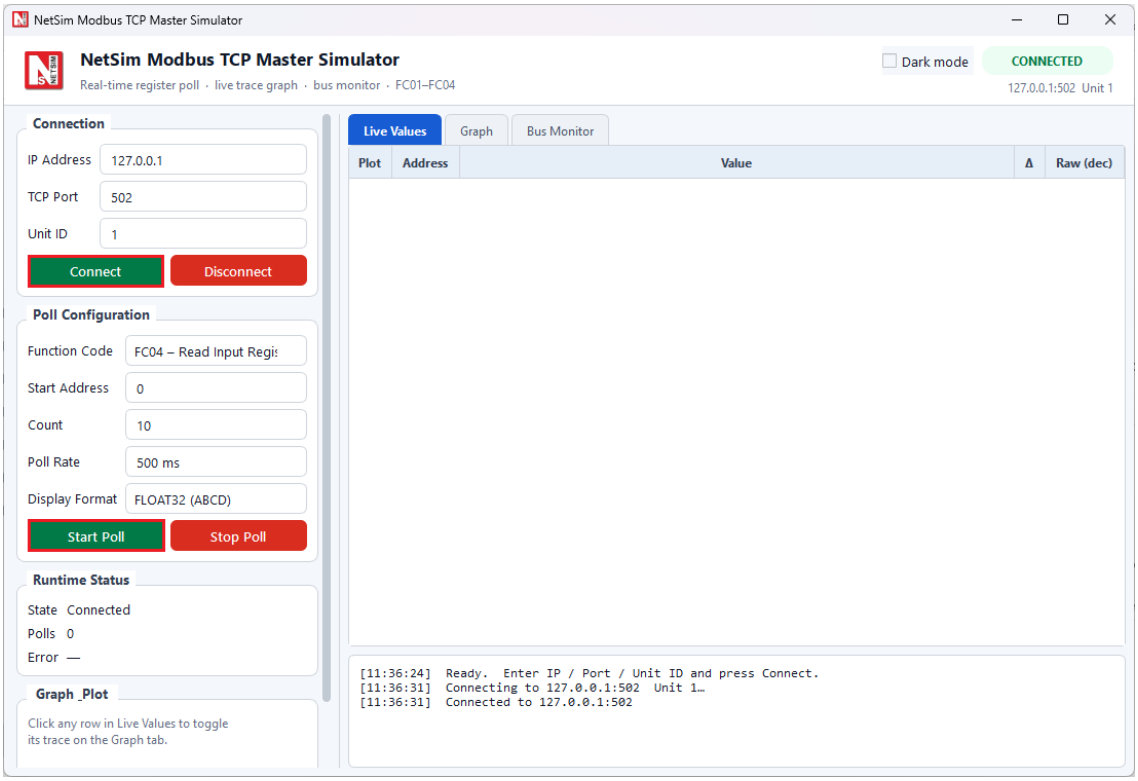


Figure 10-55: *Master configured to poll ten FC04 input registers every 500 ms.*

Open the **Graph** tab and select Voltage Phase A. Confirm that the value changes slightly over time. This varying live trace is the source sequence for the replay buffer.

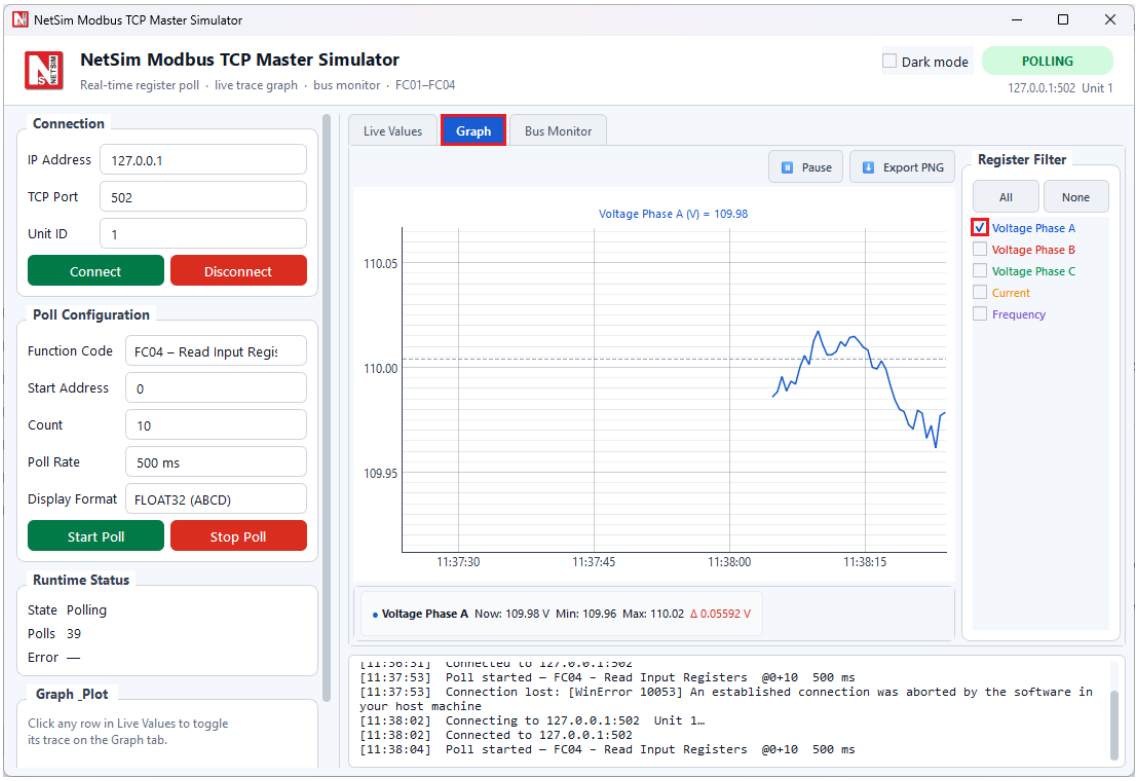


Figure 10-56: *Live Voltage Phase A trace before recording starts.*

10.3.5 Record the FC04 Sequence

Open the NetSim Modbus TCP Attacker, select the **Replay** tab, open Target Function Codes and select **FC04 Input Registers**.

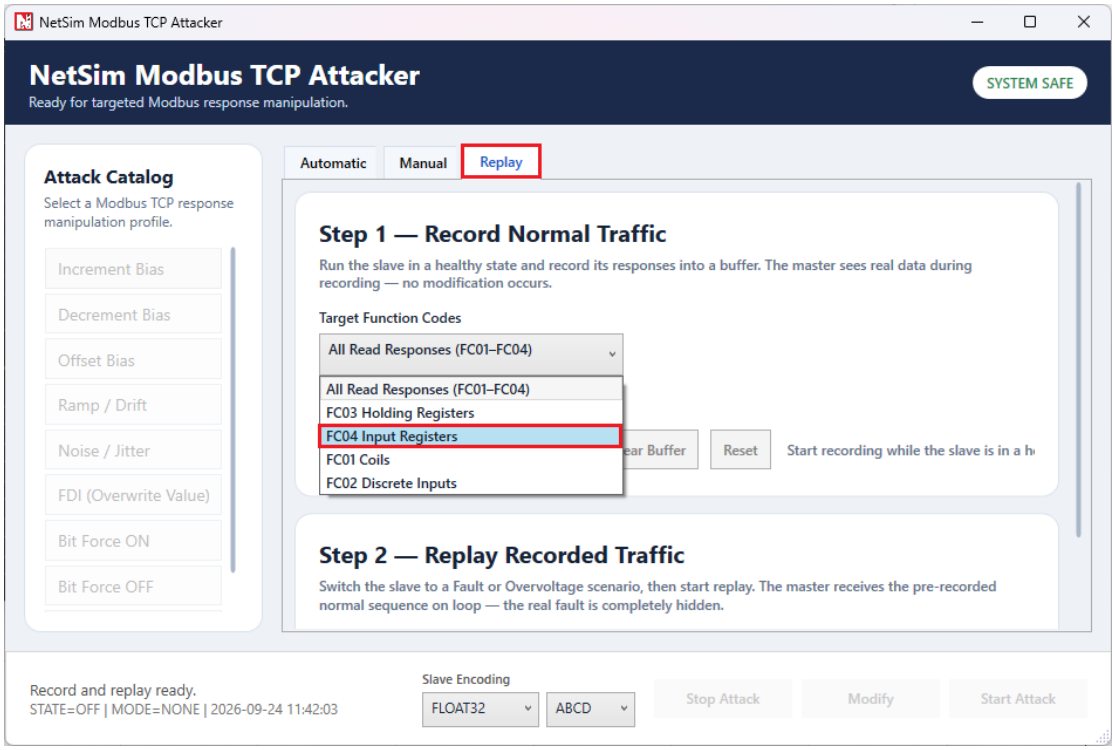


Figure 10-57: *FC04 Input Registers selected as the recording and replay target.*

Select **Start Recording** while the Master continues polling and the Slave remains in its normal simulation state.

