BACnet or "BADnet"? On the (In)Security of Implicitly Reserved Fields in BACnet

Qiguang Zhang[†], Junzhou Luo[†§], Zhen Ling[†], Yue Zhang[‡], Chongqing Lei[†], Christopher Morales[*], Xinwen Fu[*]

†Southeast University

‡Shandong University

§Fuyao University of Science and Technology

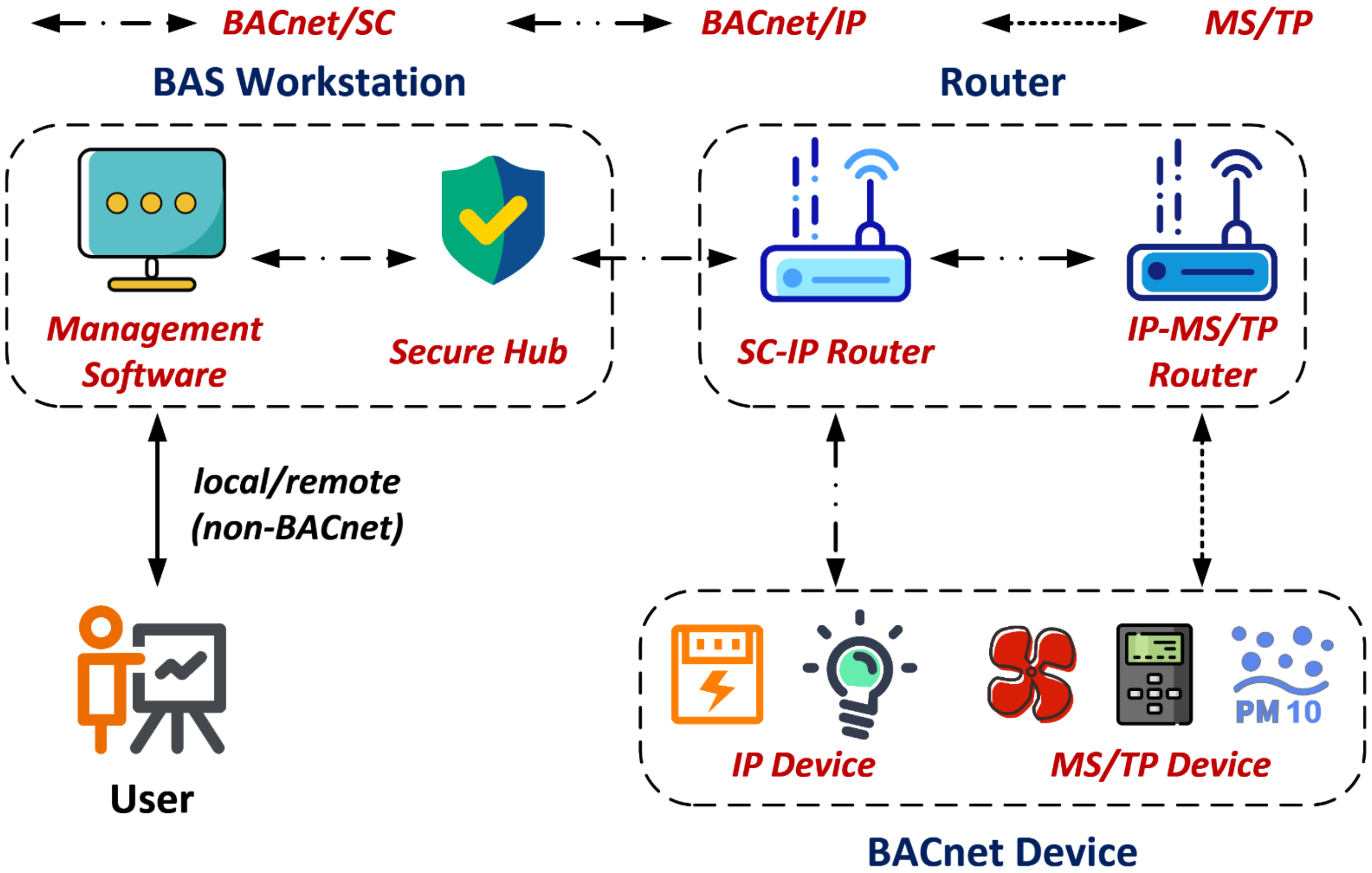
*University of Massachusetts Lowell

Background – Building Automation System

- **BAS:** A Building Automation System (BAS) transforms traditionally isolated building services into a networked system, enabling centralized supervision and automated coordination across HVAC, lighting, energy, and security subsystems
- **BACnet:** BACnet provides the standardized communication protocol enabling interoperability among BAS devices

