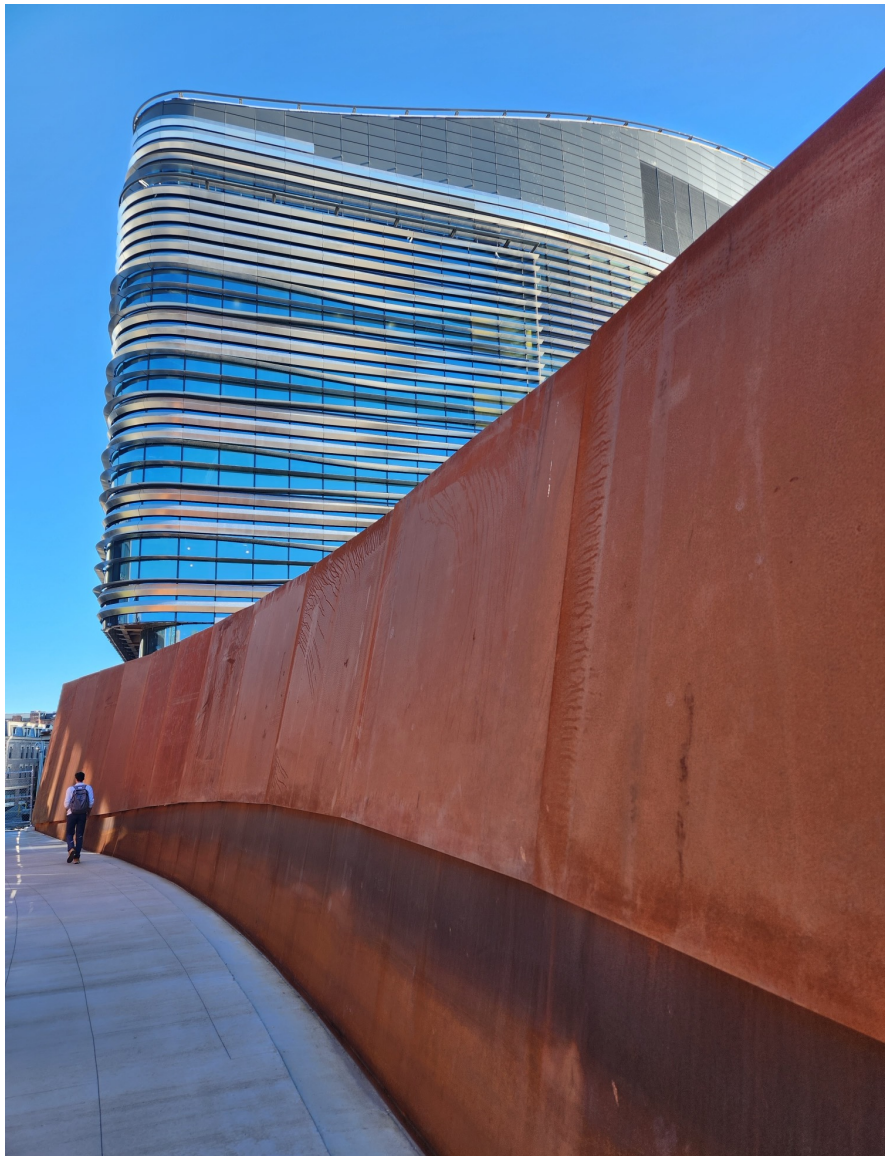
From Natural Language to Verified Systems: Uncovering Protocol Vulnerabilities with Formal Methods

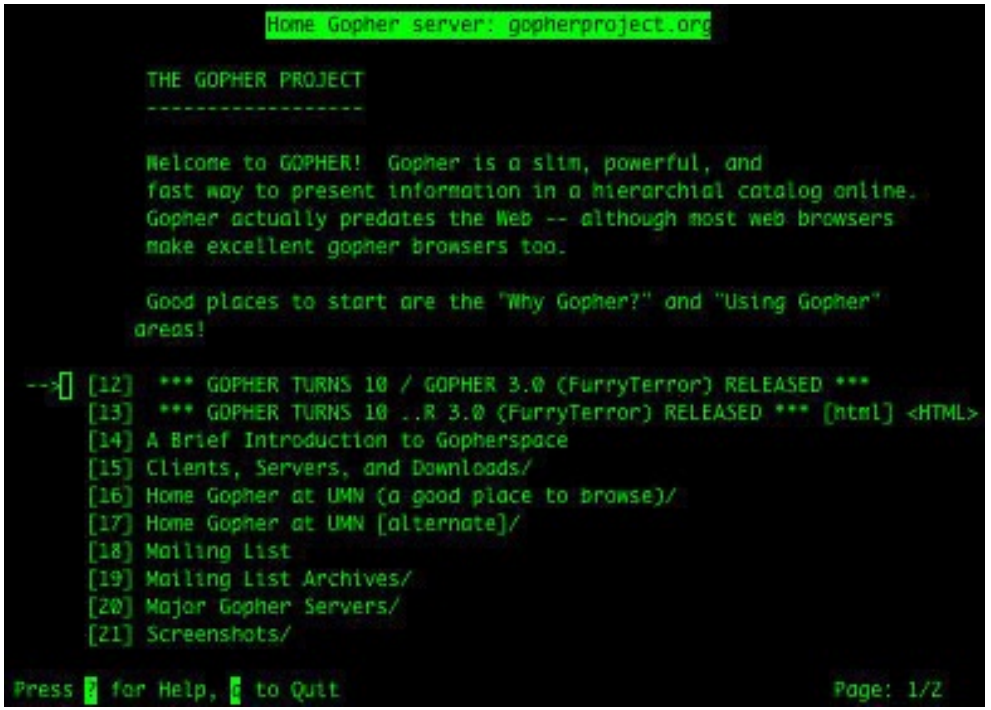
Cristina Nita-Rotaru, Professor

Khoury College of Computer Science
Northeastern University

SecDev 2025



How I met the … Internet



and went to graduate school to study it

- PhD studying security of distributed systems
- Designed and built secure group communication systems

- **Routing**: multi-hop communication
- **Transport protocols**: TCP, TLS, QUIC
- **Coordination:** fault-tolerant, state machine replication services (Paxos, PBFT)
- **Scalability:** overlays**,** distributed ledgers

What are network protocols?

Define how two (or more parties) exchange information
- Parties can be devices or applications
- Reliability, security, congestion control, fairness

Standardized by different organizations
- IETF
- IEEE
- 3GPP
- Bluetooth SIG … and more

Examples:

Cellular:	LTE
	5G
Wireless:	802.11
	ZigBee
	Bluetooth
Routing:	BGP
	OSPF, RIP
Transport:	TCP,
	UDP
Security:	TLS
	QUIC
Web:	HTTP/HTTPS

How do we know they work correctly?

Many of them were not designed with adversaries in mind

Do these protocols work as intended?

- Faults or misconfigurations

Testing

- Unit, stress

They now have to operate correctly in the presence of adversaries

Examples of attacks:

TCP: SYN Flooding

BGP: black-hole

DNS: Kaminsky attack

Overlays: Eclipse

Blockchains: 51% double-spending

Adversarial testing

State machine replication:
- Turret [ICDCS 2013]; Gatling [NDSS 2012]

Wireless routing:
- Turret-W [WiSec 2012, TON 2016]

TCP:
- Snake[DSN 2015], TCPwn [NDSS 2018], aBBR [RAID 2020]

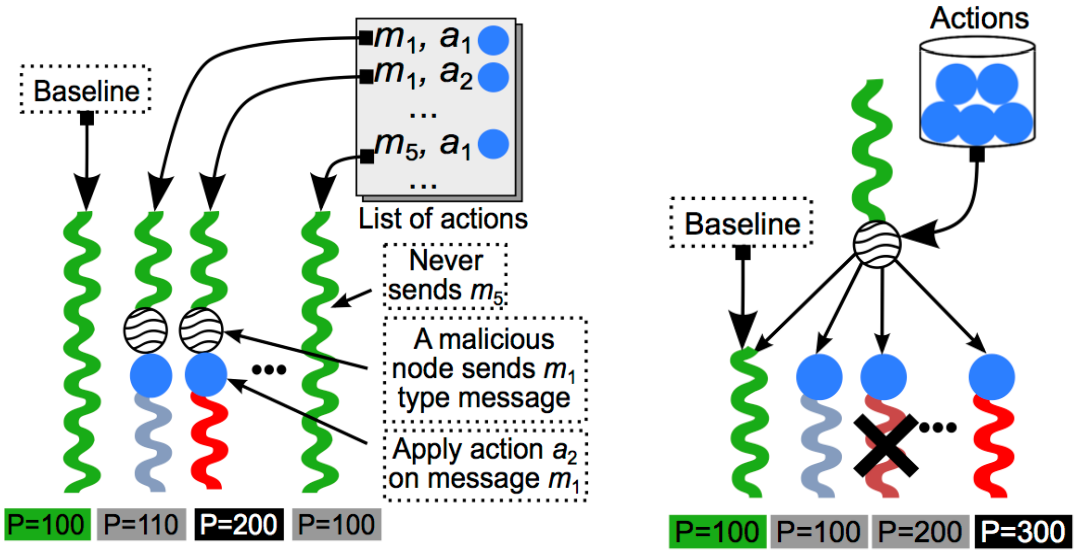
OpenFlow:
- BEADS [RAID 2017]

IoT:
- CHIRON [DSN 2017]

X509 certificates:
- SymCert [S&P 2017]

Secure protocols:
- QUIC [S&P2015], OpenVPN [2024]

Greedy search in Turret

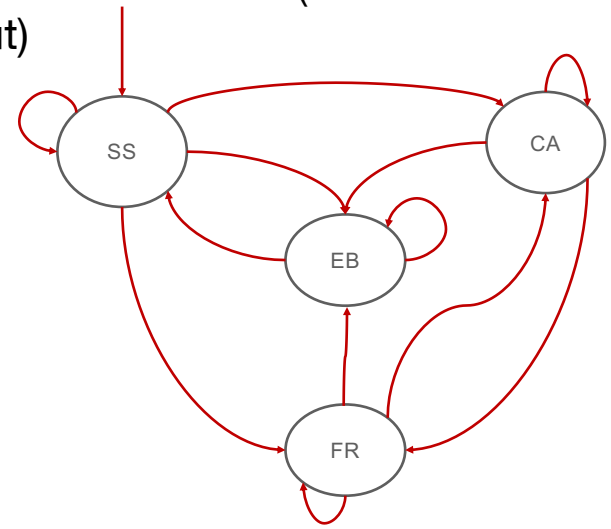


Model-based adversarial testing

Use abstract model to generate abstract strategies

Map abstract strategies to concrete strategies

Execute concrete strategies on implementations to find performance attacks (decrease/increase/stall throughput)