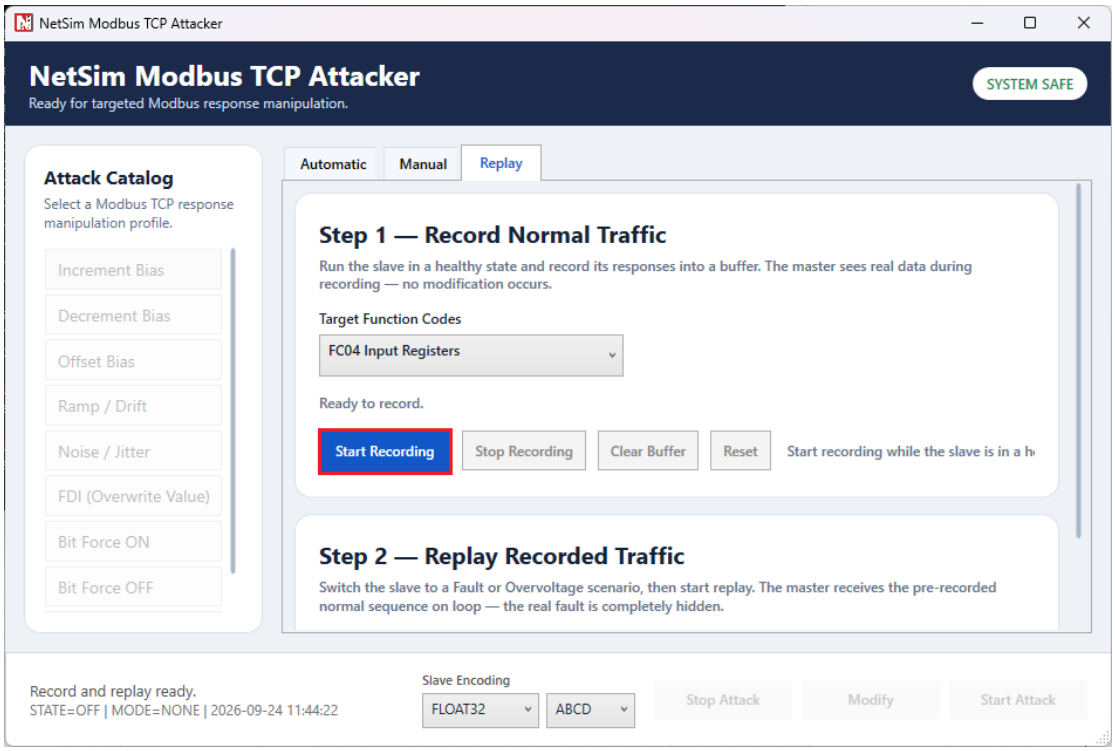


Figure 10-58: *Start Recording control for FC04 responses.*

The packet counter increases as valid FC04 response bodies are added to the replay buffer.

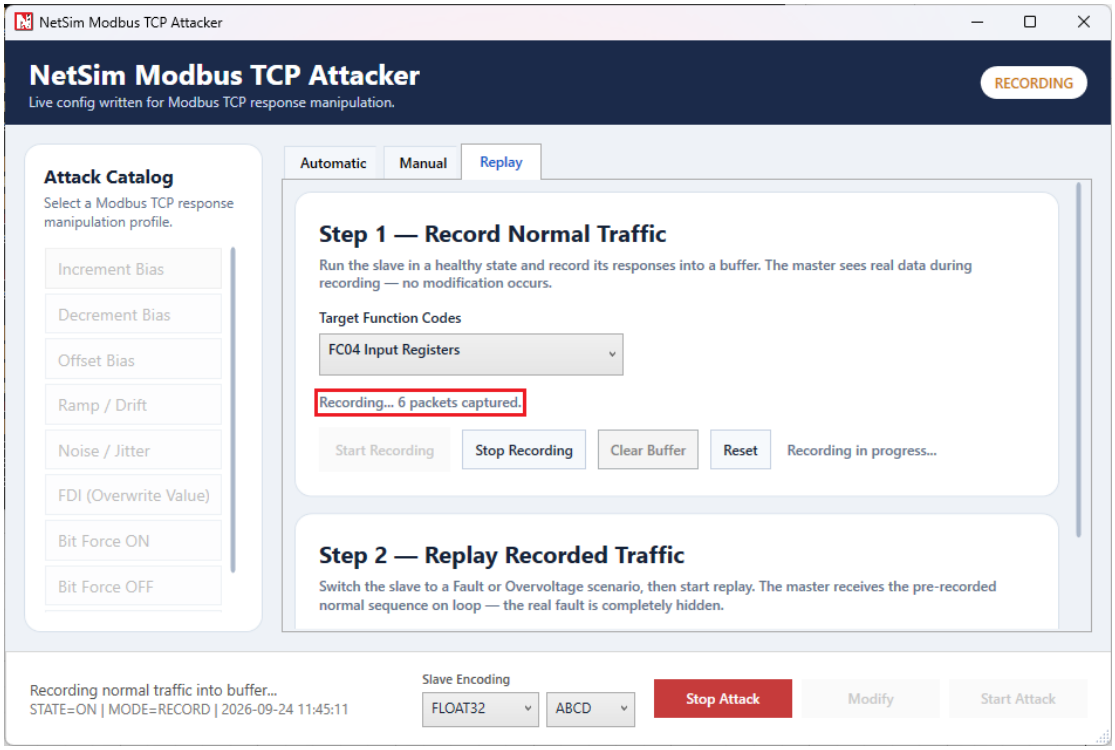


Figure 10-59: *FC04 recording in progress.*

Select **Stop Recording** after the required sequence has been captured. Do not select Clear Buffer or Reset because either action removes the sequence.

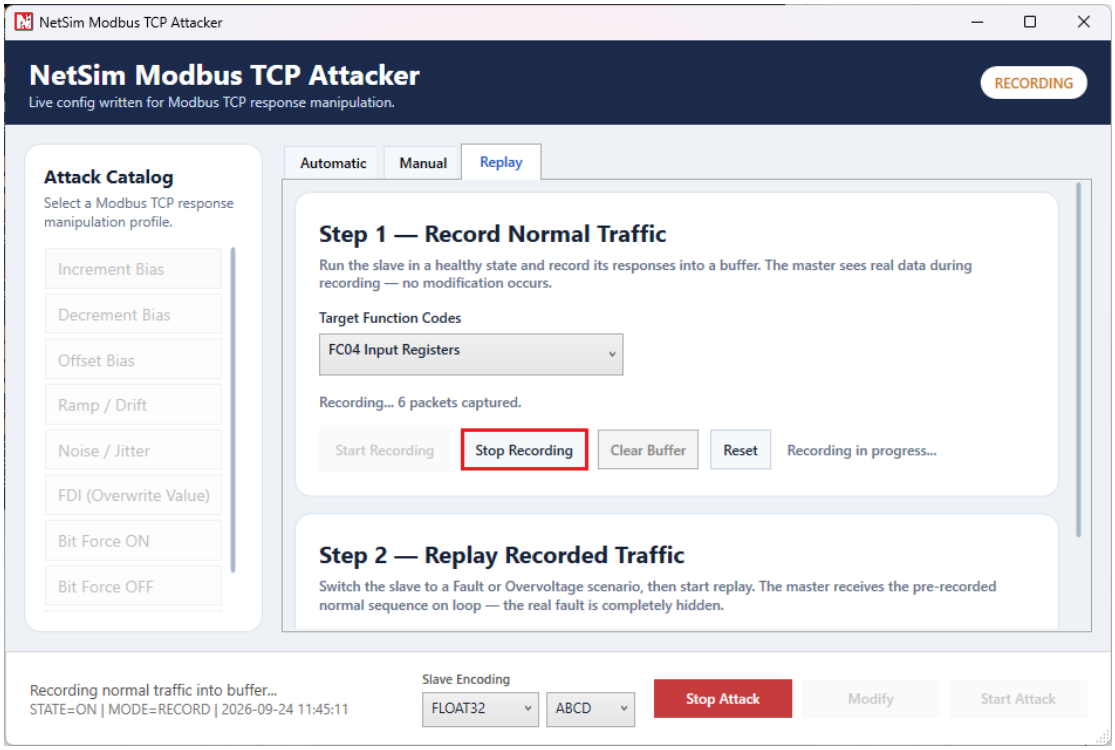


Figure 10-60: *Stop Recording after capturing the varying FC04 sequence.*

Confirm that the attacker reports **Buffer is ready**. The run shown here captured 19 responses. At a 500 ms poll interval, one traversal of the buffer lasts 9.5 seconds.