Background – BACnet Network Topology

- **Workstation:** Centralized BAS management and operator interface
- **Router:** Protocol translation across BACnet networks
- **Devices:** Field sensors, actuators, and controllers



Background – Fuzzing

- **Fuzzing faces the following challenges when analyzing BAS devices**
  - BAS devices are typically closed-source
  - Monitoring device states and failures is difficult
- **Blackbox fuzzing is suitable for analyzing BAS devices**
  - Identifying which protocol fields to fuzz remains a key problem

Whitebox Fuzzing
Fully understand the target program

Greybox Fuzzing
Partially understand the target program

Blackbox Fuzzing
No need to understand the target program

Key Insights

Table 6-2. BACnet DADR and SADR Encodi

BACnet Data Link Layer	DLEN	SLEN
ARCNET, as defined in Clause 8	1	1
BACnet/IP, as defined in Annex J	6	6
BACnet/IPv6, as defined in Annex U	3	3
BACnet/SC, as defined in Annex AB	6	6
Ethernet, as defined in Clause 7	6	6
LonTalk domain wide broadcast	2	2
LonTalk multicast	2	2
LonTalk unicast	2	2
LonTalk, unique Neuron_ID	7	2
MS/TP, as defined in Clause 9	1	1
ZigBee, as defined in Annex O	3	3

DNET = 2-octet ultimate destination network number.
DLEN = 1-octet length of ultimate destination MAC layer address

(A value of 0 indicates a broadcast on the destination network.)

DADR = Ultimate destination MAC layer address.
DA = Local network destination MAC layer address.
SNET = 2-octet original source network number.
SLEN = 1-octet length of original source MAC layer address.
SADR = Original source MAC layer address.
SA = Local network source MAC layer address.