Attack Class	Attacker	Impact	OS	New?
Optimistic Ack	On-path	Increased Throughput	ALL	No
On-path Repeated Slow Start	On-path	Increased Throughput	Ubuntu 11.10, Ubuntu 16.10	Yes
Amplified Bursts	On-path	Increased Throughput	Ubuntu 11.10	Yes
Desync Attack	Off-path	Connection Stall	ALL	No
Ack Storm Attack	Off-path	Connection Stall	Debian 2, Windows 8.1	No
Ack Lost Data	Off-path	Connection Stall	ALL	Yes
Slow Injected Acks	Off-path	Decreased Throughput	Ubuntu 11.10	Yes
Sawtooth Ack	Off-path	Decreased Throughput	Ubuntu 11.10, Ubuntu 14.04, Ubuntu 16.10, Windows 8.1	Yes
Dup Ack Injection	Off-path	Decreased Throughput	Debian 2, Windows 8.1	Yes
Ack Amplification	Off-path	Increased Throughput	Ubuntu 11.10, Ubuntu 14.04, Ubuntu 16.10, Windows 8.1	Yes
Off-path Repeated Slow Start	Off-path	Increased Throughput	Ubuntu 11.10	Yes

Automated Attack Discovery in TCP Congestion Control Using a Model-guided Approach. S, Jero, E. Hoque, D. Choffnes, A. Mislove, C. Nita-Rotaru. NDSS 2018, **CISCO Network Security Distinguished Paper Award.**

Formal methods can help

Rigorous specifications can
- Disambiguate systems specifications
- Make implicit assumptions explicit

Formal analysis can
- Provide mathematical proofs
- Find counterexamples that expose flaws

Examples:

CHORD

TLS 1.3, QUIC

SDN

5G Authentication

RAFT, PAXOS

RESEARCH AND ADVANCES

Systems and Networking

It Takes a Village: Bridging the Gaps between Current and Formal Specifications for Protocols

The accelerated migration to advanced services will be accompanied by unprecedented complexity, and security and reliability concerns that must be addressed by the network-engineering and formal-methods communities.

By David Basin, Nate Foster, Kenneth L. McMillan, Kedar S. Namjoshi, Cristina Nita-Rotaru, Jonathan M. Smith, Pamela Zave, and Lenore D. Zuck

Posted Jul 23 2025

How to obtain rigorous specifications?

Specifications in natural language text

- Informal
- Inconsistent
- Incomplete

Reference implementations

- Differences between text and implementation
- Not enough (do not capture intention)

3.4. Establishing a connection

The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

RFC 793 (TCP)

implementation of a trade-off between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in figure 7 below. The figures should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (-->) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (<--), indicate the reverse. Ellipsis (...) indicates a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state AFTER

What tool to select to use for analysis?

- Current tools require substantial expertise to use, particularly for complex protocols.
- Most successful analysis efforts have been carried out using domain-specific tools and often by the developers of those tools.
- There is no single “point of entry” for understanding what tools are available, what inputs and expertise are required to use them, and what their capabilities and limitations are.

Tools for protocols:

ACL2, Alloy, Rocq, CryptoVerif, Dafny, Isabelle, Ivy, Proverif, Spin, TLA+, Tamarin, Verifpal, and more

In this talk

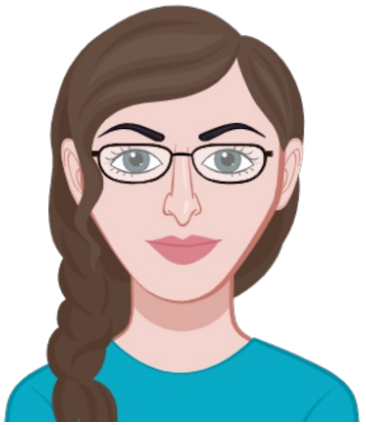
Show several examples of our work where we used formal methods to solve different problems related to protocol security, after the protocols were already designed

- Attack synthesis

- Patch verification

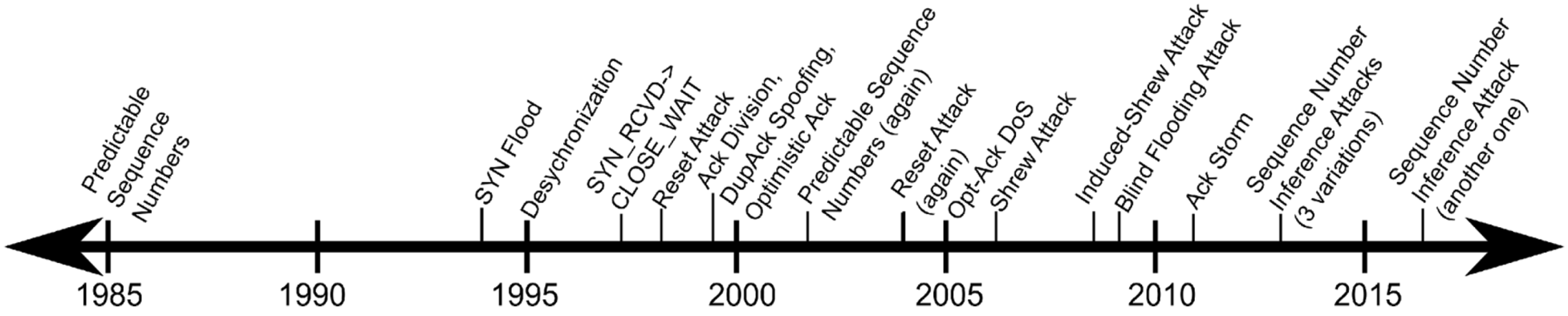
- Ambiguity resolution

- Security definition



Example: TCP

Transport protocol used by most of Internet traffic (including TLS, BGP)
Functionality described in over 20 RFCs, thousands of implementations
Provides reliability, in-order delivery, flow control, congestion control
… attacked for 40 years



As of 2025 there are still new attacks found against TCP

How were these attacks found?

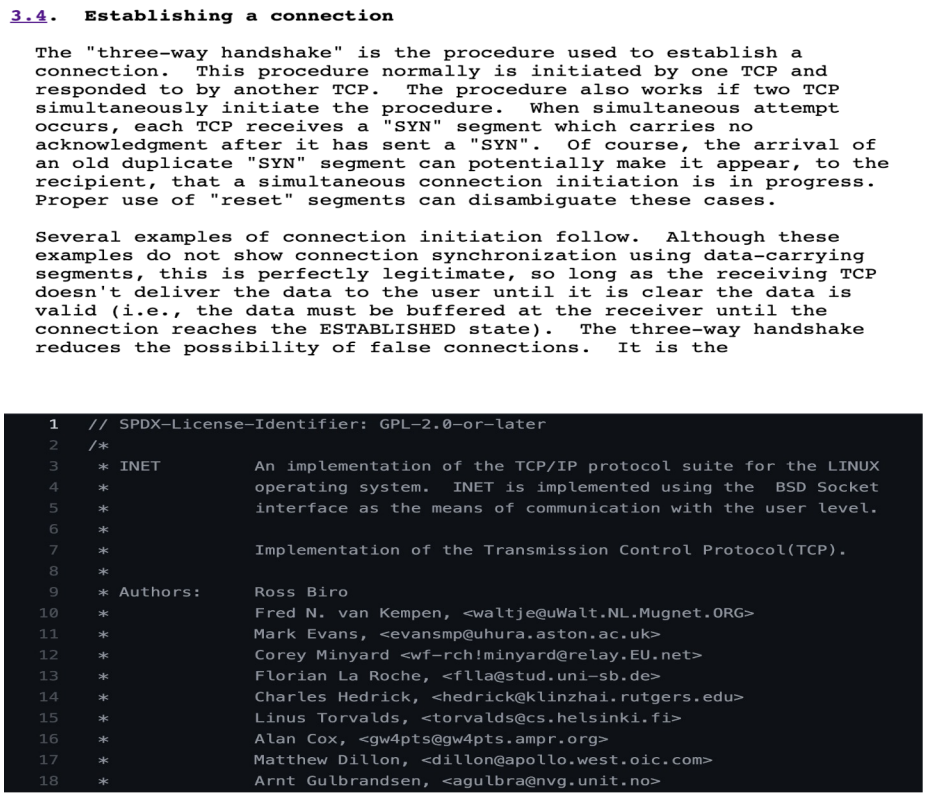
Manually

- Reading specification
- Observing behavior

Automatically

- Fuzzing
- Model-based testing

Can we prove that there are no other attacks?



3.4. Establishing a connection

The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

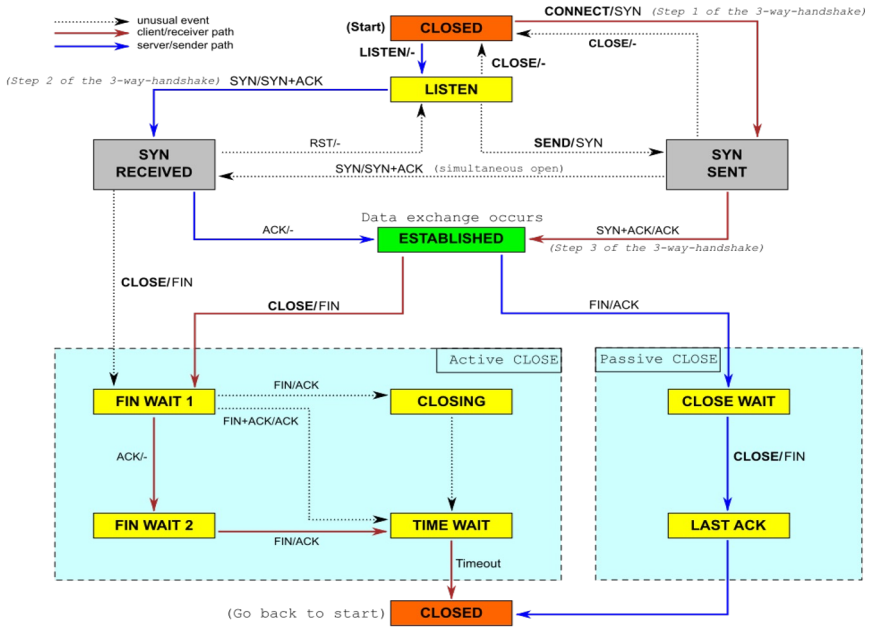
Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

```
1  // SPDX-License-Identifier: GPL-2.0-or-later
2  /*
3   * INET         An implementation of the TCP/IP protocol suite for the LINUX
4   *              operating system.  INET is implemented using the  BSD Socket
5   *              interface as the means of communication with the user level.
6   *
7   *              Implementation of the Transmission Control Protocol(TCP).
8   *
9   * Authors:     Ross Biro
10  *              Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
11  *              Mark Evans, <evansmp@uhura.aston.ac.uk>
12  *              Corey Minyard <wf-rch!minyard@relay.EU.net>
13  *              Florian La Roche, <flla@stud.uni-sb.de>
14  *              Charles Hedrick, <hedrick@klinzhai.rutgers.edu>
15  *              Linus Torvalds, <torvalds@cs.helsinki.fi>
16  *              Alan Cox, <gw4pts@gw4pts.ampr.org>
17  *              Matthew Dillon, <dillon@apollo.west.oic.com>
18  *              Arnt Gulbrandsen, <agulbra@nvg.unit.no>
```

Problem definition

Given

- Protocol model
- Properties the protocol must meet
- Attacker model(s)



QUESTION:

Can we prove that there are no attacks breaking those properties?