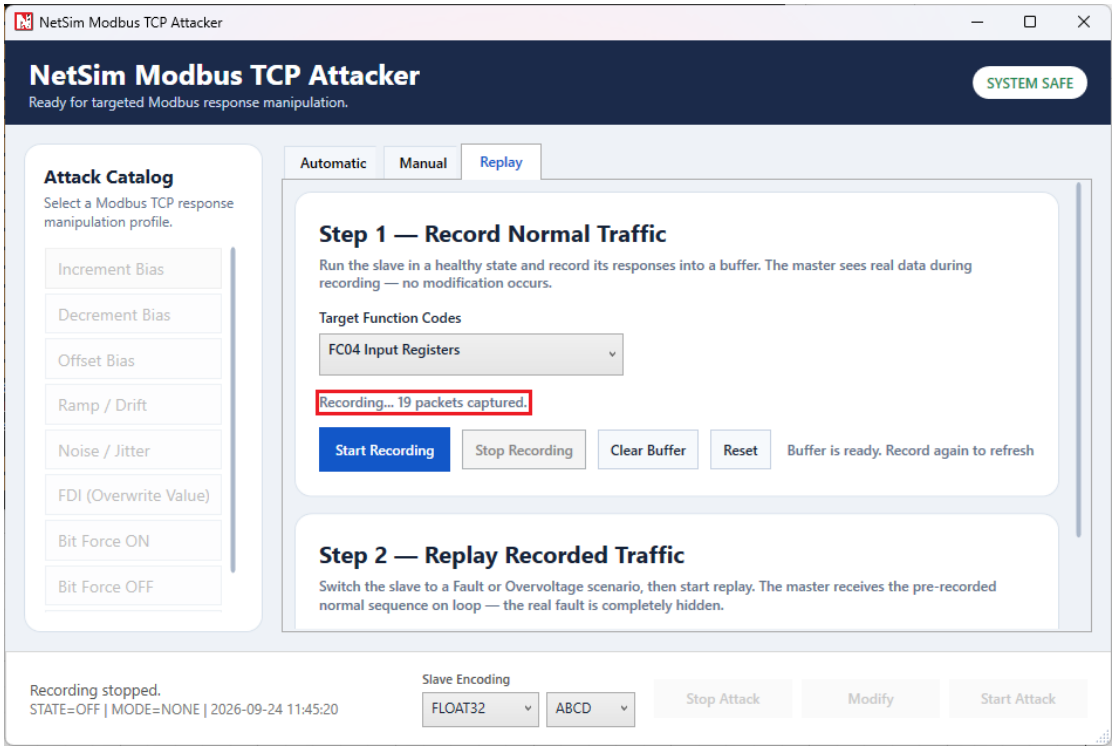


Figure 10-61: *Replay buffer ready with 19 recorded FC04 responses.*

For a longer interval before the sequence repeats, record more packets. The loop duration equals the recorded response count multiplied by the poll interval.

10.3.6 Apply the Overvoltage Scenario

After stopping the recording, return to the Slave Simulator and select **Overvoltage** under Quick Scenarios. FC04 is read-only to a Modbus client; the simulator scenario changes the internal sensor values rather than issuing a Modbus register write.

Confirm that Voltage Phase A, B and C are approximately 124 V and that the Overvoltage Alarm is ON.

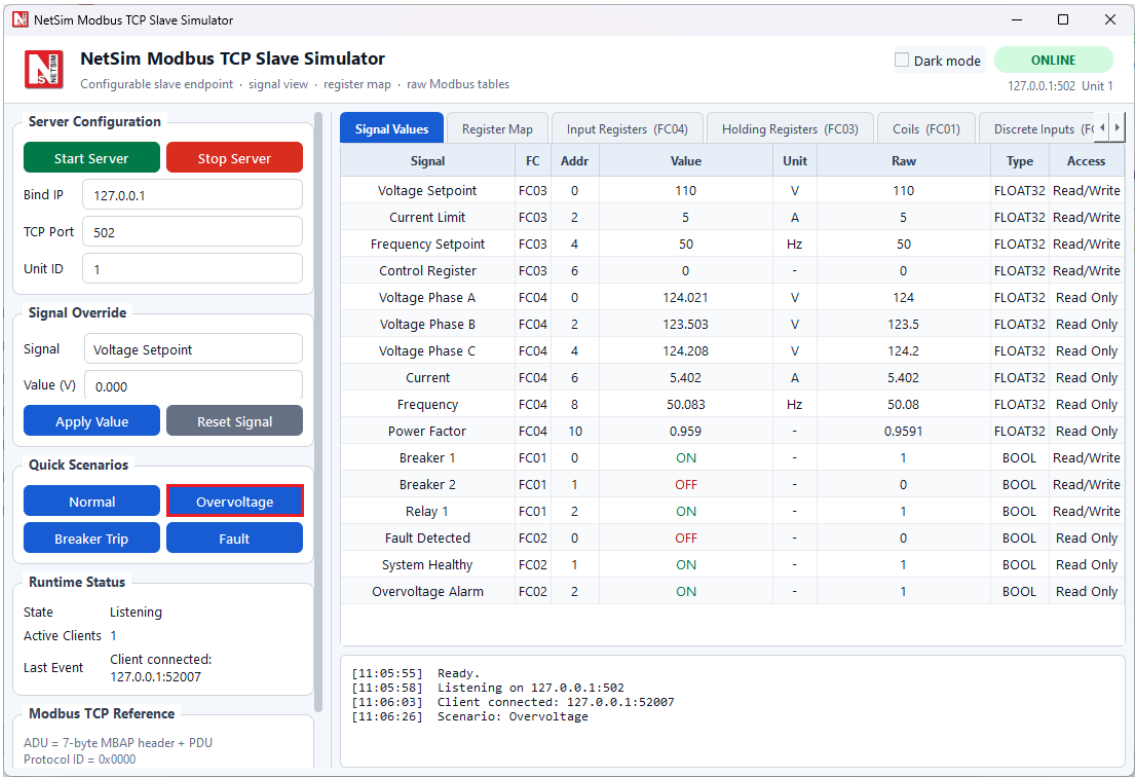


Figure 10-62: *Slave in the Overvoltage scenario with current FC04 voltage near 124 V.*

Before starting replay, confirm that the Master receives the changed live value. The graph should rise from the normal value near 110 V to the Overvoltage value near 124 V.

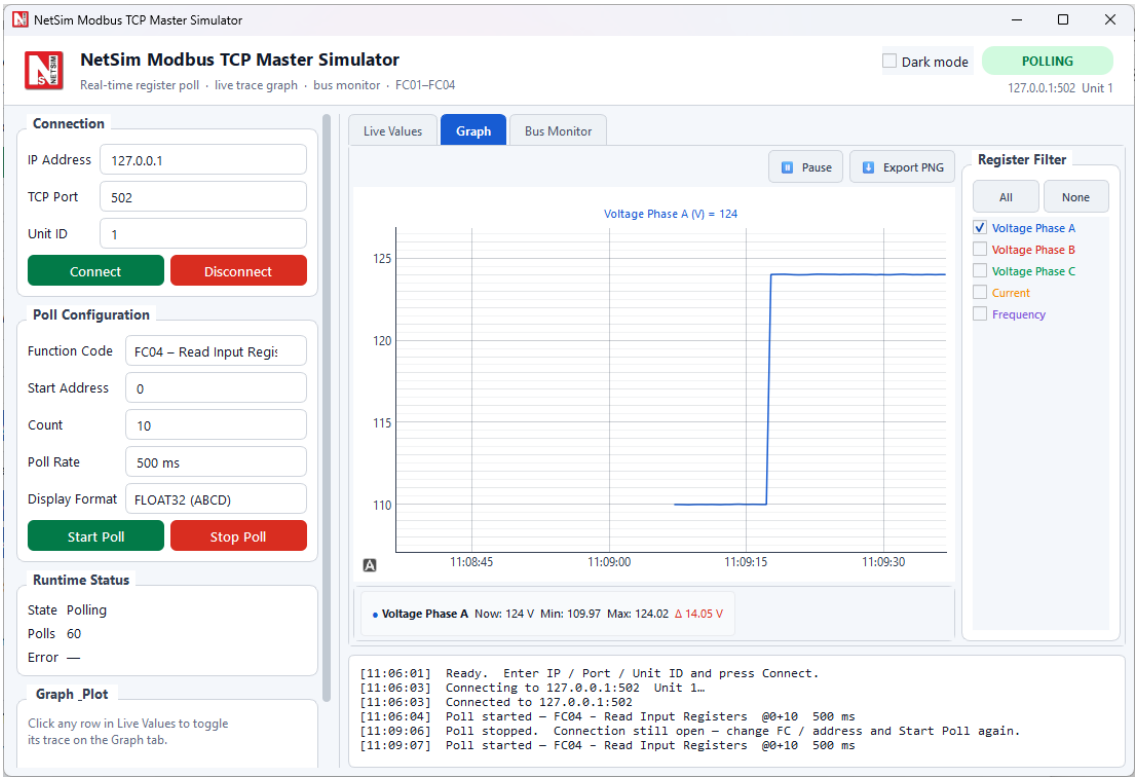


Figure 10-63: *Master receiving the live Overvoltage value before replay starts.*

10.3.7 Run the Replay Attack

Keep the Slave in the Overvoltage scenario and keep the Master polling with the same FC04 settings used during recording.

Select **Start Replay**.