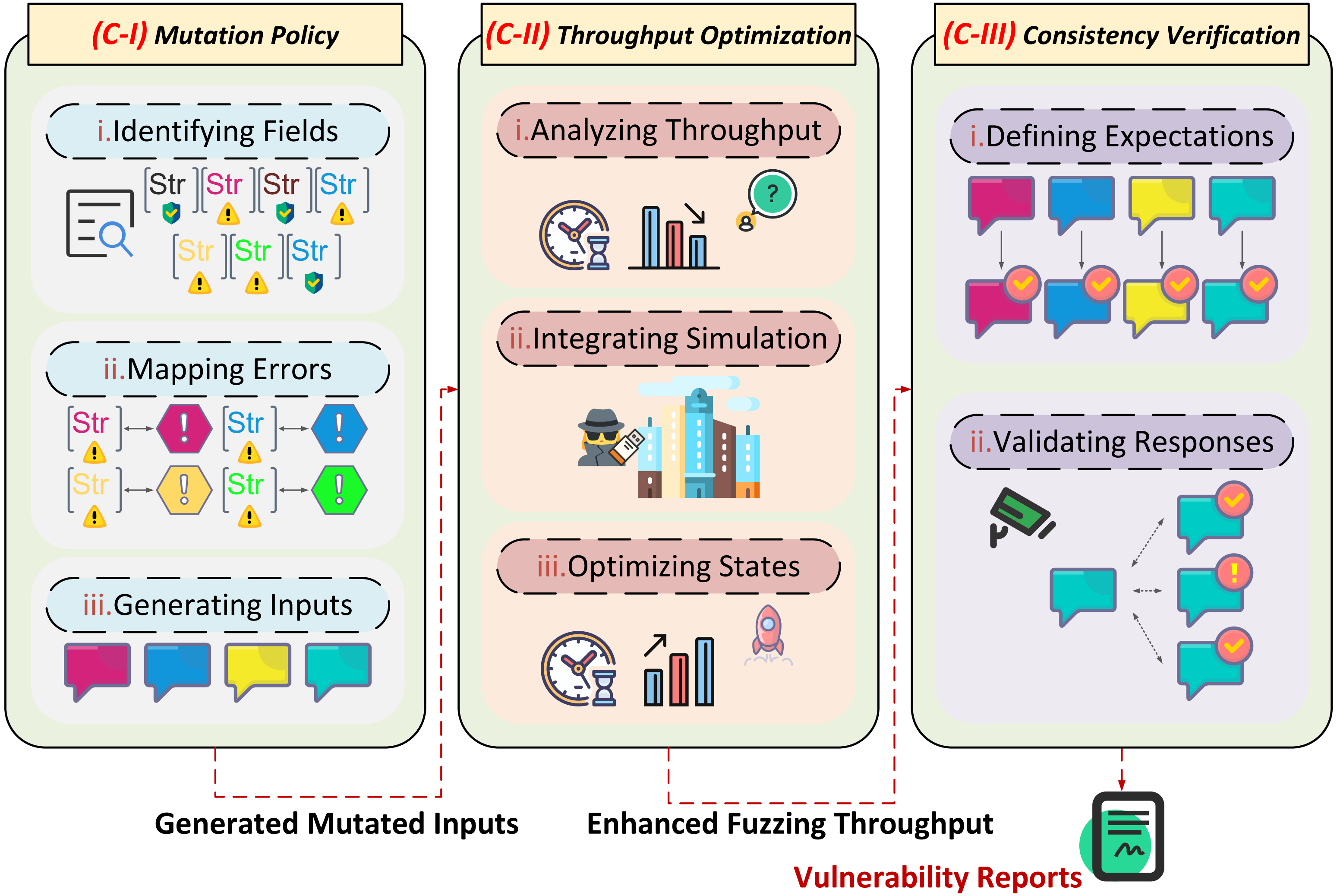
➢ Implicitly Reserved Fields

- ◆ DLEN and SLEN are defined as 1-octet fields (0x00–0xFF)
- ◆ Only a subset of values is valid in practice (0x01–0x03,0x06, 0x07 and 0x01–0x03,0x06)
- ◆ This discrepancy introduces vendor inconsistencies

BACSFUZZ Design – Challenges

- **Challenge I: Complexity in BACnet Message Field Mutation**
  - BACnet employs a layered, nested structure (LPDU → NPDU → APDU)
  - Message formats vary across layers and field types
  - Identifying structures and locating implicitly reserved fields is non-trivial
- **Challenge II: Low Data Throughput in Bus Networks**
  - Most fuzzers inject mutated BACnet/IP packets from the workstation via routers
  - Router-based translation (BACnet/IP → MS/TP) introduces performance constraints
  - MS/TP processes only a few packets per second, creating a throughput bottleneck
- **Challenge III: Black-Box Nature in Monitoring Fuzzing Status**
  - BAS devices are proprietary and heterogeneous
  - Internal execution states are difficult to observe
  - Liveness-based monitoring detects DoS but often fails to capture semantic violations

BACSFUZZ Design – Overview



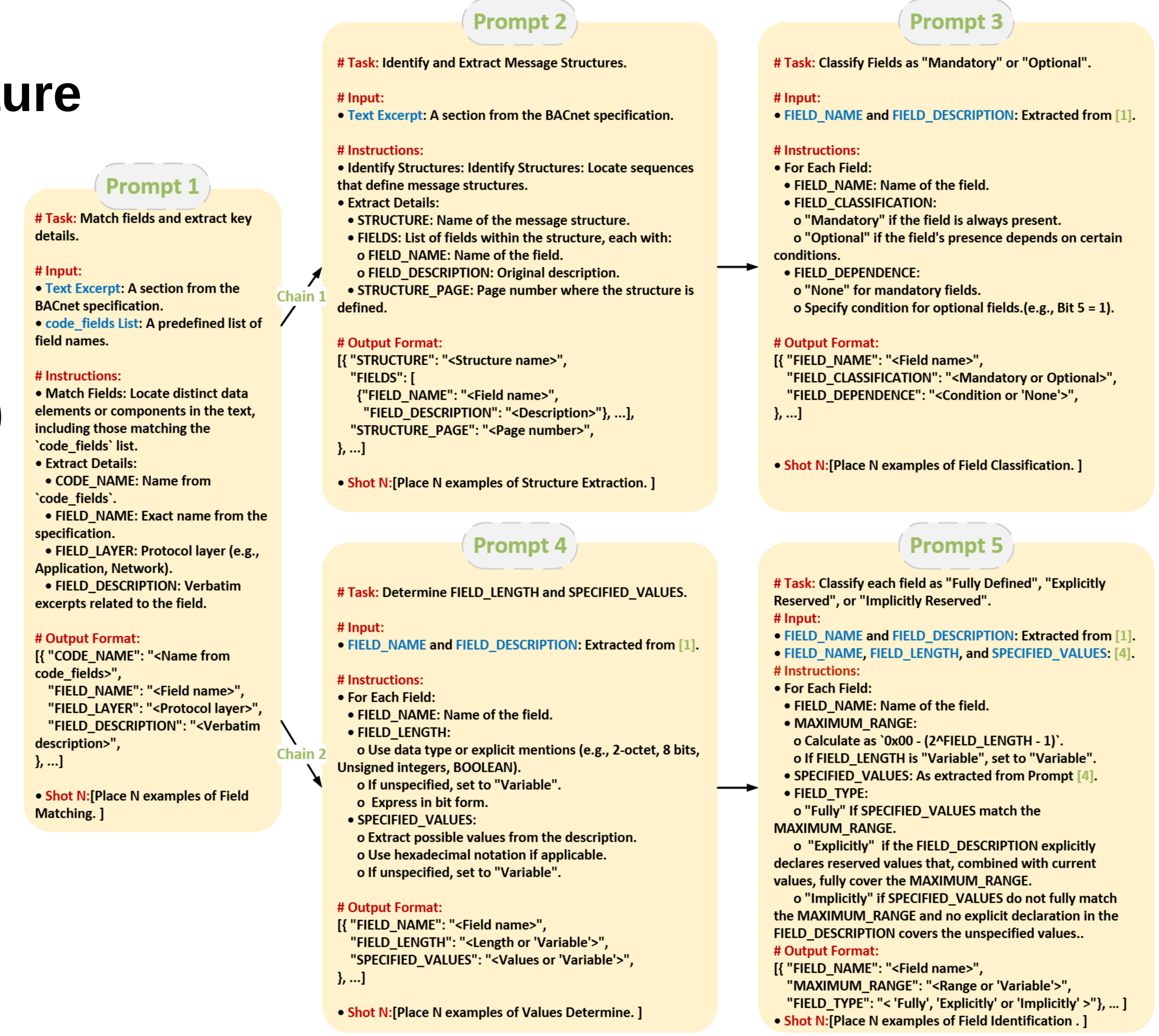
- ➢ **Implicitly Reserved Field-Based Mutation Policy → C-I**
- ➢ **Token-Seize-Assisted Throughput Optimization → C-II**
- ➢ **Byte Stream Format-Oriented Field Consistency Verification → C-III**