Our approach: Attack synthesis

An attack is a counterexample violating some (security) property

Define an attacker as a process that when composed with target system, results in a protocol property violation

Look for counterexamples on the composed system

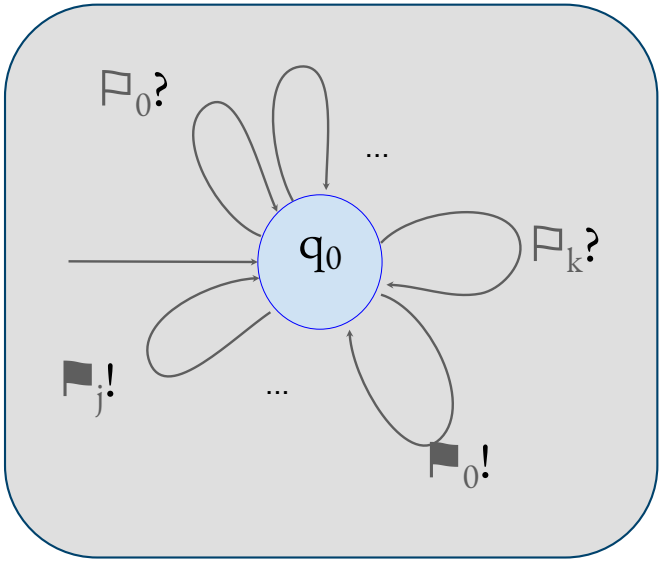
Attacker models

Off-Path: knows only IPs & ports

On-Path: can read and send any packet

Replay: can replay seen packets

Evil-Server: can send verified packets



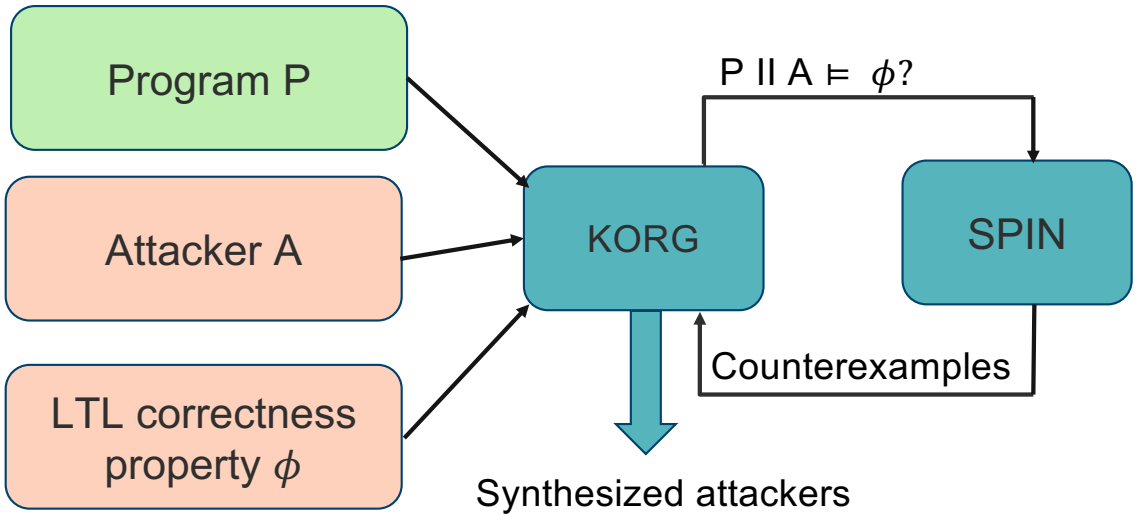
On-path attacker that nondeterministically exhausts the space of input and output events of a vulnerable process

KORG

KORG

- Implemented in Spin
- Program and the attacker are written in PROMELA
- Properties are written in LTL

For the systems we analyzed KORG never ran out of memory (either found attacks or fully exhausted the search-space).



Code available at
github.com/maxvonhippel/AttackerSynthesis

Case studies

TCP	DCCP	SCTP
Reliable in-order delivery.	Unreliable (maybe out of order) delivery.	Reliable in-order delivery.
Has passive/active and active/active establishment and teardown flows.	No active/active establishment and teardown flows.	Supports active/passive and active/active routines.
Establishment flow follows a 3-way handshake.	Establishment flow follows a 3-way handshake.	Increased security via 4-way establishment handshake.
One of the core protocols of the internet.	Good for streaming data, e.g., movies, video games.	Multihoming & multistreaming. Used by WebRTC.

Protocol Properties

TCP

ϕ1 =“No half-open connections.”

ϕ2 =“Passive/active establishment eventually succeeds.”

ϕ =“Peers don’t get stuck.”

ϕ4 =“SYN_RECEIVED is eventually followed by ESTABLISHED, FIN_WAIT_1, or CLOSED.”

DCCP

θ1 =“The peers don’t both loop into being stuck or infinitely looping.”

θ2 =“The peers are never both in TIME_WAIT.”

θ3 =“The first peer doesn’t loop into being stuck or infinitely looping.”

θ4 =“The peers are never both in CLOSE_REQ.”

Stream Control Transmission Protocol (SCTP)

Multi-streaming multi-homing

Reliable and resilient TCP alternative

4-way handshake

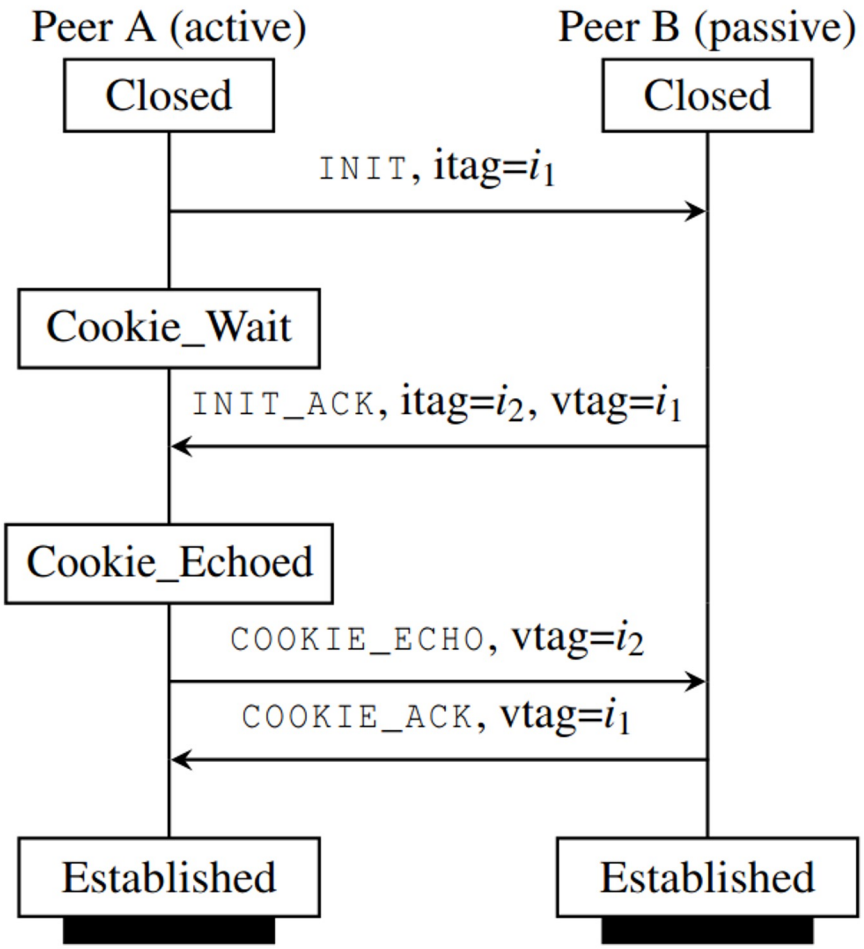
Telecoms

WebRTC Data Channel

Example: SCTP

4-way Handshake:

- Packets carry initialization tags (itag) & verification tags (vtag)
- itags are carried by INIT and INIT_ACK chunks
- vtags are carried by everything else
- Checks for **both** to ensure validity