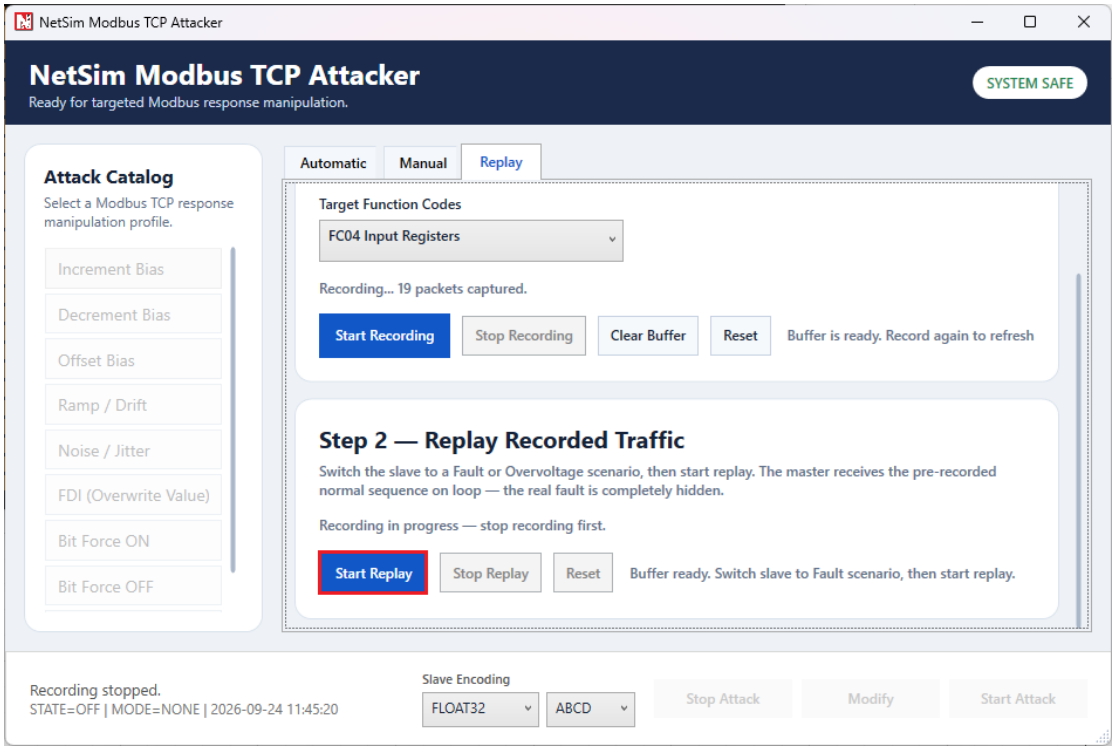


Figure 10-64: *Start Replay using the recorded FC04 buffer.*

The attacker enters REPLAY mode and advances through the 19 recorded response bodies. For each matching live FC04 response, it substitutes the next recorded body and forwards the modified live response to the Master.

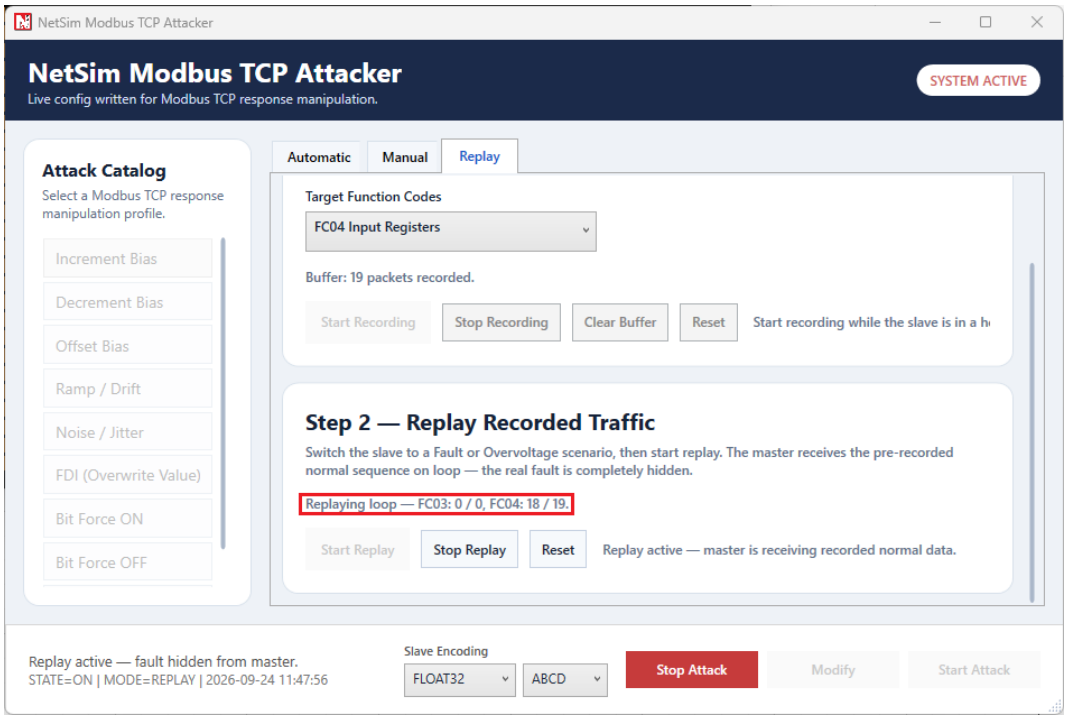


Figure 10-65: *FC04 replay loop running with the recorded buffer.*

Select **Stop Replay** after observing several loop traversals. The Master then resumes receiving the current FC04 response bodies.

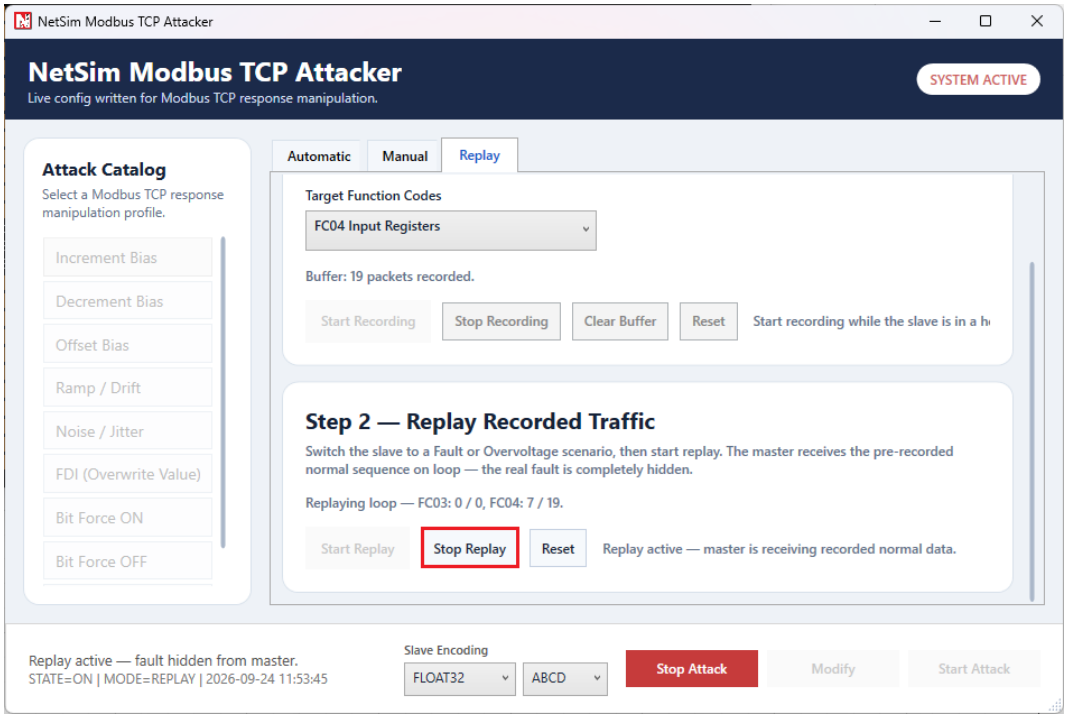


Figure 10-66: *Stop Replay after validating the repeated sequence.*

10.3.8 Verify Overvoltage Concealment

Figure 10-67 shows two replay intervals. While replay is stopped, the Master receives the live Overvoltage value near 124 V. Each time replay is started, the Master drops to the recorded normal signal near 110 V even though the Slave remains in Overvoltage. Stopping replay returns the Master to the current value near 124 V.

