

PCrafter Raw HTTP/TCP

SEQ/ACK 결함 분석·수정 및 전체 함수/클래스 의존 구조도

대상: pcrafter (1).zip / 수정본: fixed source package
문서 형식: A3 가로형 벡터 PDF - 확대 시 구조도와 코드 식별자 유지

분석 대상	핵심 결함	수정 검증
10개 Python 모듈 런타임 심볼 100개	응답 ACK 생성 시 client_seq가 이전 요청 시작 SEQ에 머무름	재현 1건 + 회귀시험 4건 PASS 전체 py_compile PASS

수정된 핵심 파일

TCP_pac.py - SND.NXT/SND.UNA 분리 및 ACK 범위 검증
HTTP_pac.py - 응답 1개만 소비, 전체 버퍼 삭제와 임의 sleep 제거
Vulnmart_ATTACK.py - 한 TCP 연결에서 전체 요청 흐름 재사용

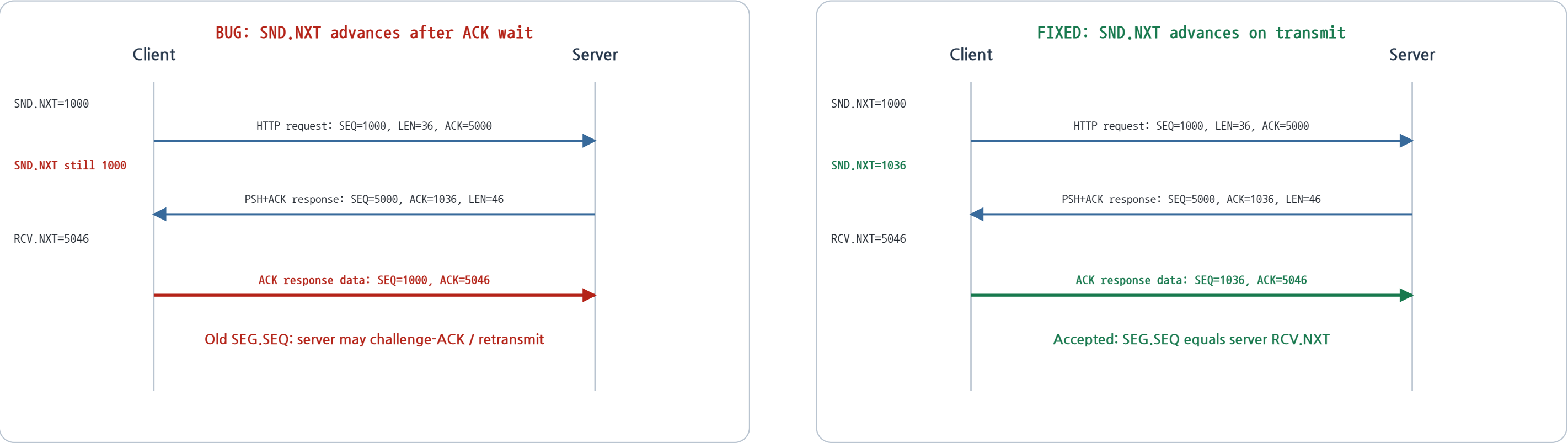
목차

1. SEQ/ACK 문제의 정확한 원인	3
버퍼 삭제가 원인이 아닌 이유	3
2. 전체 모듈 의존 구조	4
3. 클래스와 객체 소유 관계	5
4. HTTP 요청 1회의 호출 흐름	6
5. TCP 상태 머신과 핵심 상태 변수	7
6. 수정 내용과 코드 기준점	8
6.1 최초 송신 성공 직후, ACK 대기 전에 SND.NXT 이동	8
6.2 허용 범위를 넘는 ACK를 SND.UNA에 반영하지 않음	8
6.3 HTTP 수신 버퍼는 파싱한 응답 길이만 소비	8
6.4 한 RawHttpSession으로 전체 시나리오 수행	8
7. 재현 및 회귀 검증	9
8. 전체 함수·클래스 카탈로그	10
8.1 Vulnmart_ATTACK.py	10
8.2 HTTP_pac.py	10
8.3 TCP_pac.py	12
8.4 IP_pac.py	15
8.5 Ethernet_pac.py	15
8.6 Address_resolution.py	16
8.7 sender.py	16
8.8 getters.py	16
8.9 validation.py	17
8.10 ATTACK.py	17
9. 비활성 코드와 외부 의존성	18
수정 패키지에 포함된 보조 파일	18

1. SEQ/ACK 문제의 정확한 원인

업로드본의 `RawTcpSession_send_bytes()`는 데이터 세그먼트를 보내 뒤에도 `client_seq`를 유지하고 서버 ACK가 확인되 후에만 증가시킵니다. 하지만 같은 함수의 재저솟은 이미 고정되 `segment_seq`를 사용하므로 `SND.NXT`를 늘게 이동시킴 이유가 없습니다.

서버가 요청을 ACK하면서 HTTP response payload도 함께 보내는 경우, `_wait_for_ack()` -> `_process_packet()`이 그 payload를 처리하고 즉시 ACK를 생성합니다. 이때 기본 SEQ는 아직 이전 `client_seq`이므로 서버의 `RCV.NXT`보다 요청 길이만큼 뒤쳐집니다.