SCTP

Connection teardown

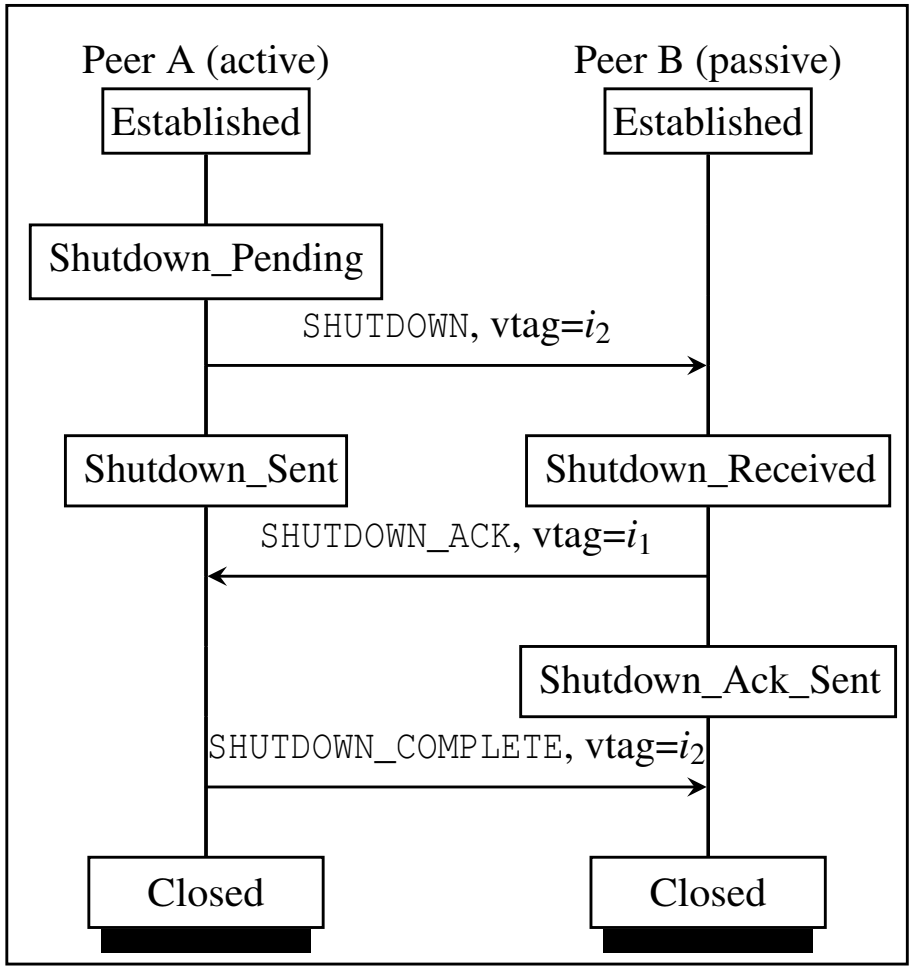
- Gracefully, via the active/passive or active/active routines

- Gracelessly, with an ABORT.

Two main implementations

- Conformance tests using PACKETDRILL

- Analysis with WIRESHARK



Example: SCTP

ϕ1: A peer in Closed either stays still or transitions to Established or Cookie_Wait.

ϕ2: One of the following always eventually happens: the peers are both in Closed, the peers are both in Established, or one of the peers changes state.

ϕ3: If a peer transitions out of Shutdown_Ack_Sent then it must transition into Closed.

ϕ4: If a peer is in Cookie_Echoed then its cookie timer is actively ticking.

ϕ5: The peers are never both in Shutdown_Received

ϕ6: If a peer transitions out of Shutdown_Received then it must transition into either Shutdown_Ack_Sent or Closed.

ϕ7: If Peer A is in Cookie_Echoed then B must not be in Shutdown_Received.

ϕ8: Suppose that in the last time-step, Peer A was in Closed and Peer B was in Established. Suppose neither user issued a User_Abort, and neither peer had a timer time out. Then if Peer A changed state, it must have changed to either Established, or the implicit, intermedi-ary state in Cookie_Wait in which it received **INIT_ACK** but did not yet transmit **COOKIE_ECHO**.

ϕ9: The same as ϕ8 but the roles are reversed. The property is: *Suppose that in the last time-step, Peer B was in Closed and Peer A was in Established...*

ϕ10: Once connection termination initiates, both peers eventually reach Closed.

Model	Prop.	Category
Off-Path	ϕ_9	Attack
	ϕ_8	Misinterpretation
	ϕ_4	Misinterpretation
Evil-Server	ϕ_1	Misuse
	ϕ_6	Misuse
	ϕ_8	Misuse
	ϕ_9	Misuse
Replay	ϕ_2	Misuse
On-Path	ϕ_5	Misuse
	ϕ_8	Misuse
	ϕ_9	Misuse

Main findings

Found various attacks against TCP, DCCP, TCP in which an adversary guides a peer through establishment to desynchronize it from the other peer.

Found attacks in SCTP & DCCP where the adversary guides both peers into passive shutdown.

Most attacks computed in seconds or minutes.

Code available at
github.com/maxvonhippel/AttackerSynthesis

In this talk

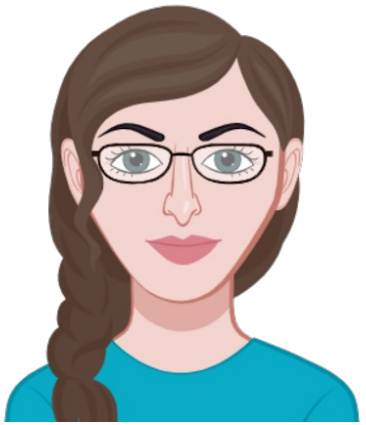
Show several examples of our work where we used formal methods to solve different problems related to protocol security

- Attack synthesis

- Patch verification

- Ambiguity resolution

- Security definition



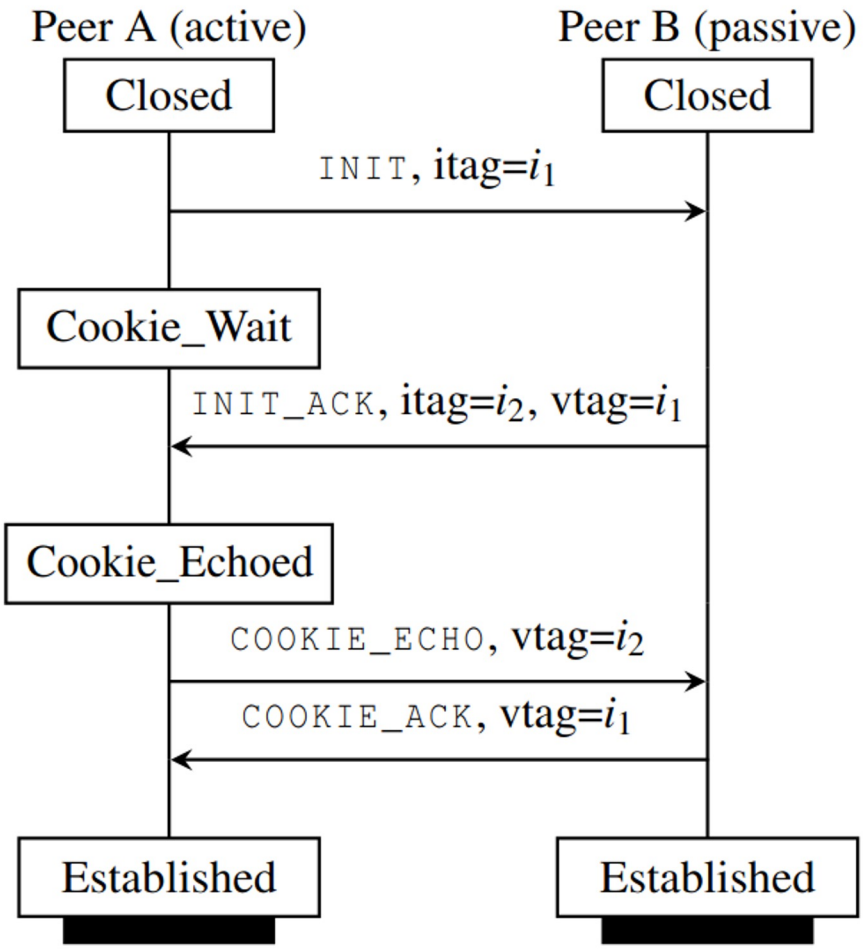
Example: SCTP

4-way Handshake:

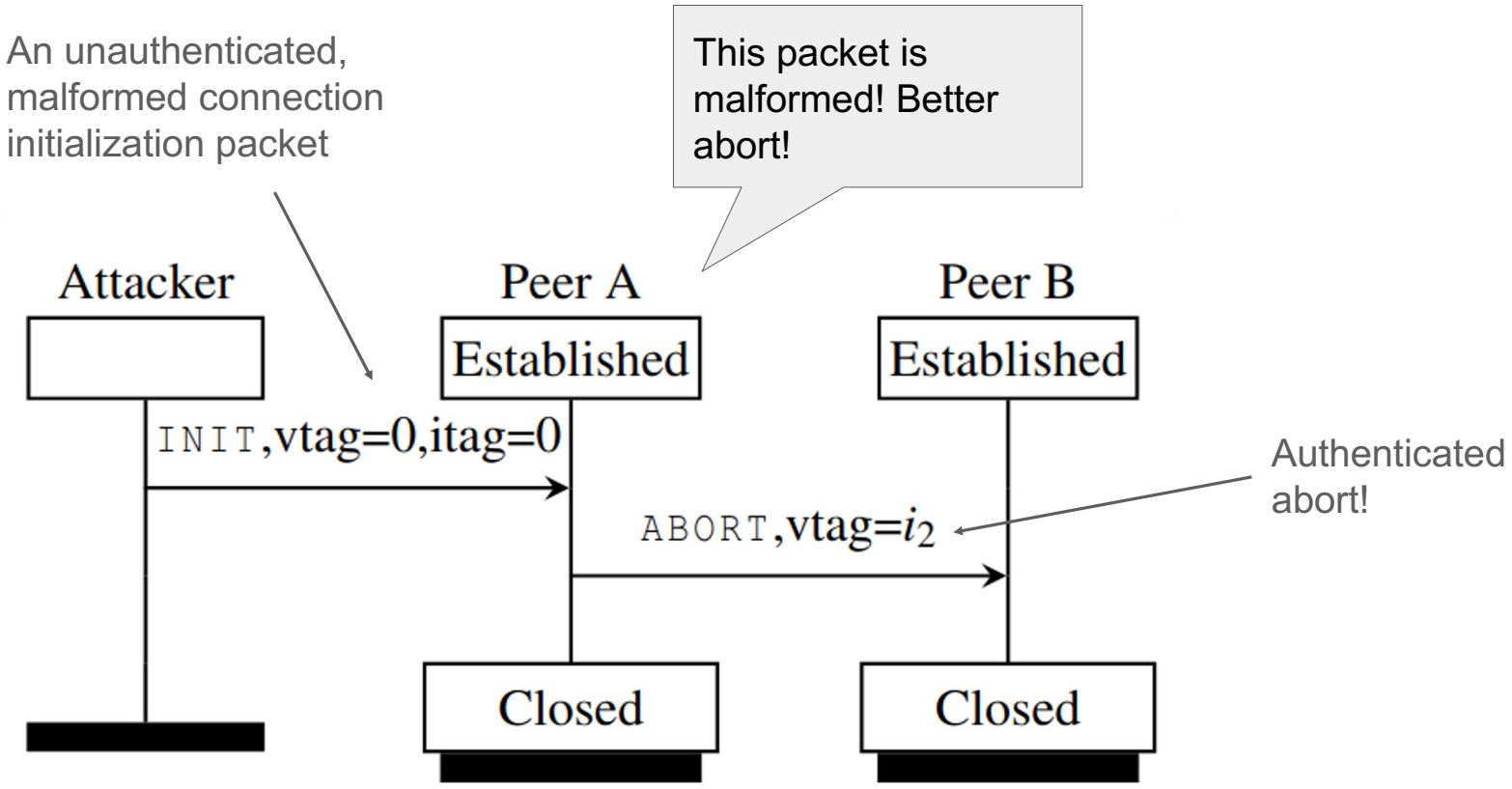
- Packets carry initialization tags (itag) & verification tags (vtag)
- itags are carried by INIT and INIT_ACK chunks
- vtags are carried by everything else
- Checks for **both** to ensure validity

Two main implementations

- Conformance tests using PACKETDRILL
- Analysis with WIRESHARK



CVE-2021-3772: kill any established SCTP connection (also found in minutes with KORG)



What did the attack happen?