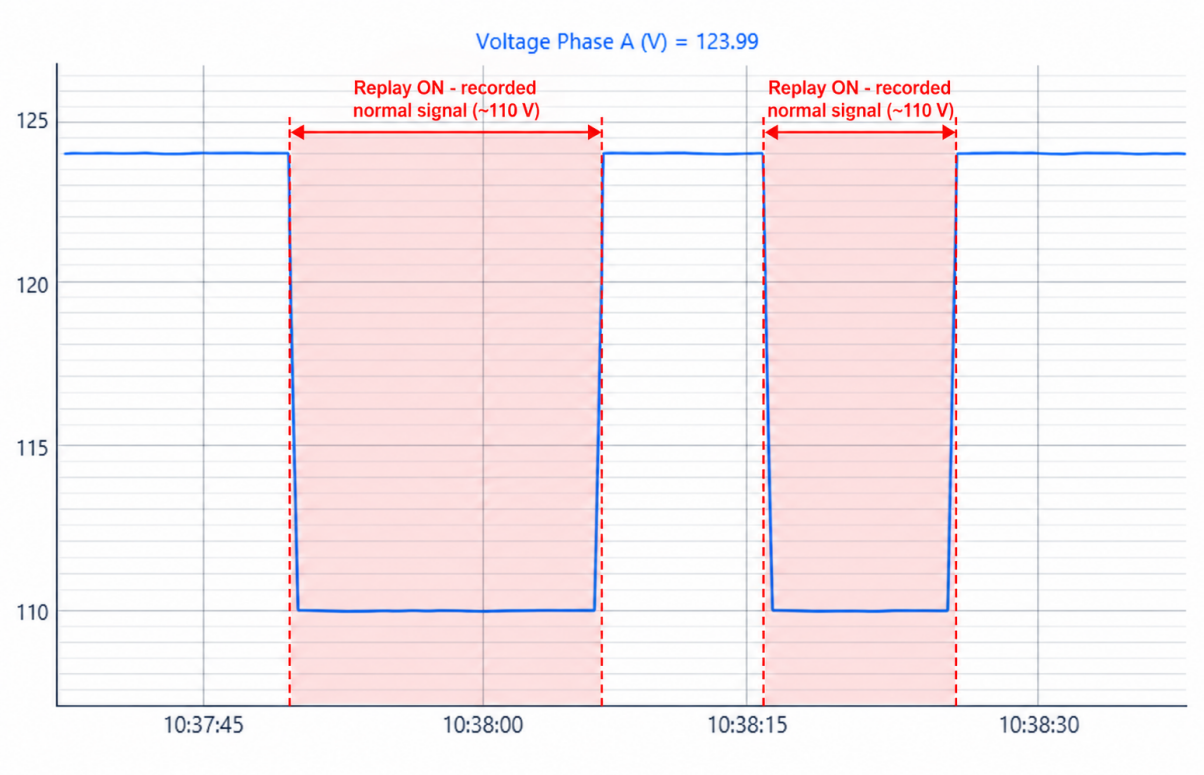


Figure 10-67: *Dotted boundaries mark the two Replay ON intervals, where the recorded signal near 110 V conceals the live Overvoltage response near 124 V.*

Graph display note. The voltage signal continues to vary during both Normal and Overvoltage operation. The variation appears nearly flat in Figure 10-67 because the automatic Y-axis must display both 110 V and 124 V. The graph shows a rolling 60-second window and recalculates its Y-axis from the values in that window. When the large 110/124 V transition moves outside the window, the axis narrows and the small variations become visible again. This is graph auto-scaling behavior, not a replay or Modbus error.

The graph is plotted against the current arrival time. Replay therefore places recorded values into current live responses and plots them at new timestamps; it does not reinsert an earlier packet with its original timestamp.

Table 10-3: *Observed behavior in the Overvoltage replay run.*

Stage	Mode	Slave	Master
Normal recording	RECORD	Normal, approximately 110 V	Normal varying values are stored
Overvoltage before replay	OFF	Overvoltage, approximately 124 V	Live value near 124 V
Replay running	REPLAY	Remains near 124 V	Recorded normal signal near 110 V
Replay stopped	OFF	Remains near 124 V	Returns to live value near 124 V

10.3.9 Packet Verification

10.3.9.1 How to Check the Capture Files

NetSim Cyber creates separate packet captures for traffic before processing (PRE), packets modified by the Threat Agent (MOD), forwarded traffic (POST) and dropped traffic (DROP). Use the following procedure after the simulation ends:

1. Open `PRE.pcap` in Wireshark.
2. Enter `tcp.port == 502` in the display-filter bar and press Enter. Read the **Displayed** count from the status bar at the bottom of the window.
3. Repeat the same filter and count check for `MOD.pcap`, `POST.pcap` and `DROP.pcap`. The same filter must be used for every file.
4. Confirm that the PRE and POST displayed counts are equal. MOD should contain only the responses changed during replay, and DROP should contain no matching packet.
5. To inspect the response used in this example, apply the following filter in PRE, MOD and POST:

```
tcp.srcport == 502 && modbus.func_code == 4 && mbtcp.trans_id == 457
```

6. Select the displayed response. In the Packet Details pane, expand **Modbus/TCP** and **Modbus**. Check the transaction ID, Unit ID, function code, byte count and register values.
7. Compare the matching response across PRE, MOD and POST. Packet numbers may differ between capture files, so use the transaction ID and protocol fields to identify the same response.

For this FC04 response, Voltage Phase A is stored as a 32-bit floating-point value across registers 0 and 1. PRE shows the live Overvoltage register pair, while MOD and POST show the recorded normal register pair. The byte count and payload length must remain unchanged.

The PRE, MOD, POST and DROP captures from the FC04 validation run were filtered with `tcp.port == 502`. Table 10-4 records the measured packet counts.

Table 10-4: *FC04 Modbus/TCP packet-count evidence.*

Capture	Packets	Interpretation
PRE	1008	Modbus/TCP packets before Threat Agent processing
MOD	80	FC04 responses whose bodies were replaced during replay
POST	1008	Modbus/TCP packets forwarded after processing
DROP	0	No Modbus/TCP packet was dropped

The PRE and POST counts are equal. The Threat Agent forwarded one packet for each intercepted packet and did not add a replay packet. MOD contains 80 replaced FC04 responses and is a subset of the forwarded traffic. DROP is zero.

10.3.9.2 Matching FC04 Response

Transaction ID 457 was selected from the replay interval for the before-and-after comparison. The PRE response carries the live Overvoltage value, while MOD and POST carry the recorded normal body. Table 10-5 lists the decoded fields.

Table 10-5: *FC04 response-body comparison for transaction ID 457.*

Field	PRE	MOD	POST
Capture frame	1491	2	1491
Unit ID	1	1	1
Function code	FC04	FC04	FC04
Byte count	20	20	20
TCP payload length	29 bytes	29 bytes	29 bytes
First register pair	17144, 288	17115, 64337	17115, 64337
Phase-A voltage	124.0022 V	109.9909 V	109.9909 V
Body state	Live Overvoltage	Recorded normal	Forwarded replay body

The transaction ID, Unit ID, function code, byte count and payload length remain unchanged. The first register pair changes from the live value near 124 V in PRE to the recorded value near 110 V in MOD. POST contains the same body as MOD, confirming that the modified live response was forwarded without creating another packet.

PRE Capture Evidence

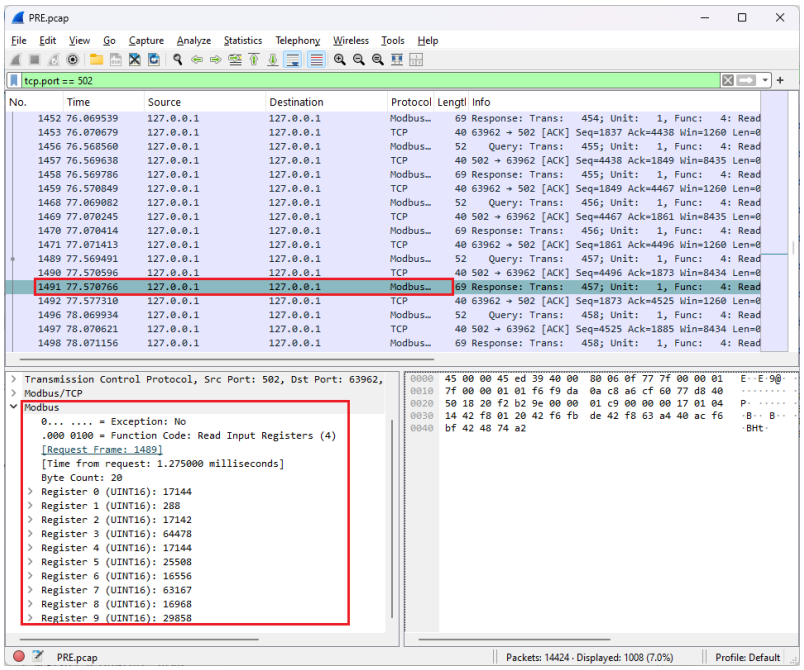


Figure 10-68: *PRE capture for transaction ID 457. The filter shows 1008 Modbus/TCP packets, and the selected FC04 response contains the live Overvoltage body near 124 V.*

MOD Capture Evidence

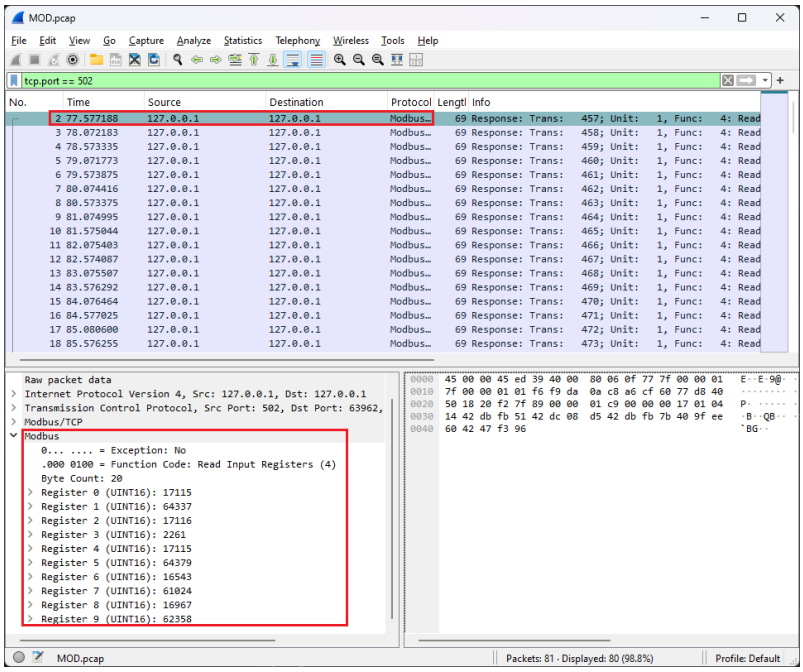


Figure 10-69: *MOD capture for transaction ID 457. The filter shows 80 modified FC04 responses, and the selected body contains the recorded normal value near 110 V.*