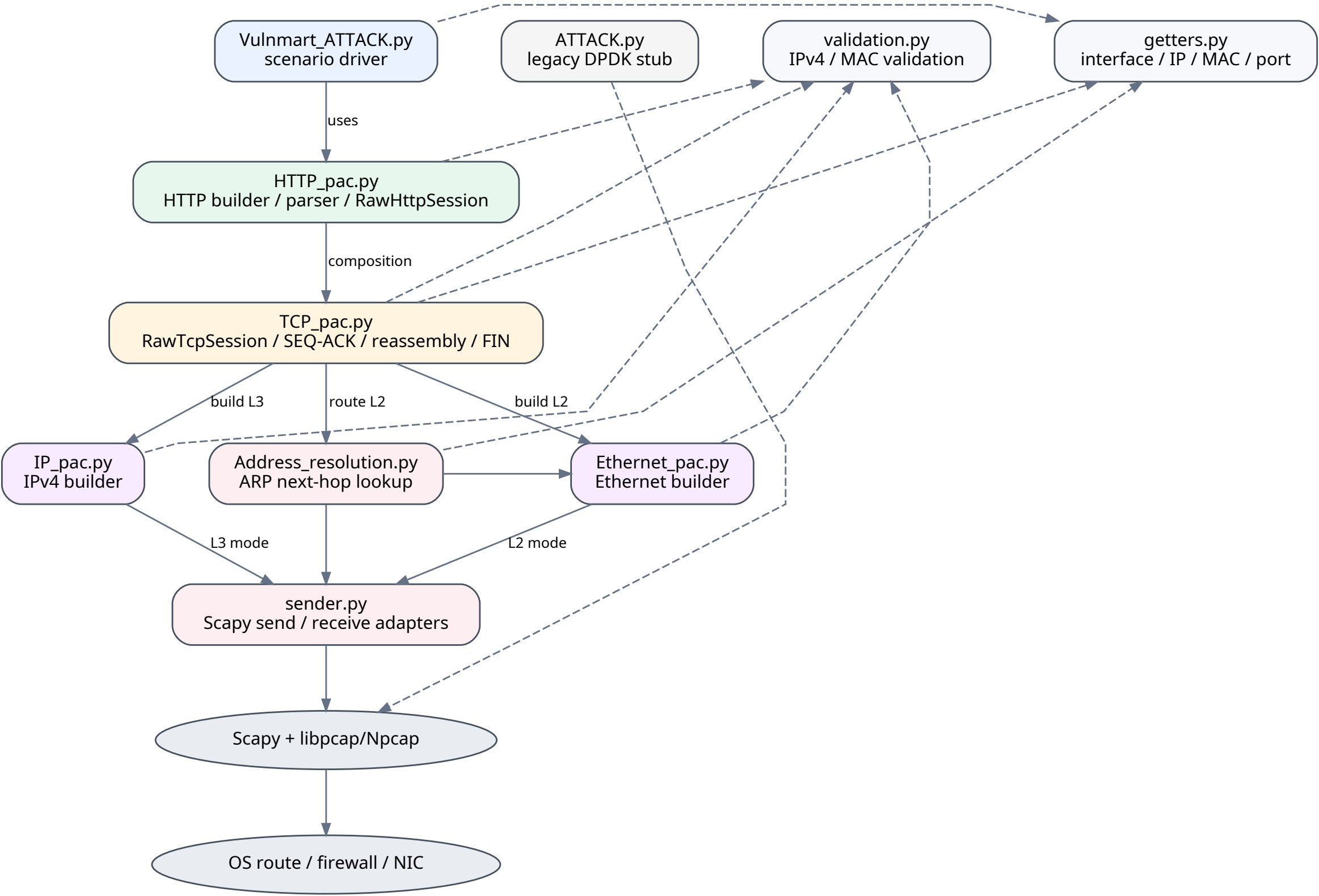
예시 값은 회귀시험과 동일합니다. 요청 길이 36바이트, 응답 길이 46바이트일 때 올바른 응답 ACK의 SEQ는 1036입니다.

버퍼 삭제가 원인이 아닌 이유

`server_next_seq(RCV.NXT)`는 TCP payload 수신 시 이미 `_accept_server_segment()`에서 증가합니다. `consume_received()`는 응용 계층 메모리만 줄이며 ACK 번호를 변경하지 않습니다. 따라서 전체 버퍼 삭제나 `sleep(0.05)`는 문제를 가릴 수 있을 뿐 수정이 아닙니다.

2. 전체 모듈 의존 구조

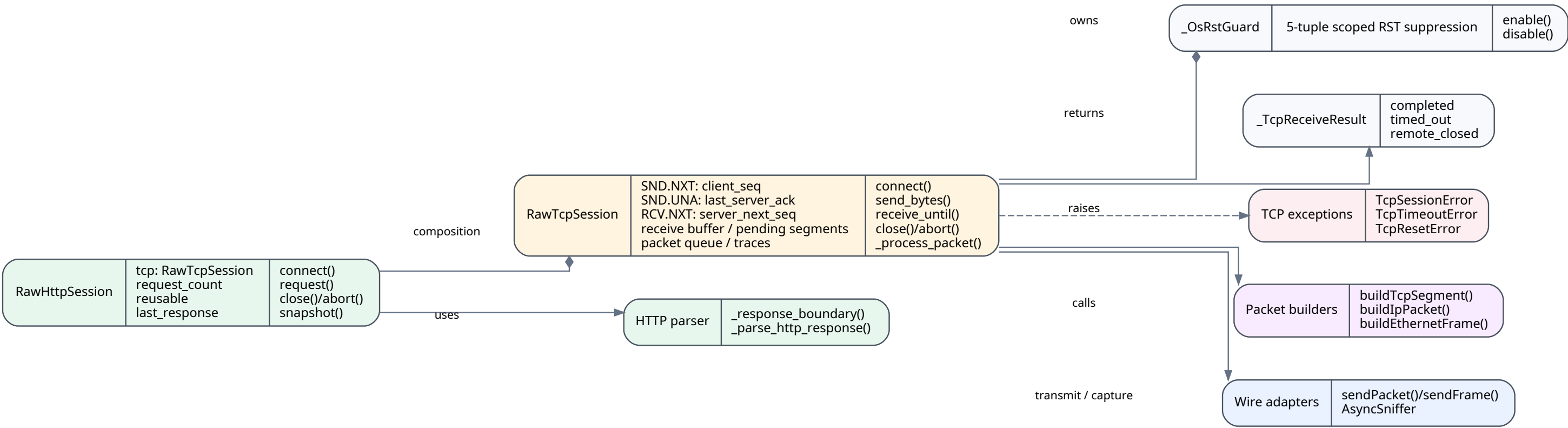
실선은 주 데이터 경로, 점선은 검증/환경 조회와 같은 공통 의존을 나타냅니다.



핵심 경로: Vulnmart_ATTACK -> HTTP_pac -> TCP_pac -> IP/Ethernet builders -> sender -> Scapy/NIC