Cause was a composition of events

An **unauthenticated** message was processed and not discarded

Resulting abort was **authenticated** when the message triggering it **was not**

Merge branch 'sctp-enhancements-for-the-verification-tag'

```
Xin Long says:

====================
sctp: enhancements for the verification tag

This patchset is to address CVE-2021-3772:

  A flaw was found in the Linux SCTP stack. A blind attacker may be able to
  kill an existing SCTP association through invalid chunks if the attacker
  knows the IP-addresses and port numbers being used and the attacker can
  send packets with spoofed IP addresses.

This is caused by the missing VTAG verification for the received chunks
and the incorrect vtag for the ABORT used to reply to these invalid
chunks.
```

Patched protocols

Protocols are implemented based on specification

Later on, bugs are found, CVEs are issued

Changes are proposed through new standards

Changes are implemented

Patched… but are there more vulnerabilities?

Problem definition

Given

- Protocol model
- Properties the protocol must meet
- Attacker model
- Vulnerability
- Patch addressing the vulnerability

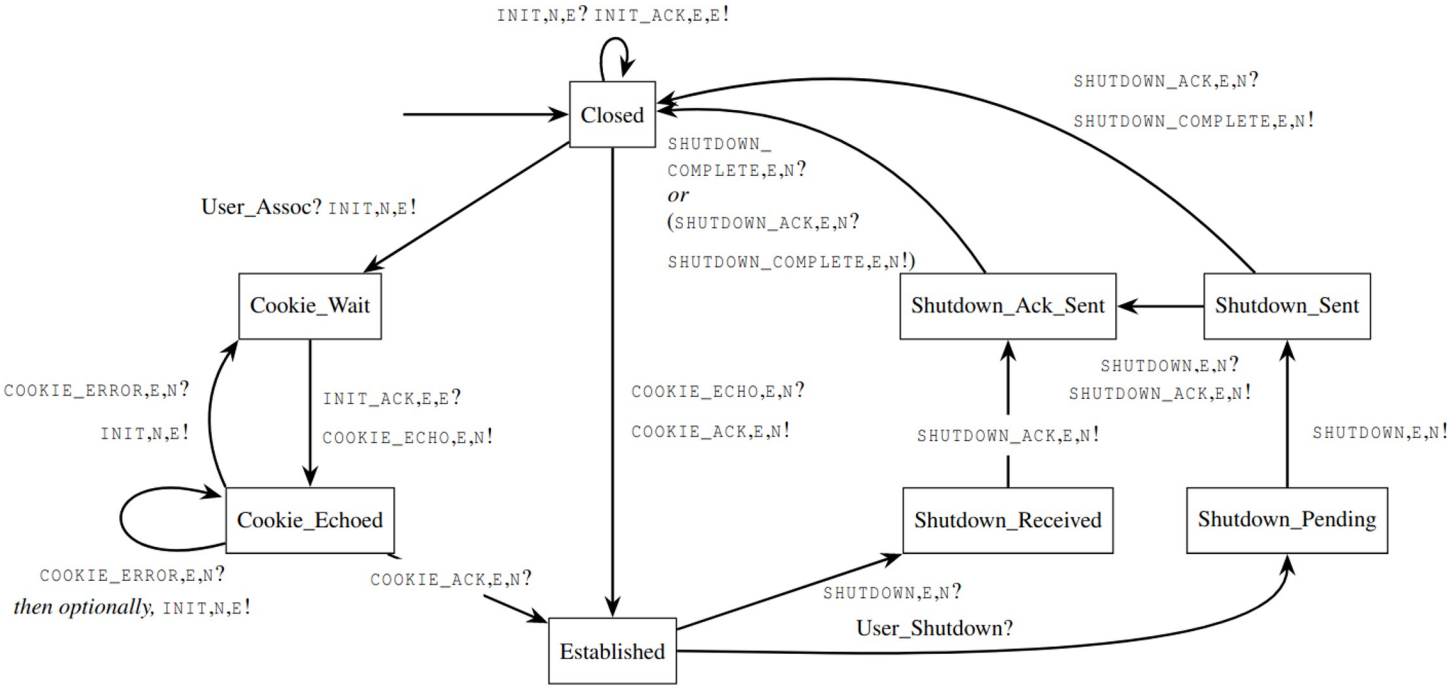
QUESTIONS:

Did the patch solve the vulnerability?

Did the patch introduce any new vulnerabilities?

Our approach

We construct models for both the original and patched system and apply KORG



SCTP model

We capture:

- All internal states
- All message handling, packet validation, and invalid packet defenses
- All connection establishment and teardown routines
- Unexpected packets

We *don't* capture:

- The *full* packet structure
- Data exchange
- Unicast peers
- Channel imperfections

Main findings

(CVE-2021-3772 automatically rediscovered)

Proved patch
- Fixed the problem
- Did not introduce new bugs

Caveat:
- With respect to the properties we considered
- It is possible that the patch could introduce an attack outside the scope of what we analyzed

Code available at github.com/sctpfm

In this talk

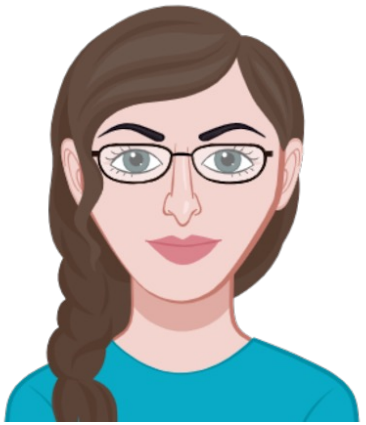
Show several examples of our work where we used formal methods to solve different problems related to protocol security

- Attack synthesis

- Patch verification

- Ambiguity resolution

- Security definition

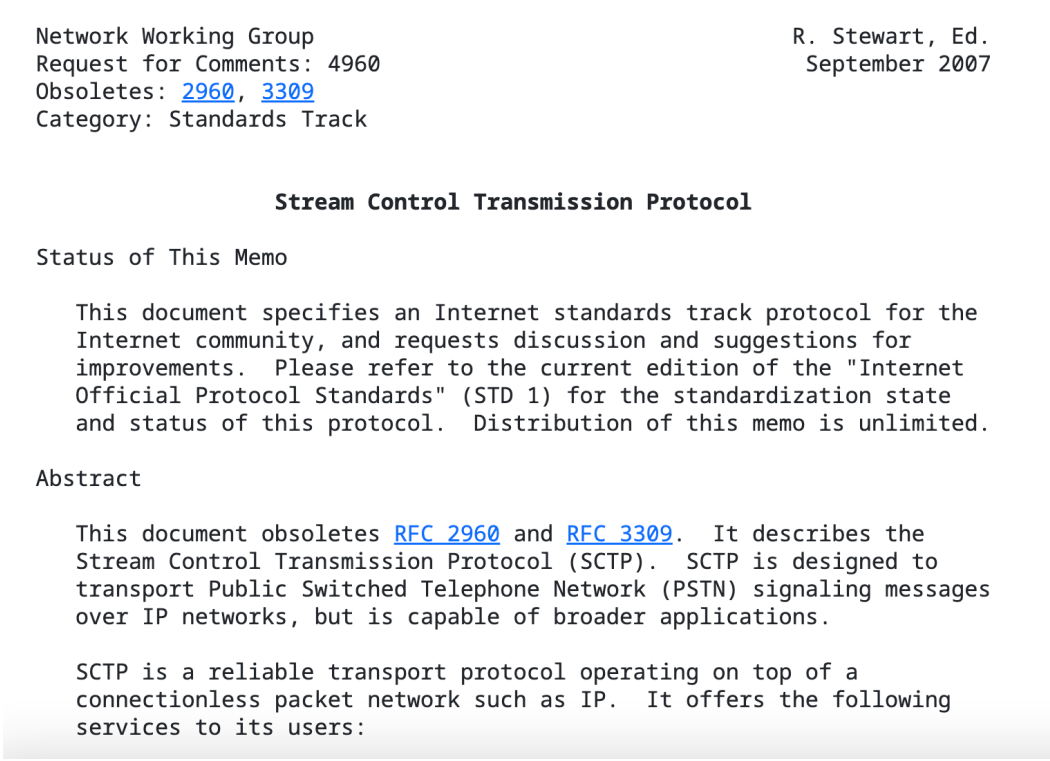


Specification of Internet Protocols

“RFC documents contain technical specifications and organizational notes for the Internet.”

Produced by IETF, describe the main Internet’s protocols such as addressing, routing, transport, or security

- Internet Standard
- Proposed Standard
- Best Current Practice
- Experimental
- Informational
- Historic



Network Working Group R. Stewart, Ed.
Request for Comments: 4960 September 2007
Obsoletes: 2960, 3309
Category: Standards Track

Stream Control Transmission Protocol

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

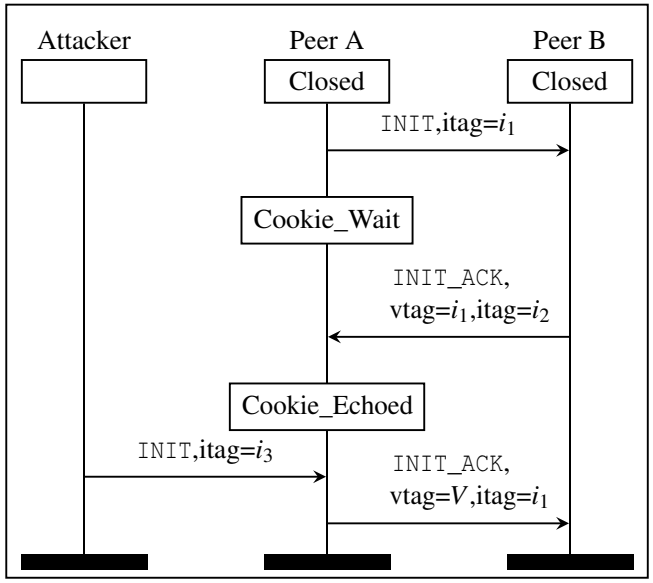
Abstract

This document obsoletes RFC 2960 and RFC 3309. It describes the Stream Control Transmission Protocol (SCTP). SCTP is designed to transport Public Switched Telephone Network (PSTN) signaling messages over IP networks, but is capable of broader applications.