POST Capture Evidence

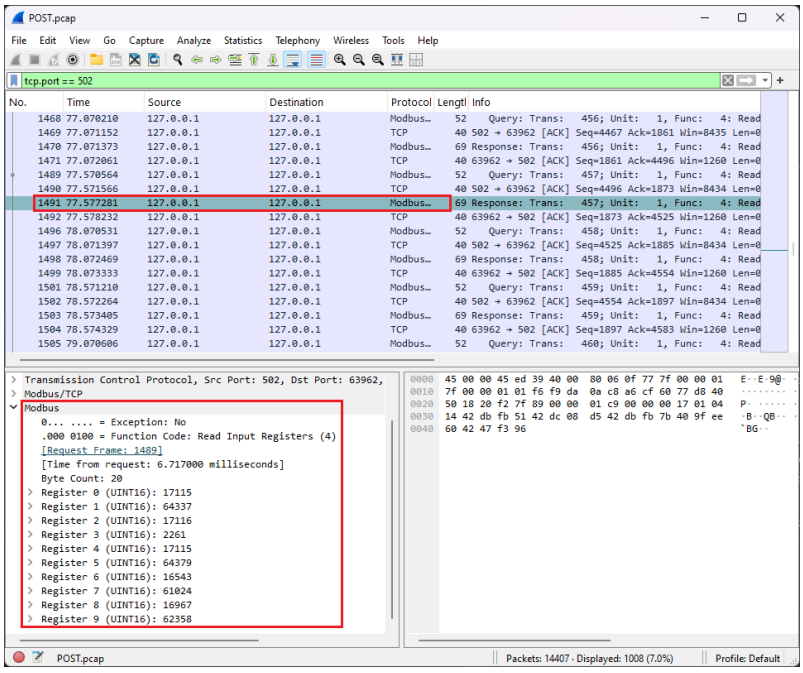


Figure 10-70: *POST capture for transaction ID 457. The filter shows 1008 forwarded Modbus/TCP packets, and the selected body matches MOD.*

10.3.10 Result

The example recorded 19 naturally varying FC04 response bodies while the Slave was in the Normal scenario. After Overvoltage was applied, the Slave and Master both showed approximately 124 V. During replay, the Slave remained near 124 V while the Master received the recorded normal signal near 110 V. Stopping replay restored the live Overvoltage value at the Master.

The run validates that NetSim Cyber replaces the body of a matching live response and forwards that packet. PRE and POST each contain 1008 Modbus/TCP packets, MOD contains 80 replaced FC04 responses and DROP contains no packets. Transaction ID 457 retains the same identifiers and body length while its Phase-A value changes from 124.0022 V in PRE to 109.9909 V in MOD and POST. The difference between the Slave and Master during replay demonstrates concealment of the current Overvoltage state without adding a replay packet.

10.3.11 Troubleshooting

Table 10-6: *Common checks.*

Observation	Check
Master cannot connect	Start the Slave first and verify `127.0.0.1`, port 502 and Unit ID 1.
Master reports an illegal address	Use FC04, start address 0 and count 10.
No values appear in the graph	Start polling and select a signal in Register Filter.
No packets are recorded	Start the NetSim run, reconnect the Master, keep polling enabled and select FC04 Input Registers in the Replay tab.
Replay does not start	Confirm that the buffer is ready and that the current response body has the same length as the recorded body.
The repeated pattern is too short	Record more responses without changing the 500 ms poll interval.
The graph looks flat after a scenario change	The automatic Y-axis includes both 110 V and 124 V, which hides the small variation. Pause and zoom if needed, or wait until the large transition leaves the 60-second graph window.
Live values do not resume	Select Stop Replay and confirm that Master polling continues.

10.4 IEEE C37.118-2005 Synchrophasor Monitoring and Defense

10.4.1 Overview

This example monitors an IEEE C37.118-2005 PMU stream carried over TCP. NetSim Cyber calls one context-aware Python callback for every packet selected by the Application Traffic Filter. The callback runs the installed synchrophasor attacker, inspects the resulting DATA frames, and returns one packet verdict.

The example combines three actions in a single callback:

- **Modify:** the installed attacker can change frequency, ROCOF, or phasor values and correct the C37.118 CRC.
- **Inspect:** the filter decodes the current payload and evaluates each station against the configured frequency, ROCOF, and voltage profile.
- **Drop:** after a condition qualifies, the callback returns `drop=True`; the complete IP packet is rejected and recorded in `DROP.pcap`.

Modified values that remain within the defense profile are forwarded as the current payload. The filter therefore demonstrates monitoring and conditional prevention, not an unconditional drop of every modified packet.

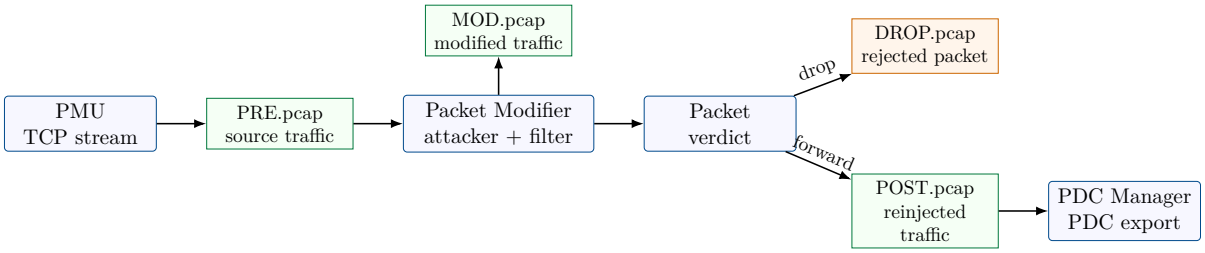


Figure 10-71: *Packet path and capture points. POST records reinjected traffic after Packet Modifier; the PDC export confirms application-level reception.*

10.4.2 Prerequisites

Item	Detail
NetSim Cyber	Version 15.1.11 or later, Cyber edition. See NetSim Cyber Manual, Section 9.3, for the Packet Modifier hook.
Simulators	IEEE C37.118-2005 PMU simulator and PDC Manager.
Attacker	Synchrophasor 2005 attacker, installed with NetSim Cyber.
Python	Interpreter bundled with NetSim Cyber.
Runtime discovery	The script locates a compatible NetSim Cyber installation automatically. `NETSIM_BIN_DIR` is an optional override.

10.4.3 The Unified Packet Verdict

`modify_payload()` returns a dictionary with three keys:

Key	Meaning
`modified`	`True` when the callback changed payload bytes.
`drop`	`True` when NetSim must reject the packet and record it in `DROP.pcap`.
`extra_delay_ms`	Additional delay applied to the packet, in milliseconds.

For a separate Boolean `filter_payload()` contract, `False` (do not forward) maps to a verdict with `drop=True`, and `True` (forward) maps to `drop=False`. A `modify_payload()` implementation must return the verdict dictionary. Returning a bare Boolean from this callback is not equivalent: the host interprets the value as the legacy `modified` result and does not receive a drop request.

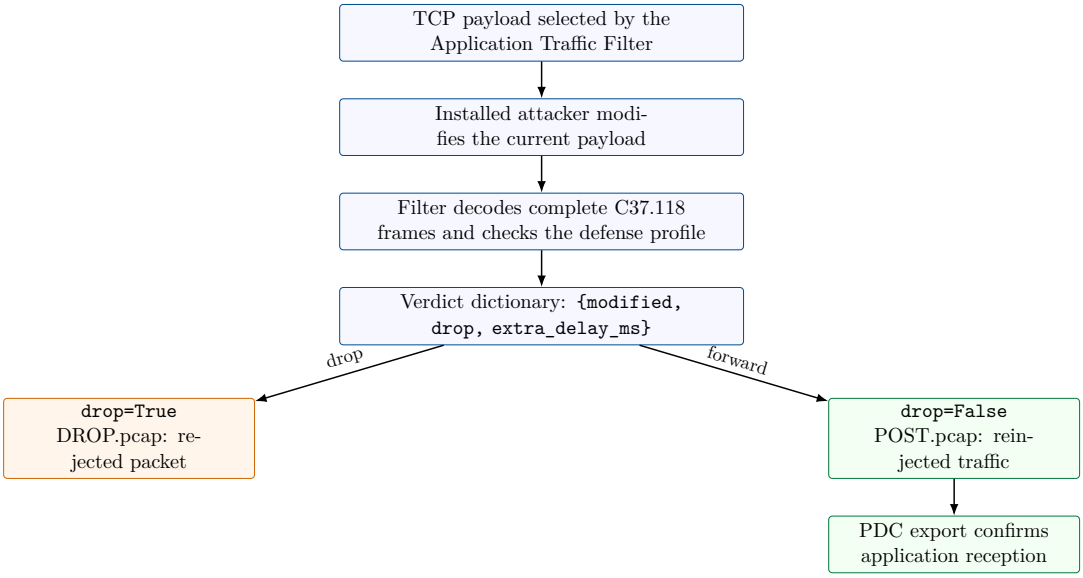


Figure 10-72: *One callback modifies, inspects, and returns a packet verdict. POST capture records reinjection; it is not by itself evidence that the PDC consumed the packet.*