BACSFUZZ Design – Mutation Policy

➢ **LLM-Assisted Message Structure Identification (Prompt 1&2&3)**

➢ **LLM-Assisted Message Field Classification (Prompt 1&4&5)**

- ◆ Fully Defined Fields
- ◆ Explicitly Reserved Fields
- ◆ Implicitly Reserved Fields



BACSFUZZ Design – Mutation Policy (Cont'd)

➢BACnet Error-Handling

- ◆Invalid field values trigger Reject / Abort / Error APDUs
- ◆APDUs act as observable indicators of device error-handling behavior

➢Mutation Strategy

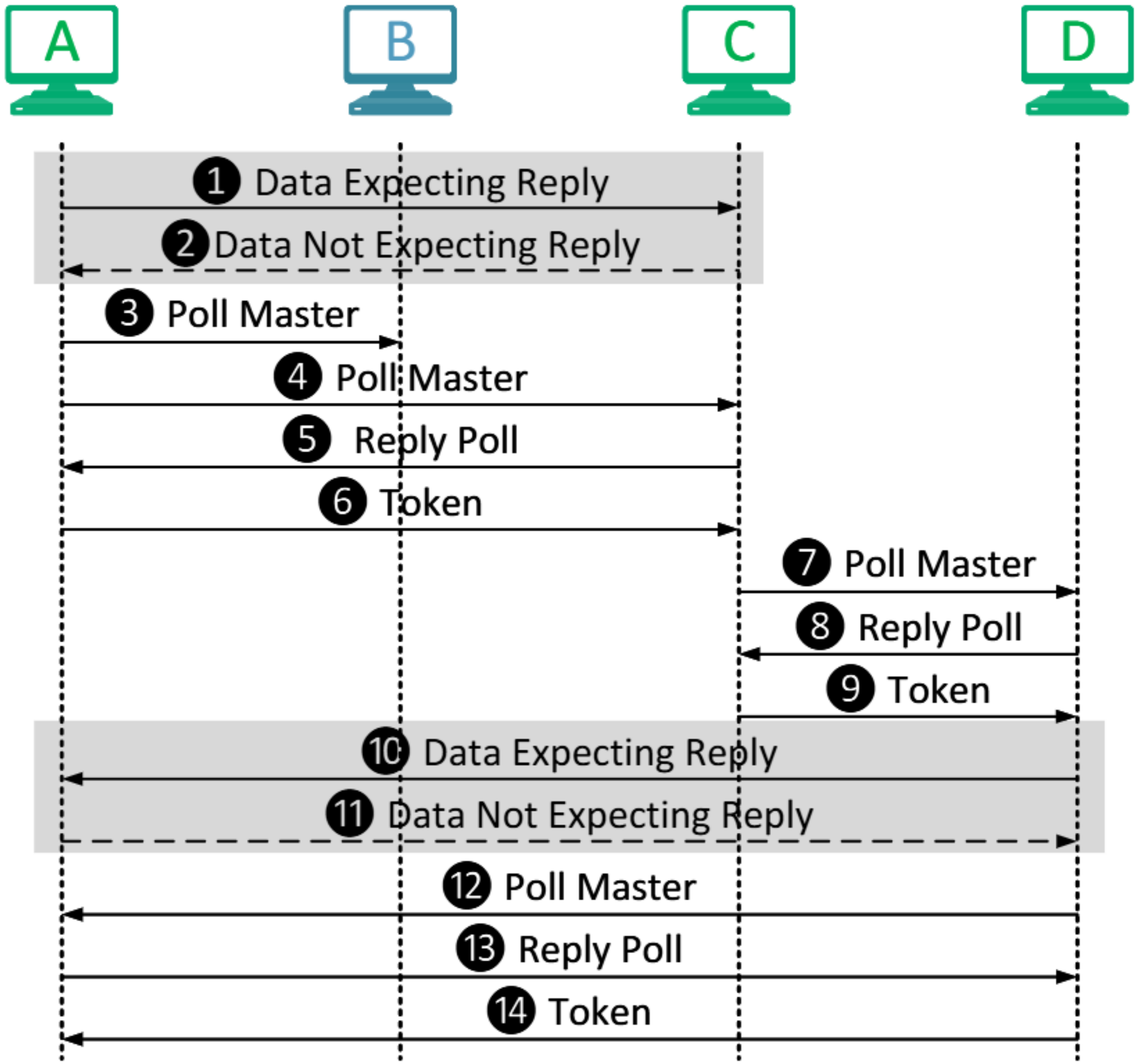
- ◆Fields with defined error semantics → Generate targeted error-triggering inputs
- ◆Fields without specified error behavior → Apply randomized out-of-range values