SCTP is a reliable transport protocol operating on top of a connectionless packet network such as IP. It offers the following services to its users:

Example: SCTP RFC Ambiguity

"Upon receipt of an INIT chunk in the Cookie_Echoed state, an endpoint MUST respond with an INIT_ACK chunk using the same parameters it sent in its original INIT chunk (including its Initiate Tag, unchanged), provided that no new address has been added to the forming association."



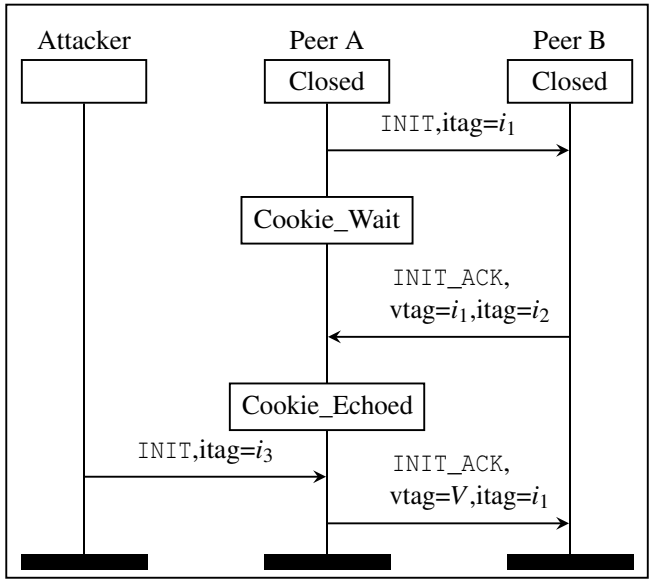
Misinterpretation:

What value should *V* take?

Example 2: SCTP RFC ambiguity

"When receiving an SCTP packet, the endpoint MUST ensure that the value in the Verification Tag field of the received SCTP packet matches its own tag."

Misinterpretation:
When the vtag check should happen with respect to other checks?



Problem definition

Given

- Protocol model
- Properties the protocol must meet
- Attacker model
- Ambiguity that can lead to several different protocol models

QUESTION:

Is the ambiguity dangerous?

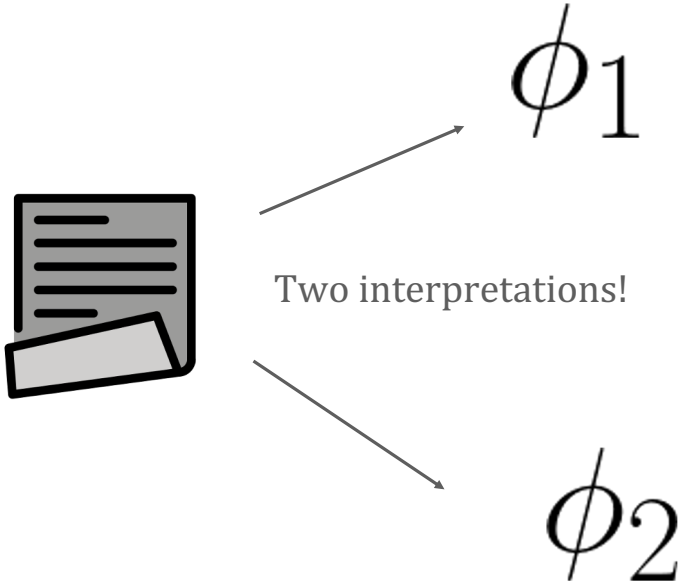
Our approach

Use KORG

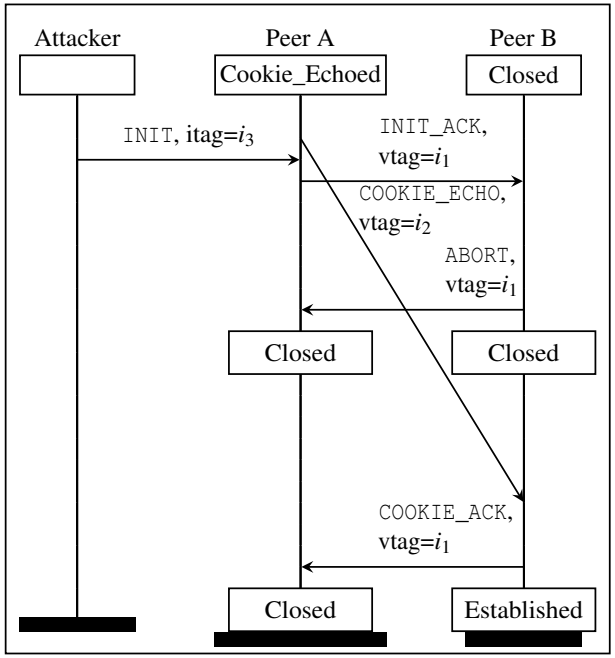
Modeled the protocol with the different interpretations of the ambiguity (2 in our case)

Show misinterpretations lead to attacks

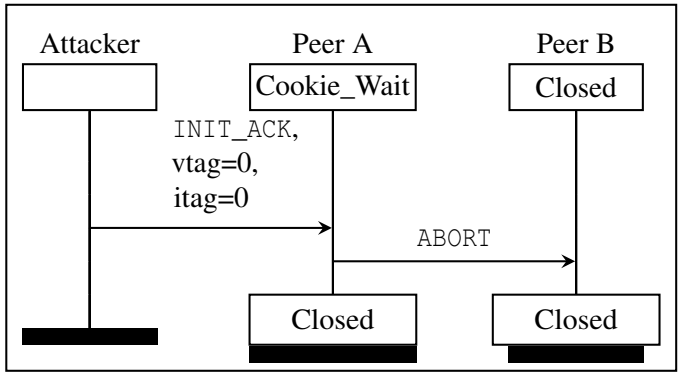
Code available at https://github.com/sctpfm



Main findings



Misinterpretations lead to attacks



Proposed errata

We proposed an errata for both ambiguities

One was approved

RFC 9260, "Stream Control Transmission Protocol", June 2022

Source of RFC: tsvwg (wit)

Errata ID: 7852
Status: Reported
Type: Technical
Publication Format(s) : TEXT
Reported By: Michael Tüxen
Date Reported: 2024-03-15

Section 8.5 says:

```
When receiving an SCTP packet, the endpoint MUST ensure that the
value in the Verification Tag field of the received SCTP packet
matches its own tag.
```

It should say:

```
When receiving an SCTP packet, the endpoint MUST first ensure that the
value in the Verification Tag field of the received SCTP packet
matches its own tag before processing any chunks or changing its state.
```

Notes:

State explicitly that the check of the verification tag needs to be done before any processing of the packet.

Thanks to Jake Ginesin, Max von Hippel, and Cristina Nita-Rotaru for reporting issue and discussing it with me.

In this talk

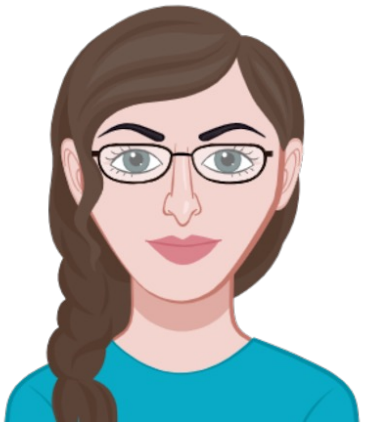
Show several examples of our work where we used formal methods to solve different problems related to protocol security

- Attack synthesis

- Patch verification

- Ambiguity resolution

- Security definition

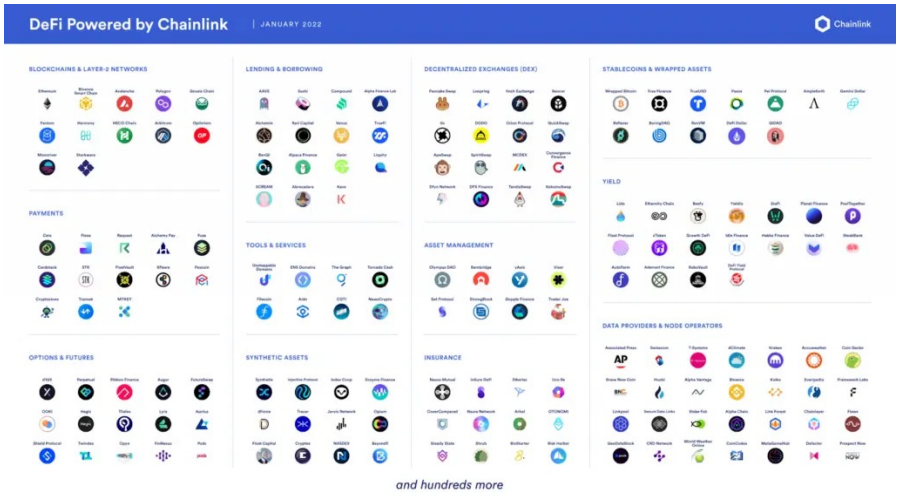


DeFi ecosystem

DeFi: Financial infrastructure as a non-custodial, permissionless, openly auditable, and highly interoperable protocol stack built on public smart contract platforms

Protocol stack

- Communication
- Settlement
- Asset
- Protocol
- Application
- Aggregation



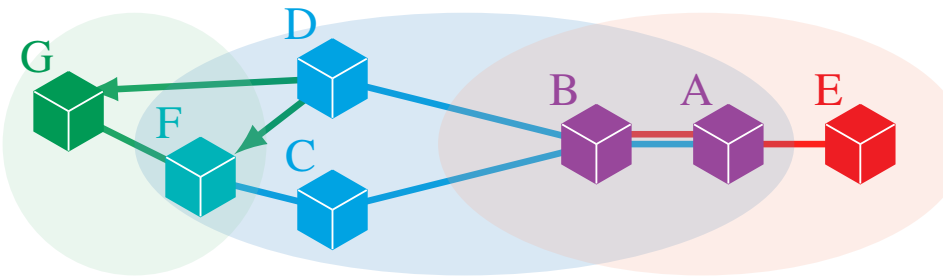
GossipSub

Pub/sub system: Builds efficient meshes to disseminate information based on topics