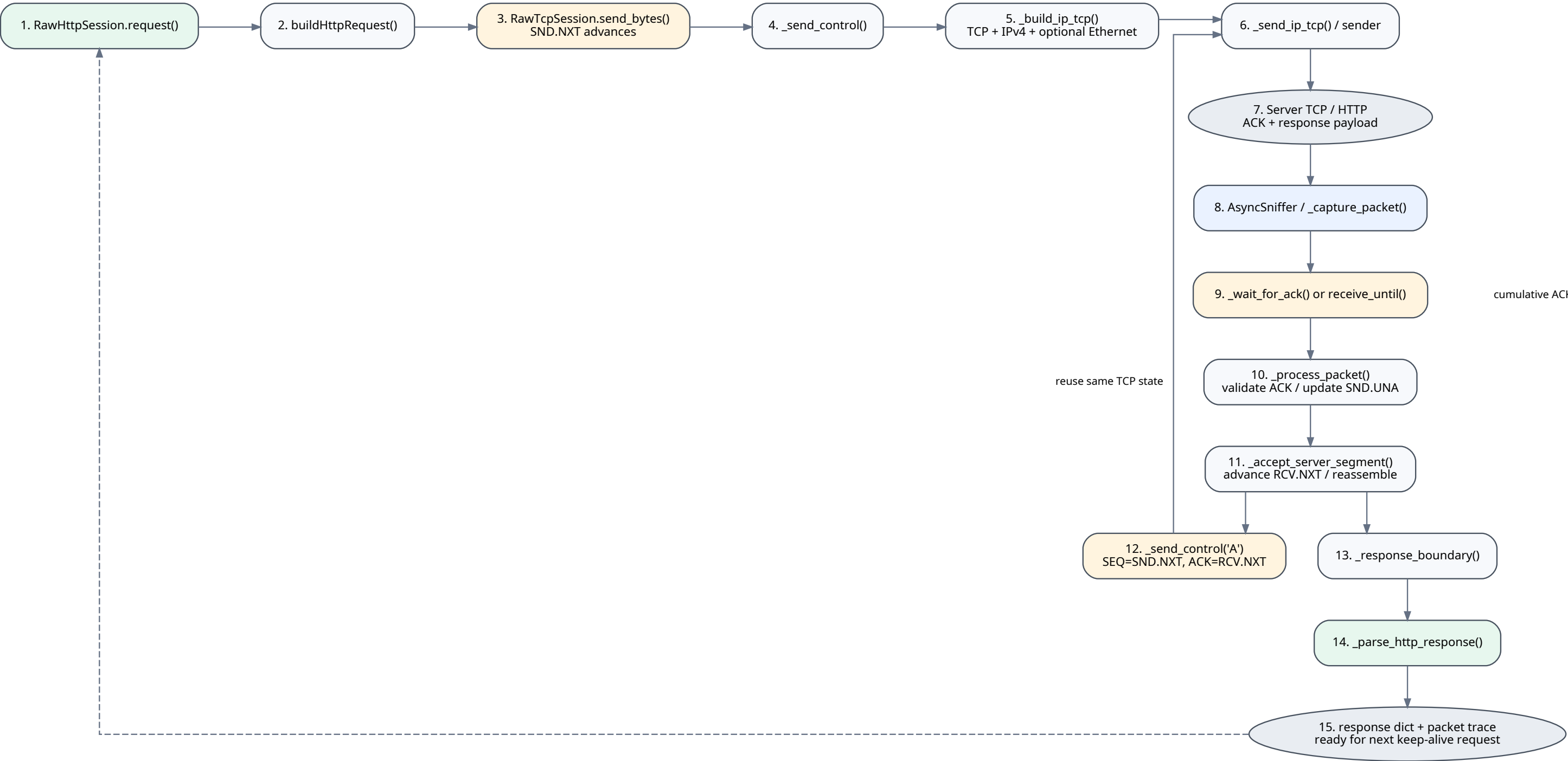
3. 클래스와 객체 소유 관계

RawHttpSession은 HTTP 상태를 따로 구현하지 않고 RawTcpSession 객체 하나를 소유합니다. SEQ/ACK의 단일 진실 공급원은 항상 RawTcpSession입니다.

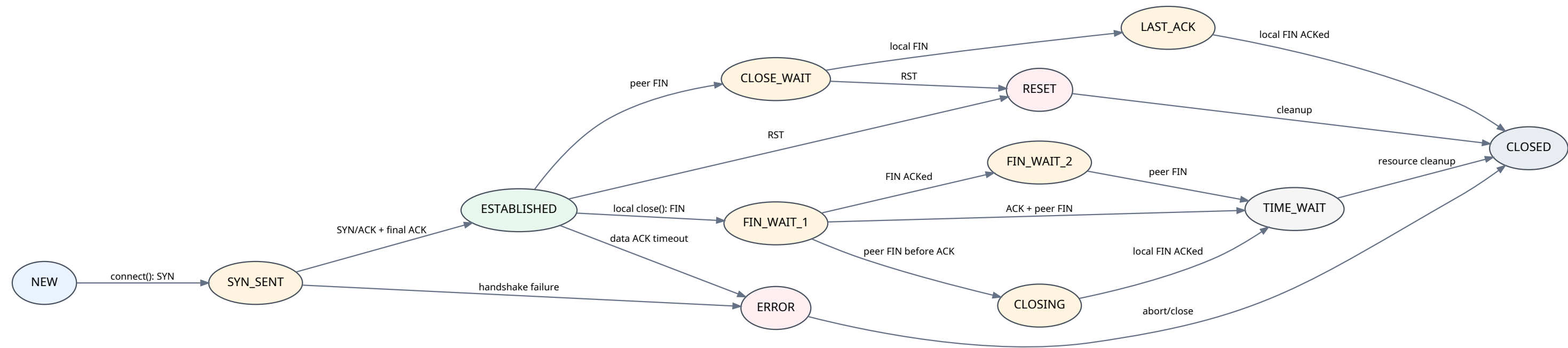


4. HTTP 요청 1회의 호출 흐름

송신, 캡처, ACK 처리, 재조립, HTTP 경계 파싱이 어떤 순서로 연결되는지 나타냅니다.



5. TCP 상태 머신과 핵심 상태 변수



변수	TCP 의미	증가 시점	사용 위치
client_seq	SND.NXT - 다음에 보낼 SEQ	SYN 1 소비, 데이터 전송 직후 LEN만큼, FIN 1 소비	_send_control 기본 SEQ, 다음 데이터/FIN 시작점
last_server_ack	SND.UNA - 서버가 누적 확인한 첫 미확인 바이트	허용된 SEG.ACK가 기존 값 이상이며 SND.NXT 이하일 때	_wait_for_ack 완료 판정
server_next_seq	RCV.NXT - 다음에 받을 서버 SEQ	연속 payload 길이와 FIN 1만큼	모든 클라이언트 ACK 번호
_receive_buffer	응용 계층에 전달할 재조립 bytes	RCV.NXT와 연속인 payload 수신 시	HTTP response parser 입력
_pending_segments	RCV.NXT보다 앞선 out-of-order payload	gap이 있는 세그먼트 수신 시	_drain_pending_segments에서 병합

6. 수정 내용과 코드 기준점

6.1 최초 송신 성공 직후, ACK 대기 전에 SND.NXT 이동

```
segment_seq = self.client_seq
target_ack = _seq_add(segment_seq, len(chunk))

# 재전송은 고정된 segment_seq 사용
self._send_control(
    flags, seq=segment_seq, ack=self.server_next_seq, payload=chunk
)

# 최초 송신 성공 후, 수신 큐 처리 전에 SND.NXT 전진
self.client_seq = target_ack
if self._wait_for_ack(target_ack, self.timeout):
    acknowledged = True
```

6.2 허용 범위를 넘는 ACK를 SND.UNA에 반영하지 않음

```
ack_accepted = not (
    self.client_seq is not None
    and _seq_signed_delta(incoming_ack, self.client_seq) > 0
)
if ack_accepted and (
    self.last_server_ack is None
    or _seq_ge(incoming_ack, self.last_server_ack)
):
    self.last_server_ack = incoming_ack
```

6.3 HTTP 수신 버퍼는 파싱한 응답 길이만 소비

```
self._stream_cursor += consumed
self.tcp.consume_received(self._stream_cursor)
self._stream_cursor = 0
```

6.4 한 RawHttpSession으로 전체 시나리오 수행

수정된 Vulnmart_ATTACK.cancelAllOrders()는 GET /, 로그인 POST, 302 Location GET, 주문 1~30 취소까지 동일 객체를 사용합니다. 요청마다 3-way/4-way를 반복하지 않습니다.