The callback structure below is abbreviated from the supplied script; the transaction order matches the implementation.

```
NETSIM_PACKET_API_VERSION = 2

def modify_payload(ipv4, ipv6, tcp, udp, payload, iopath, context):
    original = bytes(payload) if payload is not None else None
    try:
        modified = bool(_attacker_modify_payload(
            ipv4, ipv6, tcp, udp, payload, iopath))
    except Exception:
        modified = False
        if payload is not None and original is not None:
            payload[:] = original

    if payload is None or not _is_target_pmu_flow(context):
        return _verdict(modified=modified)

    flow_key = _flow_key(context)
    frames = _extract_complete_frames(payload)
    staged_configs, reasons = {}, []
    baseline_commits, accepted_commits = [], []
    for frame in frames:
        frame_type = (frame[1] >> 4) & 0x07
        frame_id = int.from_bytes(frame[4:6], "big")
        if frame_type == 3:
            _stage_configuration(frame, flow_key, staged_configs)
        elif frame_type == 0:
            config_record = (
                staged_configs.get((flow_key, frame_id))
                or _config_records.get((flow_key, frame_id))
            )
            if config_record is None:
                continue
            found, pending, accepted = _evaluate_data_frame(
                frame, flow_key, config_record)
            reasons.extend(found)
            baseline_commits.extend(pending)
            accepted_commits.extend(accepted)
```

```
    if FILTER_ENABLED and reasons:
        return _verdict(modified=modified, drop=True)

    _commit_configurations(staged_configs)
    # Commit pending baselines and accepted references only on pass.
    return _verdict(modified=modified)
```

`NETSIM_PACKET_API_VERSION = 2` selects the seven-argument callback and supplies `context`, which identifies the TCP flow. Omitting the marker retains the legacy six-argument callback. An unsupported marker causes Packet Modifier initialization to fail instead of silently selecting another contract.

Drop and state messages are printed in `NetSimCyberConsole.exe`. Cumulative inspected-frame, trigger-specific drop, retry, and BYPASS counts are written to `SynchrophasorFilterSummary.txt` in the simulation log folder. The summary is refreshed at most once per second and forced to disk at a clean Python-host shutdown.

10.4.4 Example Setup

Component	Configuration
PMU/simulator	One IEEE C37.118-2005 station, PMU ID 7734, 50 Hz nominal frequency, 30 DATA frames/s, and CFG FORMAT `0x000E`: rectangular floating-point phasors with floating-point analog, frequency, and ROCOF fields.
PDC and transport	PDC Manager on loopback TCP; PMU source port 4712.
Packet Modifier	Select only `payload_modifier_synchrophasor2005_filter.py`; it invokes the installed synchrophasor attacker internally.
Packet captures	Enable PRE, POST, MOD, and DROP capture before starting the run.

PMU ID, nominal frequency, frame rate, and DATA format are PMU/simulator settings learned from CFG-2; they are not fixed by the defense script. The script supports both 50 Hz and 60 Hz CFG-2 nominal frequencies.

10.4.5 Processing Workflow

1. The PMU sends CFG-2. The callback stores a flow-scoped configuration containing station IDs, nominal frequencies, channel metadata, DATA format, and frame rate.
2. Every eligible TCP payload passes through the installed attacker first. If the attacker raises an exception, the original bytes are restored and still inspected.
3. Every complete C37.118 frame in the payload is inspected before one verdict is returned for the IP packet. CFG-2 changes are staged and committed only when that packet is forwarded.
4. State is independent for each TCP flow, configuration epoch, frame ID, and PMU station. C37 SOC/FRACSEC identifies DATA samples. Exact, changed, reordered, and replayed samples are inspected but do not advance persistence more than once.

Copyright TETCOS LLP

5. Frequency, undervoltage, and overvoltage durations are converted to sample counts from the CFG-2 DATA rate. ROCOF uses a timestamped window of at least 0.1 s and independently checks the time-weighted reported ROCOF and the rate derived from frequency change.
6. Voltage baselines use one second of accepted, finite, positive samples per voltage channel. Pending baseline samples and accepted frequency references are committed only when the packet is forwarded.
7. A qualifying frequency or ROCOF condition is latched. It remains active across TCP retries until 0.16 s of new, in-band, rate-safe samples has been observed.

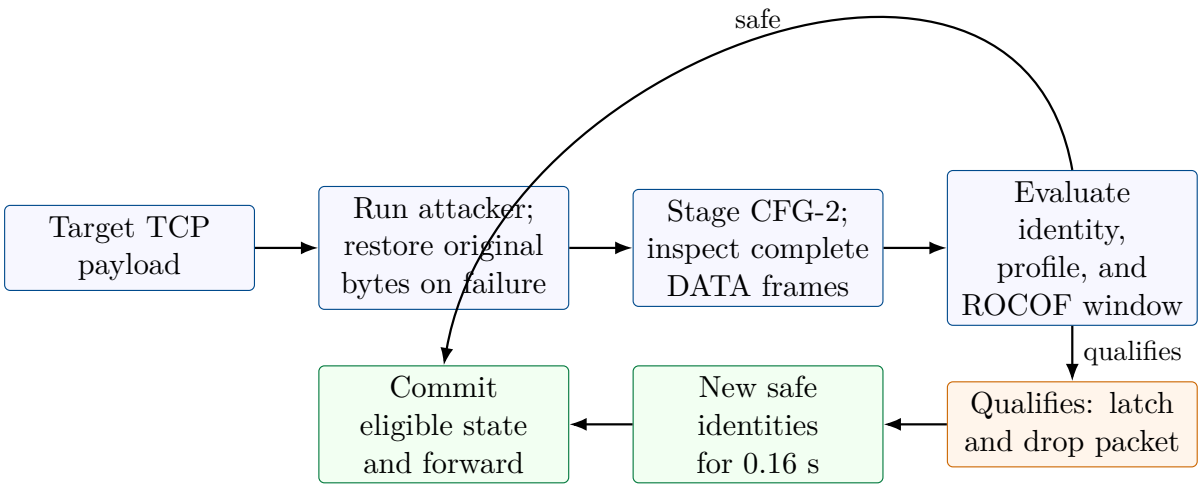


Figure 10-73: *Processing workflow. CFG-2 and accepted measurement state are committed only when the packet is forwarded; retries are inspected without advancing persistence.*

10.4.6 Detection Profile

Measurement	Example condition	Persistence
Frequency	Below 47.08 Hz or above 51.67 Hz for a 50 Hz CFG-2 nominal frequency	0.16 s (5 samples at 30 frames/s)
ROCOF	Absolute reported or frequency-derived average above 3.0 Hz/s	Timestamped window of at least 0.1 s per PMU
Undervoltage	Below 0.50 pu of the accepted baseline	2.0 s (60 samples at 30 frames/s)
Overvoltage	Above 1.20 pu of the accepted baseline	0.16 s (5 samples at 30 frames/s)

IEEE Std 1547-2018 Table 18 gives the 60 Hz UF2 and OF2 values used by this example: 56.5 Hz and 62.0 Hz, each with a 0.16 s clearing time. Section 1.4, footnote 4, states that frequency values for other nominal frequencies should be adjusted appropriately and notes proportional adjustment in coordination with the regional reliability coordinator.

For the 50 Hz demonstration, the script applies that proportional scaling: $56.5 \times 50/60 = 47.08$ Hz and $62.0 \times 50/60 = 51.67$ Hz. These are calculated example limits, not direct Table 18

entries or finalized settings for a specific utility. A PMU that declares 60 Hz in CFG-2 uses the direct 56.5–62.0 Hz values.

IEEE Std 1547-2018 Table 13 gives the voltage values used here: UV2 at 0.50 pu for 2.0 s and OV2 at 1.20 pu for 0.16 s. Table 21 gives the Category III ROCOF ride-through value of 3.0 Hz/s, and the standard requires ROCOF evaluation over an averaging window of at least 0.1 s. This example uses values outside the configured profile to request a packet drop; IEEE Std 1547 does not define synchrophasor packet-discard behavior.

These protection values cannot distinguish a genuine grid disturbance from injected data. The example demonstrates the filtering mechanism. A production detector should combine application-specific limits with protocol and consistency checks such as C37.118 STAT flags, timestamp continuity, CRC validation, and cross-measurement plausibility checks before requesting a packet drop.

10.4.7 Configuration Reference

The settings below appear at the top of the example script.