- It is used by Eth2.0, FileCoin and more

Specifications: Described in English and a reference implementation in Golang

Security: Designed to address malicious peers that drop messages, create churn, sybils



P2P Byzantine-resilient communication

Challenges:

- Identify malicious nodes (not always possible)
- Open systems - sybil attacks (same attacker with multiple identities)
- Attacker placement can prevent communication

Mitigations:

- Constrain peer placement (position in overlay decided by some invariant DHT)
- Flooding – best to guarantee eventual message delivery as long as there is a path from sender to receiver, but has high-overhead
- Avoid flooding with metrics, coding, separate meta-data from the data

GossipSub design

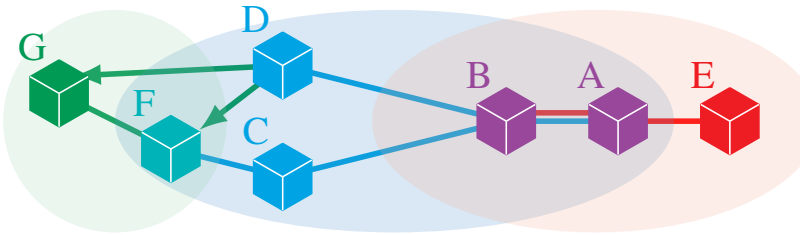
Creates meshes per-topic

Separates meta-data from message dissemination

- Metadata: push, periodically broadcast
- Messages: pull, requested by peers

Dealing with malicious nodes: “score function”

- Each peer computes a score about each of its neighbors.
- Score is used to remove (prune) and add (graft) peers



If peer A penalizes B for sending invalid messages, causing B’s score to become negative, then B will be pruned from A’s meshes.

Peer score

Captures good and bad behavior through global and per-topic indicators

Global threshold used to prune/graft peers

Configuration: Coefficients with which indicators are weighted in the score

$$score(q) = \mathrm{TC}\Big(\sum_{t \in p.T} \mathrm{tw}(t) \big(\sum_{i \in \{1,2,3,3b,4\}} w_i(t) P_i(t) \big) \Big) + \sum_{i=5}^{7} w_i P_i$$

What goes into the score computation?

Per-Topic Indicators

P1: Time in Mesh

P2: First Message Deliveries

P3: Mesh Message Delivery Rate

P3b: Mesh Message Delivery Failures

P4: Invalid Messages

Global Indicators

P5: Application-Specific Score. Score assigned to the peer by the application itself

P6: IP Colocation Factor. This indicator can be used to detect Sybils if the Sybils are IP co-located

P7: Behavioral Penalty. Keeps track on behavior for nodes that are regrafted

How do coefficients look like?

Ethereum 2.0

Constant or Weight	BLOCKS	AGG	SUB1	SUB2	SUB3
TopicWeight	0.8	0.5	0.33	0.33	0.33
TopicCap	32.72	32.72	32.72	32.72	32.72
$w_1(t)$	0.0324	0.0324	0.0324	0.0324	0.0324
$w_2(t)$	1	0.128	0.95	0.95	0.95
$w_3(t)$	-0.717	-0.064	-37.55	-37.55	-37.55
$w_{3b}(t)$	-0.717	-0.064	-37.55	-37.55	-37.55
$w_4(t)$	-140.45	-140.45	-4544	-4544	-4544
w_5 (global)	1	1	1	1	1
w_6 (global)	-35.11	-35.11	-35.11	-35.11	-35.11
w_7 (global)	-15.92	-15.92	-15.92	-15.92	-15.92
D	8	8	8	8	8

TABLE 7. ETH2.0'S TWP. ADAPTED FROM GITHUB.COM/SILESIACOIN/PRYSM-SPIKE.

FileCoin

Constant or Weight	MESSAGES	BLOCKS
TopicWeight	1	1
TopicCap	0	0
w_1(t)	2.78	0.027
$w_2(t)$	0.5	5
$w_3(t)$	0	0
$w_{3b}(t)$	0	0
$w_4(t)$	-1000	-1000
w_5 (global)	1	1
w_6 (global)	-100	-100
w_7 (global)	-10	-10
D	8	8

TABLE 6. FILECOIN'S TWP. ADAPTED FROM GITHUB.COM/FILECOIN-PROJECT/LOTUS.

How was it shown that “the score” works?

The Golang implementation was subjected to

- Unit tests
- Manual code review by expert programmers

Tested under concrete attacks and concrete deployments at scale using the TESTGROUND network emulator

How to select the configuration coefficients?

Are there other attacks?

Problem definition

Given

- GossipSub
- Configuration for an application
- Topology

QUESTION:

Does the score function upon which the defense mechanisms rely actually measure what it is intended to measure?

GossipSub specification states:

w1 (t) should be a “small positive”

w2 (t) and w5 should be “positive

w3 (t), w3b (t), w4 (t), w6 , and w7 should be “negative”

Time in mesh caps should be a “small positive value”

First message deliveries caps should be at least the corresponding mesh message deliveries thresholds;

IP colocation factor should be “at least 1”

Mesh message deliveries thresholds should be “positive” and depend on the “expected message rate for topic.” (dependency is not explained)

No guidance given for the topic weight tw(t)

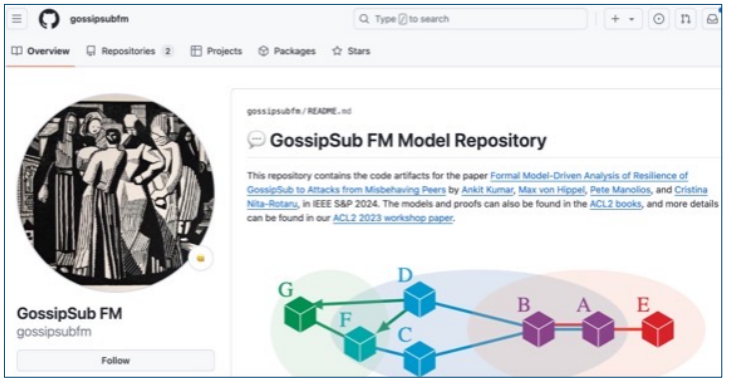
Our approach

We modeled GossipSub using ACL2, a theorem prover

We defined informally and formally what are the goals/security properties of the reputation score

We analyzed our model with the specific configurations for FileCoin and Eth2.0

We found one property violation in Eth2.0



Code available at github.com/gossipsubfm
Also included in the GossipSub code repository

Property of the defense mechanisms

Peers who behave poorly will be demoted by their neighbors. Peers who behave better-than-average will be promoted by their neighbors. Promotion/demotion is entirely based on peer behavior

(Broken into 4 properties we analyzed)

Property 1. *If a peer's score relating to its performance in any topic is continuously non-positive, then the peer's overall score should eventually be non-positive:*

$$\forall q, t :: \langle \mathbf{G}(score(q) \textit{ for topic } t \leq 0) \Rightarrow \mathbf{F}(score(q) \leq 0) \rangle$$

where score(q) for topic t is $\text{tw}(t)(\sum_{i \in \{1,2,3,3b,4\}} w_i(t) P_i(t))$.

Finding: Eth2.0 violates Prop. 1

A peer can offset a negative score in one of the several of subnet aggregators topics for a positive score all other topics combined, resulting in a positive overall score

Cause of violation is global threshold and not per-topic cap

Main findings

Attack gadget: (Attacker, Victim, Set of Topics)

- For a configuration, compute in how many topics t an attacker must be to create an attack

$$\min\{t \in \mathbb{N} \mid (7.2 + 3.2\frac{t}{T} > 24.7\frac{i}{T}) \wedge (t + i \leq T)\}$$

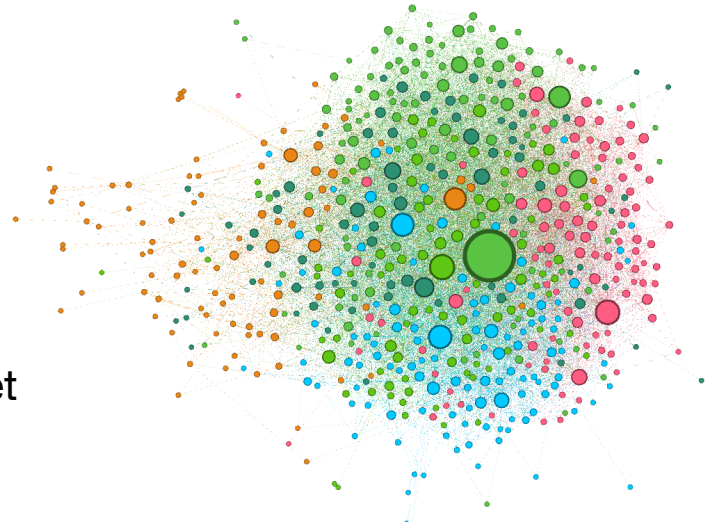
i topics under attack, T total number of topics

Throttle: A single attacker throttles a target topic for a victim without their score being decreased

Eclipse: Multiple attackers surround the victim and block target topics for it

Partition: Multiple attackers target multiple victims by blocking the target topics for each of them

Attacks Showed attack against Eth2.0 testnets Ropsten, Goerli, and Rinkeby, from TopoShot [IMC2021]



What can we say about configurations?

Our model allows developers to check if their configuration satisfies our security properties.

ACL2s may

- Prove the properties automatically, or

- Output counterexamples, or

- Fail -- the developer may need to guide the prover using supplemental lemmas, until the properties are disproven or proven

Constant or Weight	Pathological 1-5	Good 1	Good 2
`TopicWeight`	40	0.5	0.5
`TopicCap`	5	100	10
$w_1(t)$	10	0.027	0.027
$w_2(t)$	10	5	5
$w_3(t)$	-1	-1000	-1000
$w_{3b}(t)$	-1	-1000	-1000
$w_4(t)$	-1	-1000	-1000
w_5 (global)	10	1	1
w_6 (global)	-1	-100	-100
w_7 (global)	-1	10	10
`D`	5	8	8

TABLE 8. OUR PATHOLOGICAL `TWP` CONSISTS OF FIVE TOPICS ALL CONFIGURED PER COLUMN 2. OUR GOOD CONFIGURATION `TWP` CONSISTS OF TWO TOPICS, GIVEN IN COLUMNS 3 AND 4, AND SATISFIES ALL OUR PROPERTIES.