7. 재현 및 회귀 검증

검사	업로드본	수정본	판정 기준
Piggyback response ACK SEQ	1000	1036	요청 시작 1000 + LEN 36
동일 TCP에서 요청 2회	응답 ACK SEQ가 첫 요청 시작값으로 뒤처짐	둘째 요청 시작/ACK 모두 연속	SND.NXT/RCV.NXT 연속성
ACK > SND.NXT	last_server_ack에 반영 가능	무시	전송하지 않은 바이트 ACK 금지
HTTP buffer workaround	전체 삭제 + 50ms sleep	consumed만 삭제, sleep 제거	응용 버퍼와 TCP SEQ 분리
송신 호출 실패	선이동 시 SND.NXT 오염 가능	SND.NXT 유지 + ERROR	실패한 전송의 SEQ 소비 방지
회귀시험	-	5/5 PASS	네트워크 없는 결정론적 테스트
정적 문법	-	전체 py_compile PASS	수정 패키지 전 파일

```
python -m unittest -v tests/test_seq_ack_regression.py
```

```
test_ack_beyond_snd_nxt_is_not_accepted ... ok
test_http_layer_does_not_use_sleep_or_clear_entire_buffer ... ok
test_send_failure_does_not_advance_snd_nxt ... ok
test_piggyback_response_ack_uses_current_snd_nxt ... ok
test_two_requests_reuse_one_tcp_state ... ok
```

실제 네트워크 통합시험은 대상 서버, root 권한, libpcap/Npcap 및 RST 차단 환경이 필요합니다. 본 검증은 문제의 상태 전이를 실제 제품 메서드로 재현하는 단위 회귀시험입니다.

8. 전체 함수·클래스 카탈로그

아래 표는 수정본의 런타임 정의를 AST로 추출했습니다. 지역 중첩 함수 1개도 포함하며, 주석 문자열 안의 비활성 함수는 별도로 표시합니다.

8.1 Vulnmart_ATTACK.py

역할: 하나의 RawHttpSession으로 로그인, 리다이렉트, 주문 취소를 연속 수행하는 실습 시나리오 호출부.

직접 import: HTTP_pac (RawHttpSession); getters (getMainIface, getIfaceIp, getFreePort); urllib.parse (urlencode); time; typing (Dict, Mapping, Any)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
<code>_update_cookie_jar</code> <code>_update_cookie_jar(cookie_jar: Dict[str, str], response: Mapping[str, Any])</code> -> None L8-13	function	응답 딕셔너리의 cookies를 기존 쿠키 저장소에 병합한다.	-
<code>_cookie_header</code> <code>_cookie_header(cookie_jar: Mapping[str, str])</code> -> str L16-18	function	쿠키 저장소를 HTTP Cookie 헤더 문자열로 직렬화한다.	-
<code>cancelAllOrders</code> <code>cancelAllOrders(wasIp: str, username: str, password: str)</code> -> None L21-176	function	동일 TCP 연결에서 GET /, 로그인 POST, 302 추적, 주문 1~30 취소를 순차 실행하고 마지막에 4-way 종료한다.	RawHttpSession, _cookie_header, _update_cookie_jar, getFreePort, getIfaceIp, getMainIface, httpSession.abort, httpSession.close, httpSession.connect, httpSession.request 외 1개

8.2 HTTP_pac.py

역할: HTTP/1.1 메시지 조립, 응답 경계 판정/파싱, 쿠키 추출, 유지형 RawHttpSession 추상화.

직접 import: __future__ (annotations); gzip; json; re; time; zlib; http.cookies (SimpleCookie); typing (Any, Dict, List, Mapping, Optional, Tuple, Union); scapy.layers.inet (TCP); scapy.packet (Packet, Raw); TCP_pac (RawTcpSession, TcpSessionError, TcpTimeoutError); validation (isValidIpv4)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
<code>_body_to_bytes</code> <code>_body_to_bytes(body: HttpBody)</code> -> bytes L36-43	function	지원되는 HTTP body 타입을 bytes로 정규화한다.	-
<code>_find_header_key</code> <code>_find_header_key(headers: Mapping[str, str], name: str)</code> -> Optional[str] L46-51	function	대소문자를 무시하고 매핑에 존재하는 실제 헤더 키를 찾는다.	-
<code>_header_value</code> <code>_header_value(headers: Mapping[str, str], name: str, default: str='')</code> -> str L54-56	function	대소문자 비구분으로 헤더 값을 조회하고 기본값을 제공한다.	_find_header_key
<code>_validate_header</code> <code>_validate_header(name: str, value: str)</code> -> None L59-63	function	헤더 이름의 colon/개행과 값의 CR/LF 삽입을 차단한다.	-
<code>buildHttpRequest</code> <code>buildHttpRequest(method: str, uri: str, version: str='HTTP/1.1', headers: Optional[Dict[str, str]]=None, body: HttpBody=None, auto_content_length: bool=True, raw_output: bool=False)</code> -> Union[Raw, bytes] L66-111	function	Request-Line, headers, 빈 줄, body를 HTTP wire bytes 또는 Scapy Raw로 조립한다.	Raw, _body_to_bytes, _find_header_key, _validate_header
<code>buildHttpResponse</code> <code>buildHttpResponse(status_code: int, reason: str, version: str='HTTP/1.1', headers: Optional[Dict[str, str]]=None, body: HttpBody=None, auto_content_length: bool=True, raw_output: bool=False)</code> -> Union[Raw, bytes] L114-152	function	Status-Line, headers, body를 실습용 HTTP response wire 형식으로 조립한다.	Raw, _body_to_bytes, _find_header_key, _validate_header
<code>_parse_header_block</code> <code>_parse_header_block(block: bytes)</code> -> Tuple[str, int, str, List[Tuple[str, str]], Dict[str, List[str]]] L155-190	function	Status-Line과 반복 헤더를 순서 보존 목록 및 다중값 딕셔너리로 파싱한다.	-
<code>_locate_final_response</code> <code>_locate_final_response(buffer: bytes)</code> -> Optional[Tuple[int, int, str, int, str, List[Tuple[str, str]], Dict[str, List[str]]]] L193-227	function	100 Continue 등 중간 1xx를 건너뛰고 최종 응답 헤더의 시작/본문 위치를 찾는다.	_parse_header_block