Type	Code	Description
Reject	1	buffer-overflow
	2	inconsistent-parameters
	3	invalid-parameter-data-type
	4	invalid-tag
	5	missing-required-parameter
	6	parameter-out-of-range
	7	too-many-arguments
	8	undefined-enumeration
	9	unrecognized-service
	10	invalid-data-encoding
Abort	4	segmentation-not-supported
	7	window-size-out-of-range
	11	apdu-too-long

BACSFUZZ Design – Throughput Optimization

➢MS/TP Limited Throughput

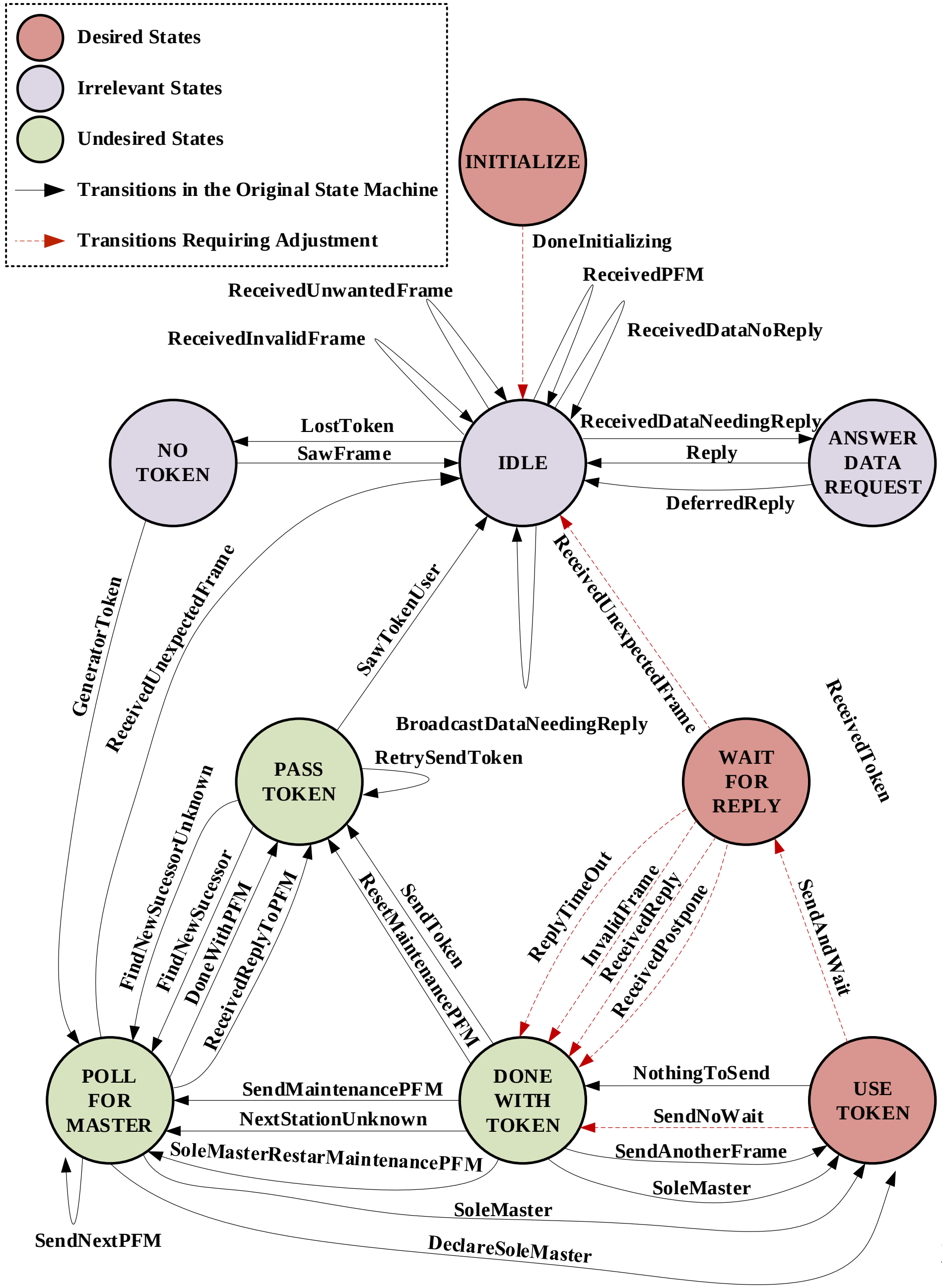
- ◆MS/TP employs a token-passing mechanism for medium access control
- ◆Only the token holder may transmit
- ◆Non-token nodes are restricted to passive responses
- ◆Token rotation introduces latency
- ◆Example: Only 4 of 14 steps involve data transmission (Steps 1, 2, 10, and 11)