In this talk

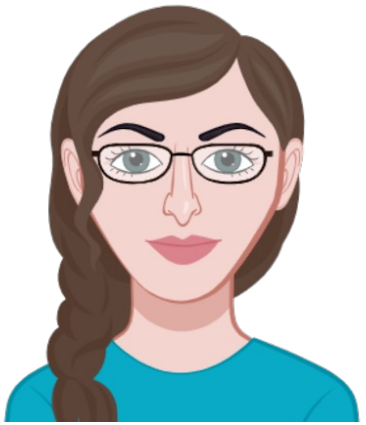
Show several examples of our work where we used formal methods to solve different problems related to protocol security

- Attack synthesis

- Patch verification

- Ambiguity resolution

- Security definition



Bonus: How to get FSMs from text

3.4. Establishing a connection

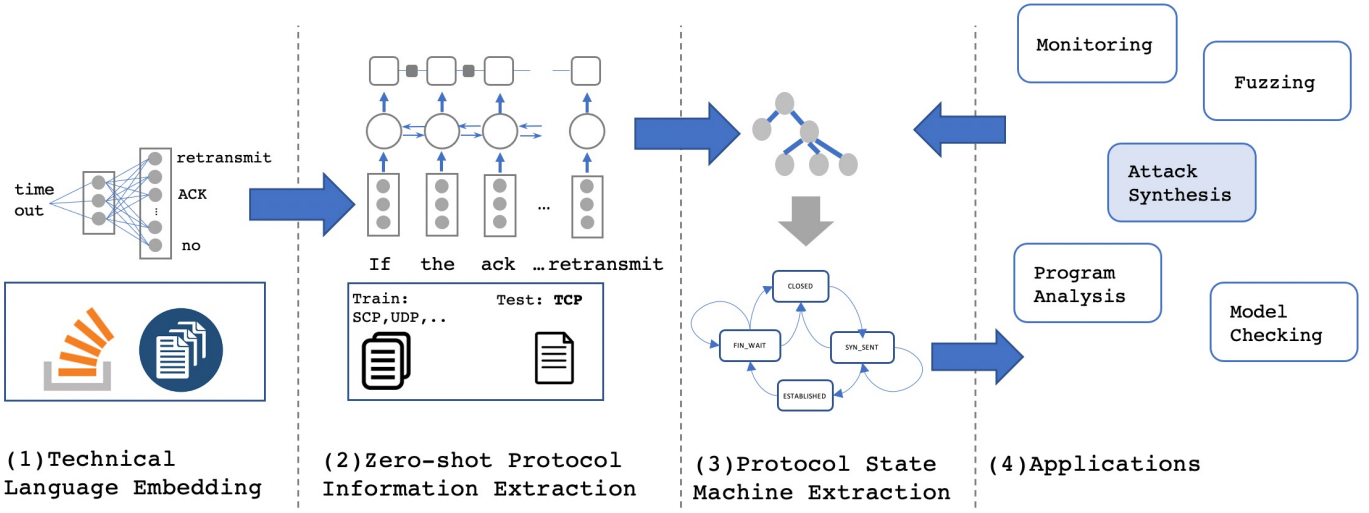
The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

[Page 30]

Our approach

1. Learn large-scale word-representation for technical language with off-the-shelf tools
2. Define and learn protocol-independent information language from RFC with focused zero-shot learning to adapt to new, unobserved protocols without re-training
3. Use rule-based mapping from protocol-independent information to a protocol FSM



Our general Protocol Grammar

- We use this grammar to define the protocol semantic we want to extract from RFCs
- Captures the semantic of a protocol FSM
- General enough that is applies to many protocols
- We use it to annotate the RFC
- Four types of annotation tags:
  - Definition tags
  - Reference tags
  - State machine tags
  - Control flow tags

```
bool      ::= true | false
type      ::= send | receive | issue
def-tag   ::= def_state | def_var | def_event
ref-state ::= ref_state id="##"
ref-event ::= ref_event id="##" type="type"
ref-tag   ::= ref-event | ref-state
def-atom  ::= <def-tag>engl</def-tag>
sm-atom   ::= <ref-tag>engl</ref-tag> | engl
sm-tag    ::= trigger | variable | error | timer
act-atom  ::= <arg>sm-atom</arg> | sm-atom
act-struct::= act-struct | act-struct act-atom
trn-arg   ::= arg_source | arg_target | arg_inter
trn-atom  ::= <trn-arg>sm-atom<trn-arg> | sm-atom
trn-struct::= trn-struct | trn-struct trn-atom
ctl-atom  ::= <sm-tag>sm-atom</sm-tag>
           | <action type="type">act-struct</action>
           | <transition>trn-struct</transition>
           | sm-atom
ctl-struct::= ctl-atom | ctl-struct ctl-atom
ctl-rel   ::= relevant=bool
control   ::= <control ctl-rel>ctl-struct</control>
e         ::= control | ctl-atom | def-atom
           |  e_0 e_1
```

FSM extraction: Transitions

- We extract all states

TCP FSM	Canonical	Extracted	Correct	Partially Correct	Incorrect	Not Found
Gold	20	18	8	8	2	4
LINEARCRF		28	2	3	23	15
LINEARCRF+R		30	7	10	13	3
NEURALCRF		11	2	3	6	15
NEURALCRF+R		30	7	10	13	3
DCCP FSM	**Canonical**	**Extracted**	**Correct**	**Partially Correct**	**Incorrect**	**Not Found**
Gold	34	24	15	1	8	18
LINEARCRF		8	1	5	2	28
LINEARCRF+R		17	6	3	8	25
NEURALCRF		20	9	1	10	24
NEURALCRF+R		19	8	3	8	23

Transitions extraction errors

FSM	Transition	Error Type	Reason	Text Excerpt
Gold TCP	FIN_WAIT_1 $\xrightarrow{\text{FIN!}}$ LAST_ACK	Not Found	Target state not explicit	CLOSE-WAIT STATE: Since the remote side has already sent FIN, RECEIVEs must be satisfied by text already on hand, but not yet delivered to the user.
Gold DCCP	PARTOPEN $\xrightarrow{\text{DCCP-CLOSE?}}$ OPEN	Incorrect	Text is ambiguous	The client leaves the PARTOPEN state for OPEN when it receives a valid packet other than DCCP-Response, DCCP-Reset, or DCCP-Sync from the server.
LINEARCRF+R and NEURALCRF+R	SYN_SENT $\xrightarrow{\text{SYN!ACK!}}$ SYN_RECEIVED	Partially Recovered (expected SYN?ACK!)	Receive action is not explicit	If the state is SYN-SENT then enter SYN-RECEIVED, form a SYN,ACK segment and send it.

Recent work is trying to use LLMs

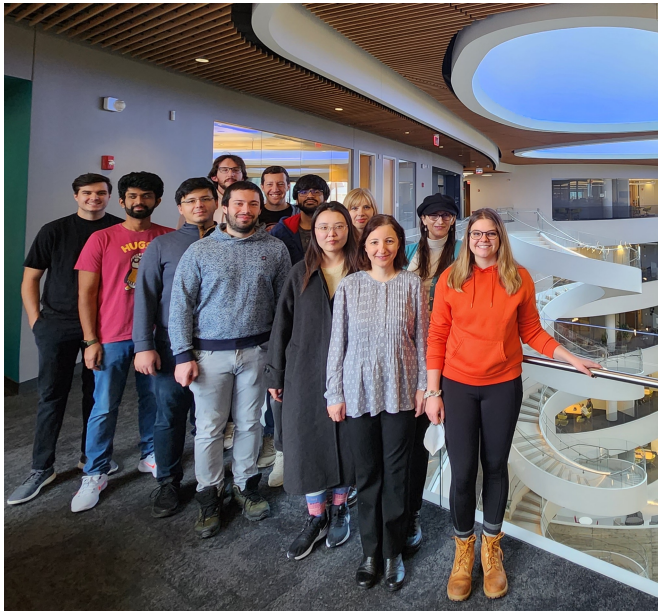
- To extract FSMs
- To create sequences of messages for fuzzers
- To learn format of messages for fuzzers
- To guide a fuzzer
- To create formal models, invariants

- Questions:
  - Does the observed improvement is indeed because of the LLM
  - How do we know that the output of the LLM is correct, some work reported hallucinated state machines and messages

Conclusion

Our experience with applying formal models to

- Attack synthesis
- Patch validation
- Ambiguity resolution
- Security definitions



NDS2 Lab

Impact:

- Our models endorsed by SCTP & GossipSub creators
- Applications of Formal Methods to Security of Network Protocols and Distributed Systems

https://cnitarot.github.io/courses/fmndss_Spring_2024/index.html

References

It Takes a Village: Bridging the Gaps between Current and Formal Specifications for Protocols
ACM Communications July 2025


David Basin
ETH

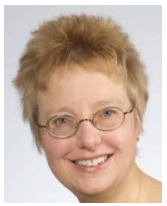

Nate Foster
Cornell University


Kenneth L McMillan
UT Austin


Kedar Namjoshi
Bell Labs


Cristina Nita-Rotaru
Northeastern University


Jonathan Smith
U Penn


Pamela Zave
Princeton University


Lenore D. Zuck
University of Illinois Chicago

References

Automated Attacker Synthesis for Distributed Protocols. Safecomp 2020

Max von Hippel
Northeastern Univ.

Cole Vick
Northeastern Univ.

Stavros Tripakis
Northeastern Univ.

Cristina Nita-Rotaru
Northeastern Univ.

A Formal Analysis of SCTP: Attack Synthesis and Patch Verification. Usenix Security 2024

Jacob Ginesin
Northeastern Univ.

Max von Hippel
Northeastern Univ.

Evan Defloor
Northeastern Univ.

Michael Tuxen
FH Munster

Cristina Nita-Rotaru
Northeastern Univ.

Formal Model-Driven Analysis of Resilience of GossipSub to Attacks from Misbehaving Peers. IEEE S&P 2024

Ankit Kumar
Northeastern Univ.

Max von Hippel
Northeastern Univ.

Pete Manolios
Northeastern Univ.

Cristina Nita-Rotaru
Northeastern Univ.

Thank you

Contact Information
Cristina Nita-Rotaru
c.nitarotaru@northeastern.edu
cnitarot.github.io