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
_decode_chunked _decode_chunked(data: bytes) -> Tuple[bytes, bool, int] L230-263	function	chunk size, data, trailer를 해석해 decoded body, 완료 여부, 소비 길이를 반환한다.	-
_response_has_no_body _response_has_no_body(method: str, status_code: int) -> bool L266-271	function	HEAD, 1xx, 204, 304 응답이 본문을 갖지 않는지 판정한다.	-
_response_boundary _response_boundary(buffer: bytes, method: str, *, remote_closed: bool=False) -> Optional[Dict[str, Any]] L274-333	function	Content-Length, chunked, connection-close 규칙으로 응답 1개의 정확한 끝을 계산한다.	_decode_chunked, _locate_final_response, _response_has_no_body
_http_response_complete _http_response_complete(buffer: bytes, method: str) -> bool L336-337	function	원격 FIN 없이도 현재 버퍼에서 응답 경계가 완성됐는지 빠르게 판정한다.	_response_boundary
_parse_http_response _parse_http_response(raw_tcp_stream: bytes, *, method: str, remote_closed: bool, timed_out: bool) -> Dict[str, Any] L340-500	function	헤더/본문/chunked/content-encoding/charset/JSON/cookie를 파싱하고 남은 바이트와 완료 상태를 반환한다.	SimpleCookie, _decode_chunked, _locate_final_response, _response_has_no_body, gzip.decompress, json_module.loads, zlib.decompress
parseHttpResponse parseHttpResponse(rawResponse: bytes, *, requestMethod: str='GET', finReceived: bool=False, timedOut: bool=False) -> Dict[str, Any] L503-518	function	이미 TCP 순서대로 재조립된 bytes를 공개 파서 인터페이스로 처리한다.	_parse_http_response
_tcp_payload_bytes _tcp_payload_bytes(packet: Packet) -> bytes L521-527	function	Scapy Packet의 TCP payload를 안전하게 bytes로 추출한다.	-
RawHttpSession class RawHttpSession(-) L530-820	class	RawTcpSession을 소유하고 HTTP 요청/응답 경계, keep-alive 재사용 여부와 응답 메타데이터를 관리한다.	bases: -
RawHttpSession.__init__ __init__(self, dstIp: str, dstPort: int=80, *, srcIp: Optional[str]=None, srcPort: Optional[int]=None, iface: Optional[str]=None, srcMac: Optional[str]=None, dstMac: Optional[str]=None, handshakeTimeout: float=2.0, retries: int=2, ttl: int=64, tcpWindow: int=64240, mss: int=1460, useLayer2: bool=True, blockOsRst: bool=True, maxResponseBytes: int=10 * 1024 * 1024, capturePackets: bool=True, defaultHeaders: Optional[Mapping[str, str]]=None, debug: bool=False) -> None L537-594	method	HTTP 기본 헤더와 내부 RawTcpSession, stream cursor, 재사용/종료 상태를 초기화한다.	RawTcpSession, _validate_header, isValidIpv4, self.default_headers.items
RawHttpSession.state @property state(self) -> str L597-598	method	내부 TCP 상태를 노출한다.	-
RawHttpSession.seq @property seq(self) -> Optional[int] L601-602	method	내부 TCP SND.NXT를 노출한다.	-
RawHttpSession.ack @property ack(self) -> Optional[int] L605-606	method	내부 TCP RCV.NXT를 노출한다.	-
RawHttpSession.connected @property connected(self) -> bool L609-610	method	내부 TCP가 ESTABLISHED인지 반환한다.	-
RawHttpSession.request_count @property request_count(self) -> int L613-614	method	현재 TCP 연결에서 완료한 HTTP 요청 수를 반환한다.	-
RawHttpSession.connect connect(self) -> 'RawHttpSession' L616-620	method	종료 여부를 확인하고 내부 RawTcpSession.connect를 호출한다.	TcpSessionError, self.tcp.connect
RawHttpSession._prepare_headers _prepare_headers(self, headers: Optional[Mapping[str, str]]) -> Dict[str, str] L622-646	method	기본/요청 헤더를 대소문자 중복 없이 병합하고 Host, Accept, User-Agent, keep-alive를 보완한다.	_find_header_key, _validate_header

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
RawHttpSession.request request(self, method: str='GET', uri: str='/', headers: Optional[Mapping[str, str]]=None, body: HttpBody=None, *, version: str='HTTP/1.1', responseTimeout: float=5.0, idleTimeout: Optional[float]=None, autoContentLength: bool=True) -> Dict[str, Any] L648-776	method	HTTP bytes 전송, TCP 응답 수신, 경계 파싱, 소비 바이트 제거, keep-alive 판정과 trace 수집을 한 번에 수행한다.	TcpSessionError, TcpTimeoutError, _body_to_bytes, _header_value, _header_value().lower, _parse_http_response, _response_boundary, _tcp_payload_bytes, self._prepare_headers, self.connect 외 4개
RawHttpSession.close close(self, timeout: Optional[float]=None, *, requireComplete: bool=False) -> Dict[str, Any] L778-794	method	내부 TCP 4-way 종료 결과에 HTTP 요청 수와 snapshot을 결합한다.	self.tcp.close, self.tcp.snapshot
RawHttpSession.abort abort(self) -> None L796-799	method	내부 TCP를 RST/정리하고 HTTP 세션을 재사용 불가로 표시한다.	self.tcp.abort
RawHttpSession.snapshot snapshot(self) -> Dict[str, Any] L801-810	method	HTTP 상태와 내부 TCP snapshot을 함께 반환한다.	self.tcp.snapshot
RawHttpSession.__enter__ __enter__(self) -> 'RawHttpSession' L812-813	method	with 진입 시 connect한다.	self.connect
RawHttpSession.__exit__ __exit__(self, exc_type: Any, exc: Any, traceback: Any) -> bool L815-820	method	정상 종료면 close, 예외면 abort한다.	self.abort, self.close
RawHttpSession._prepare_headers.<locals>.set_default set_default(name: str, value: str) -> None L636-638	nested function	요청 헤더에 동일 이름이 없을 때만 기본 헤더 값을 추가하는 지역 헬퍼 함수.	_find_header_key
openHttpSession openHttpSession(dstIp: str, dstPort: int=80, **sessionOptions: Any) -> RawHttpSession L827-837	function	연결 완료된 유지형 RawHttpSession을 반환한다.	RawHttpSession, RawHttpSession().connect
sendHttpRequest sendHttpRequest(dstIp: str, dstPort: int=80, method: str='GET', uri: str='/', headers: Optional[Mapping[str, str]]=None, body: HttpBody=None, *, srcIp: Optional[str]=None, srcPort: Optional[int]=None, iface: Optional[str]=None, srcMac: Optional[str]=None, dstMac: Optional[str]=None, handshakeTimeout: float=2.0, responseTimeout: float=5.0, idleTimeout: Optional[float]=None, closeTimeout: Optional[float]=None, retries: int=2, ttl: int=64, tcpWindow: int=64240, mss: int=1460, useLayer2: bool=True, blockOsRst: bool=True, maxResponseBytes: int=10 * 1024 * 1024, capturePackets: bool=True, requireCloseComplete: bool=False, debug: bool=False) -> Dict[str, Any] L840-907	function	요청 1회용 RawHttpSession을 만들고 요청 후 close까지 수행하는 편의 함수.	RawHttpSession, session.abort, session.close, session.request, session.tcp.snapshot