BACSFUZZ Design – Throughput Optimization (Cont’d)

➢ **Protocol Behavior-Driven Fuzzer**

- ◆ Model token passing as a controllable protocol behavior
- ◆ State classification: Expected / Irrelevant / Undesired
- ◆ Native MS/TP interaction (no router overhead)
- ◆ Expected-state retention → persistent token control



BACSFUZZ Design – Consistency Verification

➢ Error Propagation Effect

- ◆ BACnet messages use a continuous byte-stream format
- ◆ Field position and value are critical for correct parsing
- ◆ Field shifts propagate parsing errors

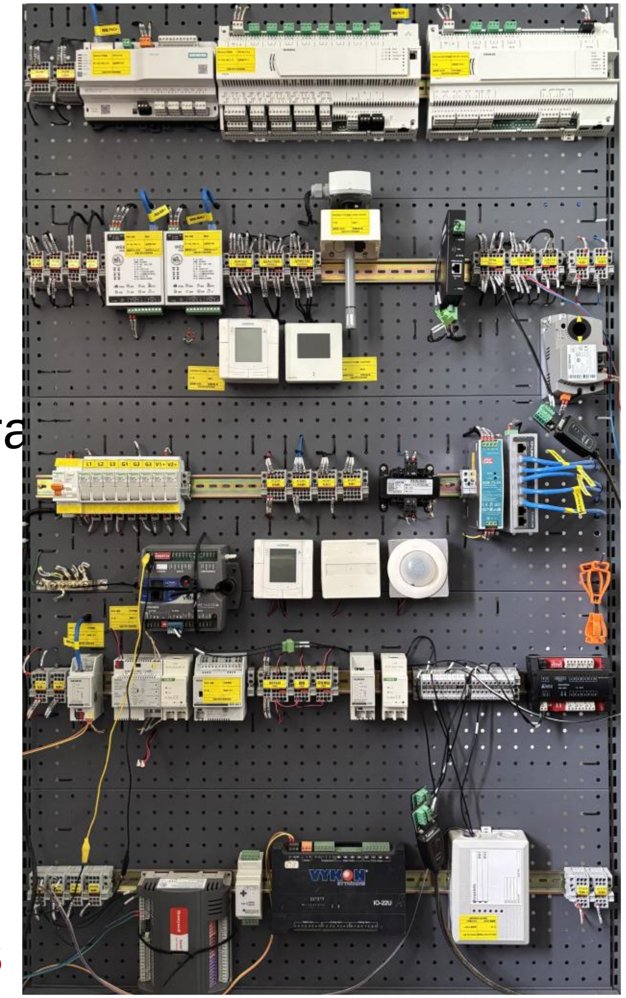
➢ Verification Strategy

- ◆ Validate response bytes at expected offsets
- ◆ Deviations → semantic inconsistencies
- ◆ No response → potential DoS

Evaluation: RQ1: How Effective is BACSFUZZ?