Setting	Default	Meaning
`FILTER_ENABLED`	True	Master switch for packet-drop decisions.
`REFERENCE_NOMINAL_FREQUENCY_HZ`	60.0 Hz	Reference used to scale the 60 Hz limits for another CFG-2 nominal frequency.
`MIN_FREQUENCY_HZ`	56.5 Hz	Direct lower limit for a 60 Hz PMU; 47.08 Hz after 50 Hz scaling.
`MAX_FREQUENCY_HZ`	62.0 Hz	Direct upper limit for a 60 Hz PMU; 51.67 Hz after 50 Hz scaling.
`FAST_TRIP_SECONDS`	0.16 s	Out-of-band frequency duration before a trip.
`RECOVERY_CONFIRMATION_SECONDS`	0.16 s	Safe-sample duration required to release a frequency/ROCOF latch.
`MAX_ABS_ROCOF_HZ_PER_S`	3.0 Hz/s	Reported and frequency-derived ROCOF boundary.
`ROCOF_AVERAGING_WINDOW_SECONDS`	0.1 s	Minimum timestamped averaging window per PMU.
`MAGNITUDE_BASELINE_SECONDS`	1.0 s	Accepted samples used to learn each voltage-channel baseline.
`MIN_VOLTAGE_PU/MAX_VOLTAGE_PU`	0.50 / 1.20	Voltage limits relative to the learned baseline.
`UNDERVOLTAGE_TRIP_SECONDS`	2.0 s	Sustained undervoltage duration before a drop.
`OVERVOLTAGE_TRIP_SECONDS`	0.16 s	Sustained overvoltage duration before a drop.
`MAX_SAMPLE_GAP_PERIODS`	2.5	Maximum continuity gap in expected DATA periods; longer gaps reset transient persistence without releasing a latch.
`TARGET_SOURCE_PORT`	4712	PMU source port inspected by the filter.

10.4.8 Test Procedure

1. In Configure Wireshark, enable PRE, POST, MOD, and DROP captures.

2. Select `payload_modifier_synchrophasor2005_filter.py` as the only Packet Modifier script. Start NetSim Cyber and confirm four `AttackPcap: Opened` messages in the console.
3. Configure the PMU for PMU ID 7734, 50 Hz nominal frequency, and 30 DATA frames/s. Start the PMU and connect the PDC.
4. Wait for the console to report CFG-2 learning and `Voltage-magnitude baseline ready`. Confirm normal measurements in the PDC export before enabling an attack.
5. Apply one condition at a time and hold it for at least four seconds: 55 Hz frequency, 5 Hz/s reported ROCOF, VA at 120 kV for overvoltage, or VA at 30 kV for undervoltage.
6. Stop the attack and continue the run until a normal PDC sample confirms recovery.
7. Correlate the console log, `SynchrophasorFilterSummary.txt`, PRE/MOD/DROP/POST captures, and the PDC export using TCP sequence data and C37 SOC/FRACSEC.

The original console capture below explicitly records the CFG-2 nominal frequency, DATA rate, and accepted-sample voltage baseline from the 50 Hz run.

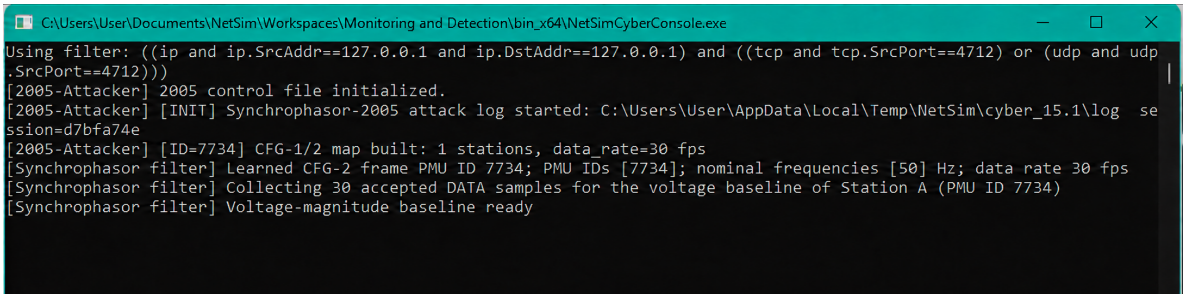


Figure 10-74: *Console showing CFG-2 nominal frequency 50 Hz, 30 frames/s, and voltage-baseline readiness.*

10.4.9 Expected Results

During normal operation, modified traffic is forwarded when the resulting measurements remain within the configured profile. When a measurement remains outside its limit for the required duration, the script requests a packet drop and prints the station, measurement, threshold, and persistence information in the console. Normal forwarding resumes after the attack is stopped and the safe-sample interval is satisfied.

Test	Drop trigger	Expected result
Frequency, 55 Hz	First qualifying detector; an abrupt override normally triggers frequency-derived ROCOF before the 0.16 s frequency condition	Console reports the detector and rejected packets are recorded in `DROP.pcap`
ROCOF, 5 Hz/s	Absolute value of either 0.1 s ROCOF average exceeds 3.0 Hz/s	Console reports a ROCOF drop and the PDC holds its last accepted value
Overvoltage, 120 kV	0.16 s above 1.20 pu (5 samples at 30 frames/s)	Initial persistence samples can be displayed; subsequent qualifying packets are dropped
Undervoltage, 30 kV	2.0 s below 0.50 pu (60 samples at 30 frames/s)	Initial persistence samples can be displayed; subsequent qualifying packets are dropped

The example PMU represents a 132 kV line-to-line system, giving a nominal phase voltage of approximately $132/\sqrt{3} = 76.2$ kV RMS. In the overvoltage demonstration below, the attacker changes VA to 120 kV, approximately 1.57 pu, which exceeds the configured 1.20 pu threshold.

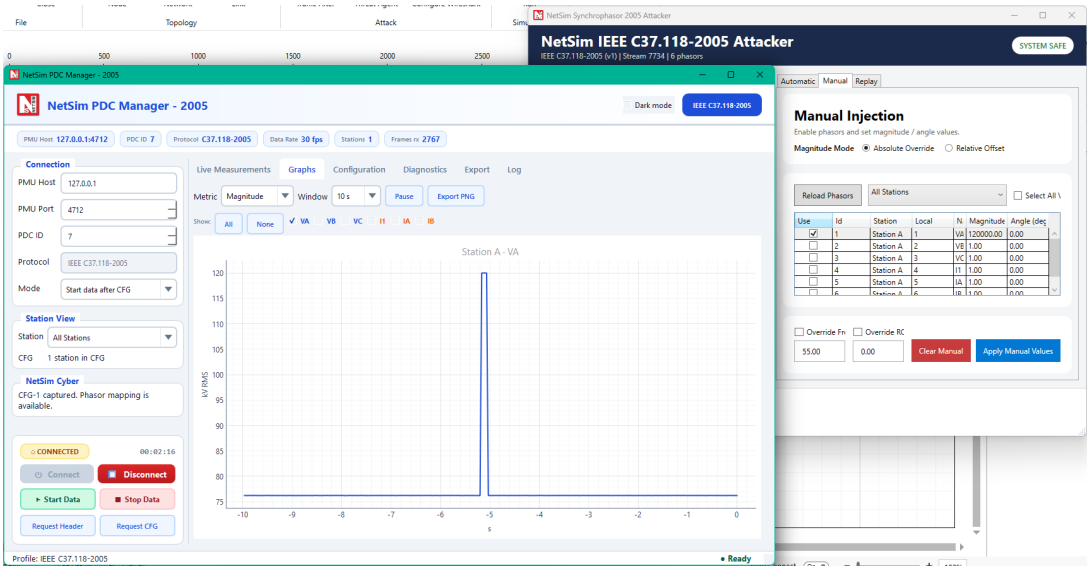


Figure 10-75: *PDC magnitude plot and active 120 kV VA override in the 50 Hz example.*

10.4.10 Packet-Level Behavior

The verdict applies to the whole captured IP packet. A TCP payload can carry more than one C37.118 DATA frame, so a packet containing a qualifying frame is dropped as a whole. If one station qualifies inside a multi-station packet, the complete packet is rejected.

TCP retransmissions are inspected but do not advance persistence more than once for the same C37 SOC/FRACSEC identity. A frequency or ROCOF condition remains latched across retries until 0.16 s of new, safe samples confirms recovery. The PDC may therefore pause on its last accepted value during sustained filtering.

10.4.11 Error Handling

Condition	Handling	Packet result
Missing CFG-2 or incomplete frame	Reported and counted; this packet-level example does not reassemble across callbacks	Forwarded unless another complete frame qualifies
Invalid measurement or decode failure	The reason is recorded in the console and summary	Dropped
Attacker exception	Original payload is restored and inspected	Determined from the restored payload
Changed CFG-2	A new configuration epoch is prepared	Committed only if the packet is forwarded

10.4.12 Summary

This example combines modification, inspection, and conditional drop for 50/60 Hz synchrophasor traffic.

11 Limitations

- Cyber-attack detection algorithms are currently not supported and are under development.
- Cyber-attack counter measures are currently not supported and are under development.

12 References

Schenato, L., Franceschetti, M., & Sastry, S. S. (2007). Foundations of Control and Estimation Over Lossy Networks. *Proceedings of the IEEE, 95*(1).