8.3 TCP_pac.py

역할: Raw TCP 3-way/데이터/재조립/ACK/재전송/4-way 상태 관리의 핵심 모듈.

직접 import: __future__ (annotations); atexit; os; platform; queue; random; shutil; subprocess; threading; time; dataclasses (dataclass); typing (Any, Callable, Dict, List, Optional, Tuple, Union); scapy.all (AsyncSniffer, IP, TCP, Packet, Raw, conf); Address_resolution (getMacByArp); Ethernet_pac (buildEthernetFrame); IP_pac (buildIpPacket); getters (getFreePort, getIfaceIp, getIfaceMac, getMainIface); sender (rSendPacket, sendFrame, sendPacket); validation (isValidIpv4)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
TcpSessionError class TcpSessionError(RuntimeError) L55-56	class	Raw TCP 세션 공통 런타임 예외의 기준 클래스.	bases: RuntimeError
TcpTimeoutError class TcpTimeoutError(TimeoutError, TcpSessionError) L59-60	class	SYN/ACK, 데이터 ACK, 종료 절차가 제한 시간 안에 완료되지 않은 경우.	bases: TimeoutError, TcpSessionError
TcpResetError class TcpResetError(ConnectionResetError, TcpSessionError) L63-64	class	상대 또는 로컬 스택의 RST를 세션 오류로 표현한다.	bases: ConnectionResetError, TcpSessionError
_seq_add _seq_add(seq: int, amount: int) -> int L67-68	function	32비트 TCP Sequence 공간에서 순환 덧셈을 수행한다.	-

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
_seq_signed_delta _seq_signed_delta(seq: int, reference: int) -> int L71-74	function	32비트 순환 공간에서 두 SEQ의 부호 있는 상대 거리를 계산한다.	-
_seq_ge _seq_ge(left: int, right: int) -> bool L77-79	function	TCP 순환 규칙을 적용해 left가 right 이상인지 판정한다.	_seq_signed_delta
_choose_ephemeral_port _choose_ephemeral_port() -> int L82-89	function	커널 소켓을 열지 않고 IANA 동적 포트 범위에서 임의 출발지 포트를 선택한다.	-
buildTcpSegment buildTcpSegment(dport: int, sport: int, seq: Optional[int]=None, ack: int=0, dataofs: Optional[int]=None, reserved: int=0, flags: Union[int, str]='', window: int=8192, chksum: Optional[int]=None, urgptr: int=0, options: Optional[List[Tuple[str, Any]]]=None, payload: Union[str, bytes, Packet, None]=None) -> TCP L92-140	function	TCP 헤더 필드, 옵션, payload를 Scapy TCP 세그먼트로 조립한다.	Raw, TCP
_TcpReceiveResult class _TcpReceiveResult(-) L144-147	class	receive_until 결과의 완료, 타임아웃, 원격 종료 여부를 담는 dataclass.	bases: -
_OsRstGuard class _OsRstGuard(-) L150-347	class	Raw TCP 5-tuple에 대해 OS가 자동 생성하는 outbound RST를 임시 차단한다.	bases: -
_OsRstGuard.__init__ __init__(self, src_ip: str, dst_ip: str, src_port: int, dst_port: int, enabled: bool=True) -> None L159-175	method	차단할 출발지/목적지 IP와 포트 및 활성화 여부를 저장한다.	-
_OsRstGuard.enable enable(self) -> None L177-194	method	운영체제를 판별해 Linux 또는 Windows용 RST 차단 규칙을 적용하고 종료 시 정리를 등록한다.	TcpSessionError, self._enable_linux, self._enable_windows
_OsRstGuard._enable_linux _enable_linux(self) -> None L196-279	method	iptables 또는 nftables에 정확한 5-tuple RST DROP 규칙을 추가한다.	PermissionError, TcpSessionError
_OsRstGuard._enable_windows _enable_windows(self) -> None L281-304	method	PowerShell New-NetFirewallRule로 주소/포트를 제한한 outbound 차단 규칙을 만든다.	TcpSessionError
_OsRstGuard.disable disable(self) -> None L306-347	method	추가했던 iptables/nftables/Windows 방화벽 규칙을 제거한다.	-
RawTcpSession class RawTcpSession(-) L350-1368	class	하나의 5-tuple에서 TCP 상태, SND.NXT/SND.UNA/RCV.NXT, 재조립, 재전송과 FIN 교환을 관리한다.	bases: -
RawTcpSession.__init__ __init__(self, dstIp: str, dstPort: int, *, srcIp: Optional[str]=None, srcPort: Optional[int]=None, iface: Optional[str]=None, srcMac: Optional[str]=None, dstMac: Optional[str]=None, timeout: float=2.0, retries: int=2, ttl: int=64, window: int=64240, mss: int=1460, useLayer2: bool=True, blockOsRst: bool=True, maxReceiveBytes: int=10 * 1024 * 1024, capturePackets: bool=True, debug: bool=False) -> None L385-473	method	세션 옵션, 경로/L2 정보, TCP 상태 변수, 수신 버퍼, pending segment, 패킷 큐와 trace를 초기화한다.	_choose_ephemeral_port, isValidIpv4
RawTcpSession.received_bytes @property received_bytes(self) -> bytes L476-477	method	재조립 완료된 TCP 수신 스트림의 현재 복사본을 반환한다.	-
RawTcpSession.seq @property seq(self) -> Optional[int] L480-482	method	다음 송신 바이트의 Sequence Number인 SND.NXT를 반환한다.	-
RawTcpSession.ack @property ack(self) -> Optional[int] L485-487	method	다음 수신할 서버 바이트의 Sequence Number인 RCV.NXT를 반환한다.	-
RawTcpSession.connected @property connected(self) -> bool L490-491	method	상태가 ESTABLISHED인지 확인한다.	-