- **Dataset: 20 devices from 9 BAS vendors**
  - **Siemens:** 7 devices (11 vulnerabilities)
  - **Company X:** 4 devices (4 vulnerabilities)
  - **GVS:** 2 devices (4 vulnerabilities)
  - **Honeywell:** 2 devices (1 vulnerability)
  - **Contemporary Controls:** 1 device (3 vulnera
  - **Delta:** 1 device (1 vulnerability)
  - **ABB:** 1 device (1 vulnerability)
  - **Sunfull:** 1 device (1 vulnerability)
  - **Johnson:** 1 device

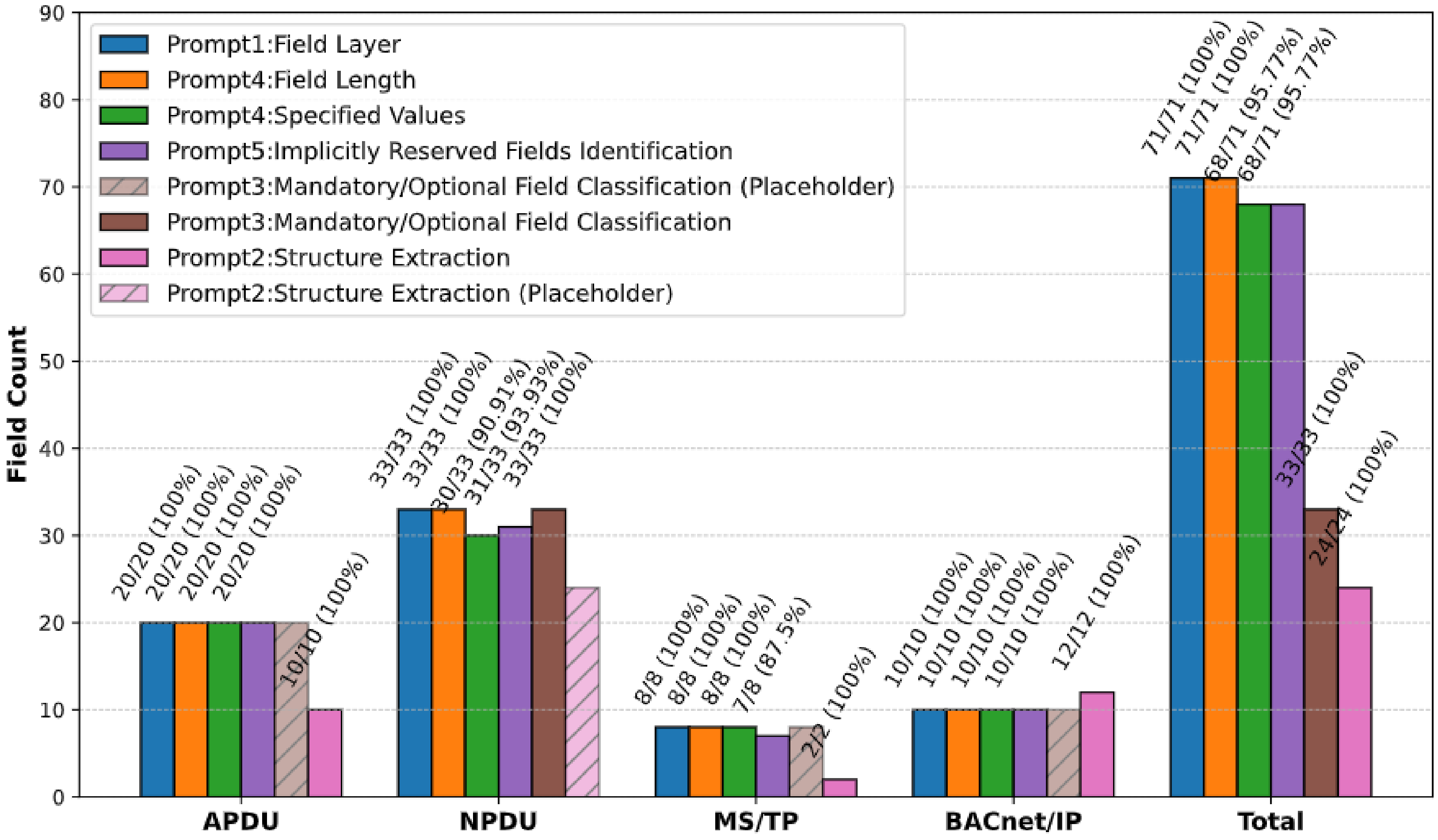
26 vulnerabilities, 24 confirmed, 9 CVEs



Conventional BACnet testbed

BACnet/SC testbed

RQ2: How Accurate is BACSFUZZ's LLM-Assisted Analysis?