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
RawTcpSession.effective_mss @property effective_mss(self) -> int L494-495	method	로컬 MSS와 peer MSS 중 안전한 최소값을 계산한다.	-
RawTcpSession.consume_received consume_received(self, count: int) -> bytes L497-511	method	응용 계층이 처리한 바이트만 재조립 버퍼 앞에서 제거한다. RCV.NXT에는 영향을 주지 않는다.	-
RawTcpSession._log _log(self, message: str) -> None L513-515	method	debug 옵션이 켜진 경우 세션 진단 메시지를 출력한다.	-
RawTcpSession._prepare_route _prepare_route(self) -> None L517-571	method	Scapy 라우팅 테이블로 인터페이스, 출발지 IP, 게이트웨이와 다음 홉 MAC을 결정한다.	TcpSessionError, getInterfaceIp, getInterfaceMac, getMacByArp, getMainInterface, isValidIpv4, self._log, self.dst_ip.startswith, self.interface.lower
RawTcpSession._capture_filter _capture_filter(self, packet: Packet) -> bool L573-584	method	정확한 역방향 4-tuple의 IPv4/TCP 패킷만 캡처 대상으로 판정한다.	-
RawTcpSession._capture_packet _capture_packet(self, packet: Packet) -> None L586-592	method	캡처 패킷을 trace에 복사하고 세션 큐에 넣는다.	self._packet_queue.put, self.rx_packets.append
RawTcpSession._start_sniffer _start_sniffer(self) -> None L594-614	method	AsyncSniffer를 시작하고 캡처 스레드 준비 완료를 기다린다.	AsyncSniffer, TcpSessionError, self._sniffer.start, self._sniffer_ready.clear, self._sniffer_ready.wait, self._stop_sniffer
RawTcpSession._stop_sniffer _stop_sniffer(self) -> None L616-624	method	동작 중인 AsyncSniffer를 안전하게 중지한다.	-
RawTcpSession._next_ip_id _next_ip_id(self) -> int L626-629	method	IPv4 Identification 값을 1~65535 범위에서 순환 증가시킨다.	-
RawTcpSession._build_ip_tcp _build_ip_tcp(self, *, seq: int, ack: int, flags: Union[int, str], payload: Optional[bytes]=None, options: Optional[List[Tuple[str, Any]]]=None) -> Packet L631-660	method	현재 5-tuple과 상태값으로 TCP 세그먼트와 IPv4 패킷을 조립한다.	TcpSessionError, buildIpPacket, buildTcpSegment, self._next_ip_id
RawTcpSession._send_ip_tcp _send_ip_tcp(self, ip_packet: Packet) -> None L662-682	method	L2 모드면 Ethernet 프레임, L3 모드면 IP 패킷을 전송하고 TX trace에 기록한다.	TcpSessionError, buildEthernetFrame, self.tx_packets.append, sendFrame, sendPacket
RawTcpSession._send_control _send_control(self, flags: Union[int, str], *, seq: Optional[int]=None, ack: Optional[int]=None, payload: Optional[bytes]=None, options: Optional[List[Tuple[str, Any]]]=None) -> None L684-707	method	현재 또는 지정된 SEQ/ACK로 ACK/SYN/FIN/RST/data 세그먼트를 구성해 전송한다.	TcpSessionError, self._build_ip_tcp, self._send_ip_tcp
RawTcpSession.connect connect(self) -> 'RawTcpSession' L709-803	method	RST guard와 sniffer를 준비하고 SYN, SYN/ACK, ACK를 수행해 ESTABLISHED로 전환한다.	TcpResetError, TcpSessionError, TcpTimeoutError, _OsRstGuard, _seq_add, self._build_ip_tcp, self._cleanup_resources, self._log, self._packet_queue.get, self._prepare_route 외 4개
RawTcpSession._drain_pending_segments _drain_pending_segments(self) -> None L805-846	method	현재 RCV.NXT에 이어지는 out-of-order segment와 FIN을 반복 병합한다.	TcpSessionError, _seq_add, _seq_signed_delta, self._receive_buffer.extend
RawTcpSession._accept_server_segment _accept_server_segment(self, seq: int, payload: bytes, fin: bool) -> bool L848-891	method	중복/겹침을 잘라내고 연속 데이터는 버퍼에 놓으며 미래 데이터는 pending에 보관한다.	TcpSessionError, _seq_add, _seq_signed_delta, self._drain_pending_segments, self._pending_fin_sequences.append, self._pending_segments.append, self._receive_buffer.extend
RawTcpSession._process_packet _process_packet(self, packet: Packet) -> Dict[str, bool] L893-997	method	RST/ACK/payload/FIN을 상태에 반영하고 허용 ACK만 SND.UNA에 적용하며 필요한 누적 ACK를 전송한다.	TcpResetError, _seq_ge, _seq_signed_delta, self._accept_server_segment, self._log, self._send_control
RawTcpSession._wait_for_ack _wait_for_ack(self, target_ack: int, timeout: float) -> bool L999-1020	method	큐의 패킷을 처리하면서 SND.UNA가 목표 ACK에 도달할 때까지 기다린다.	_seq_ge, self._packet_queue.get, self._process_packet
RawTcpSession.send_bytes send_bytes(self, data: Union[bytes, bytearray, memoryview]) -> None L1022-1092	method	payload를 MSS 단위로 전송한다. SND.NXT는 전송 직후 이동하고 재전송은 고정 segment_seq를 사용한다.	TcpSessionError, TcpTimeoutError, _seq_add, self._log, self._send_control, self._wait_for_ack