- **All LLM outputs manually verified**
  - **Structure extraction → 100%**
  - **Field length extraction → 100%**
  - **Field value extraction → 95.77%**
  - **Implicitly reserved field identification → 95.77%**

RQ3: How Efficient is Protocol Behavior–Driven Fuzzing?

➢ Degradation (Table IV)

- ◆ BASE[1] : up to −57.77% (7 devices, 30 min)
- ◆ BACSFUZZ: fluctuation within ~1%

➢ Improvement (Table V)

- ◆ +272.49% (single device, 30 min)
- ◆ +776.01% (7 devices, 30 min)

➢ Takeaway

- ◆ BACSFUZZ mitigates token-induced delay
- ◆ Sustained high-throughput fuzzing

[1] Collapse Like A House of Cards: Hacking Building Automation System Through Fuzzing. (CCS 2024)

TABLE IV: BASE Throughput Degradation Analysis.

Min.	Type	2 vs.1	3 vs.1	4 vs.1	5 vs.1	6 vs.1	7 vs.1
5 min	BASE	-4.72%	-12.31%	-20.93%	-28.52%	-44.36%	-57.41%
	BACsFuzz	1.50%	-1.56%	0.23%	-0.35%	0.00%	0.02%
10 min	BASE	-3.97%	-12.66%	-25.35%	-31.21%	-48.37%	-58.83%
	BACsFuzz	0.08%	0.04%	0.06%	0.09%	-0.04%	0.08%
30 min	BASE	-4.11%	-13.59%	-19.48%	-29.83%	-47.45%	-57.77%
	BACsFuzz	-1.30%	0.11%	0.13%	0.05%	-1.06%	-0.68%

TABLE V: BACsFuzz Throughput Improvement Analysis.

Min.	Type	1 dev	2 devs	3 devs	4 devs	5 devs	6 devs	7 devs
5 min	BASE	1,357	1,293	1,190	1,073	970	755	578
	BACsFuzz	5,122	5,199	5,042	5,134	5,104	5,122	5,123
	↑	277.45%	302.09%	323.70%	378.47%	426.19%	578.41%	786.33%
10 min	BASE	2,820	2,708	2,463	2,105	1,940	1,456	1,161
	BACsFuzz	10,239	10,247	10,243	10,245	10,248	10,235	10,247
	↑	263.09%	278.40%	315.87%	386.70%	428.25%	602.95%	782.60%
30 min	BASE	8,240	7,901	7,120	6,635	5,782	4,330	3,480
	BACsFuzz	30,693	30,295	30,728	30,732	30,708	30,367	30,485
	↑	272.49%	283.43%	331.57%	363.18%	431.10%	601.32%	776.01%

n dev(s) : number of devices in the MS/TP network.
↑ : throughput improvement of BACsFuzz, compared to BASE.

RQ4: How does BACSFUZZ perform compared to a SOTA method

- **All compared fuzzers were evaluated only for DoS vulnerability detection**
  - BASE → V2
  - AFLnet / BooFuzz → No detection
  - BASE(Sto) → V7, V11
  - BooFuzz(Sto) → V2, V7
  - BACSFUZZ: Detects all vulnerabilities with significantly fewer packets

TABLE VII: Comparison with BASE, AFLnet, and BooFuzz.

	V1	V2	V3	V5	V7	V10	V11	V14	V16
BACSFUZZ	174	6,668	53	26,524	70	2,301	309	4,707	146
BASE	✗	8,521	✗	✗	✗	✗	✗	✗	✗
BASE(S_{to})	✗	✗	✗	✗	80,403	✗	366,791	✗	✗
AFLnet	✗	✗	✗	✗	✗	✗	✗	✗	✗
AFLnet (S_{to})	✗	✗	✗	✗	✗	✗	✗	✗	✗
BooFuzz	✗	✗	✗	✗	✗	✗	✗	✗	✗
BooFuzz (S_{to})	✗	16,067	✗	✗	29,700	✗	✗	✗	✗

Sto: Token-Seize-Assisted Throughput Optimization

Conclusion

- **BACSFUZZ is the first protocol behavior–driven fuzzer for BAS, improving fuzzing throughput by up to 776.01%**
- **We reveal a novel attack surface rooted in implicitly reserved fields, highlighting a general specification-level weakness**
- **Our evaluation uncovers 26 vulnerabilities — 24 confirmed by vendors, including 9 assigned CVEs**
- **Notably, the token-seize vulnerability was acknowledged by ASHRAE as a protocol-level flaw**

Q&A

Thanks!