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
RawTcpSession.receive_until receive_until(self, completeCheck: Optional[Callable[[bytes], bool]]=None, *, totalTimeou t : float=5.0, idleTimeout: Optional[float]=None) -> _TcpReceiveResult L1094-1151	method	재조립 스트림이 완료 조건을 만족하거나 FIN/타임아웃이 발생할 때까지 수신한다.	TcpSessionError, _TcpReceiveResult, self._packet_queue.get, self._process_packet
RawTcpSession.close close(self, timeout: Optional[float]=None, *, requireComplete: bool=False) -> Dict[str, Any] L1153-1283	method	active/passive FIN 흐름을 처리하고 양쪽 FIN/ACK 완료 여부와 trace를 반환한다.	TcpSessionError, TcpTimeoutError, _seq_add, self._cleanup_resources, self._packet_queue.get, self._process_packet, self._send_control
RawTcpSession.abort abort(self) -> None L1285-1305	method	가능하면 RST를 보내고 sniffer와 방화벽 규칙을 즉시 정리한다.	self._cleanup_resources, self._send_control
RawTcpSession._cleanup_resources _cleanup_resources(self) -> None L1307-1314	method	sniffer와 RST guard를 한 번만 정리한다.	self._rst_guard.disable, self._stop_sniffer
RawTcpSession.snapshot snapshot(self) -> Dict[str, Any] L1316-1358	method	상태, 5-tuple, SND.NXT/SND.UNA/RCV.NXT, FIN 플래그, 재전송과 패킷 trace를 딕셔너리로 반환한다.	-
RawTcpSession.__enter__ __enter__(self) -> 'RawTcpSession' L1360-1361	method	with 진입 시 connect를 수행한다.	self.connect
RawTcpSession.__exit__ __exit__(self, exc_type: Any, exc: Any, traceback: Any) -> bool L1363-1368	method	정상 종료면 close, 예외면 abort를 수행한다.	self.abort, self.close
openTcpSession openTcpSession(dstIp: str, dstPort: int=80, **sessionOptions: Any) -> RawTcpSession L1374-1388	function	3-way handshake까지 완료된 유지형 RawTcpSession을 반환한다.	RawTcpSession, RawTcpSession().connect
do4wayHandshake do4wayHandshake(session: RawTcpSession, timeout: Optional[float]=None, *, requireComplete: bool=False) -> Dict[str, Any] L1391-1400	function	전달받은 RawTcpSession의 close를 호출하는 호환 편의 함수.	session.close
do3wayHandshake do3wayHandshake(dstIp: str, dstPort: int, timeout: int=2) -> Optional[Dict[str, Any]] L1406-1472	function	기존 호출부 호환용 단발성 3-way handshake를 수행하고 5-tuple/SEQ/ACK를 반환한다.	TcpSessionError, TcpTimeoutError, _seq_add, buildIpPacket, buildTcpSegment, getFreePort, getIfaceIp, getMainiface, rSendPacket, sendPacket

8.4 IP_pac.py

역할: 사용자 지정 IPv4 헤더와 상위 payload를 결합하는 L3 빌더.

직접 import: scapy.all (IP, IPOption, Packet, Raw); typing (Union, Optional); validation (isValidIpv4)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
buildIpPacket buildIpPacket(srcIp: str, dstIp: str, ver: int=4, ihl: Optional[int]=None, tos: int=0, total_len: Optional[int]=None, id: int=1, flags: Union[int, str]=0, frag: int=0, ttl: int=64, protocol: Union[int, str]='tcp', checksum: Optional[int]=None, options: Optional[bytes]=None, payload: Union[str, bytes, Packet, None]=None) -> IP L36-118	function	IPv4 헤더 필드, 옵션, payload를 Scapy IP 객체로 조립한다.	IP, IPOption, Raw, isValidIpv4

8.5 Ethernet_pac.py

역할: Ethernet 헤더, payload, 선택적 FCS와 preamble을 결합하는 L2 빌더.

직접 import: zlib; typing (Union, Optional); scapy.all (Packet, Ether, Raw, bytes_encode); validation (isValidMac)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
buildEthernetFrame buildEthernetFrame(dstMac: str, srcMac: str, payload: Union[str, bytes, Packet, None]=None, etherType: Union[int, str, None]=None, fcs: Optional[Union[int, bytes, bool]]=None, preamble: Optional[bytes]=None) -> Ether L16-87	function	Ethernet dst/src/type와 payload를 조립하고 선택적으로 CRC32 FCS와 preamble을 덧붙인다.	Ether, Raw, isValidMac

8.6 Address_resolution.py

역할: ARP Request/Reply로 같은 L2 구간의 대상 또는 게이트웨이 MAC을 확인.

직접 import: __future__ (annotations); typing (Optional); scapy.all (ARP); Ethernet_pac (buildEthernetFrame); getters (getIfaceIp, getIfaceMac, getMainIface); sender (rSendFrame); validation (isValidIpv4)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
getMacByArp getMacByArp(dst_ip: str, iface: Optional[str]=None, timeout: int=2, retry: int=2) -> Optional[str] L13-63	function	브로드캐스트 ARP Request를 전송해 다음 홉 MAC을 재시도 횟수 내에서 반환한다.	ARP, buildEthernetFrame, getIfaceIp, getIfaceMac, getMainIface, isValidIpv4, rSendFrame

8.7 sender.py

역할: Scapy send/sendp/sr/sr1/srp1 호출을 감싸는 실제 전송 어댑터.

직접 import: scapy.all (IP, TCP, srp1, sr1, Ether, Packet, sendp, send); getters (*); typing (Optional); os; sys; platform; atexit; subprocess; socket; signal; scapy.all (sr, Raw, TCP); typing (Optional, Any)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
sendFrame sendFrame(frame: Ether, iface: Optional[str]=None) L12-16	function	Ethernet 프레임을 지정 인터페이스에서 sendp로 전송한다.	getMainIface
sendPacket sendPacket(packet: Packet) L18-19	function	L3 패킷을 Scapy send로 전송한다.	-
rSendFrame rSendFrame(frame: Ether, iface: Optional[str]=None, timeout: int=2) -> Optional[Ether] L21-25	function	Ethernet 프레임을 보내고 첫 L2 응답을 srp1로 기다린다.	getMainIface
rSendPacket rSendPacket(packet: Packet, timeout: int=2) -> Optional[Packet] L27-28	function	L3 패킷을 보내고 첫 응답을 sr1로 기다린다.	-
rSendPacket_HTTP rSendPacket_HTTP(packet: Any, timeout: int=3) -> Optional[Any] L33-49	function	하나의 요청에 대한 여러 응답을 sr(multi=True)로 수신하고 payload가 있는 패킷을 우선 반환한다.	-

8.8 getters.py

역할: 기본 인터페이스, 인터페이스 IP/MAC, 임시 출발지 포트를 조회.

직접 import: scapy.all (conf, get_if_hwaddr, get_if_list, get_if_addr); typing (Optional); socket

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
getMainIface getMainIface() -> str L5-6	function	Scapy conf.iface의 기본 송신 인터페이스를 반환한다.	-
getFreePort getFreePort() -> int L8-11	function	임시 TCP 소켓을 bind하여 OS가 선택한 사용 가능 포트 번호를 얻는다.	-
getIfaceMac getIfaceMac(iface: Optional[str]=None) -> str L13-35	function	인터페이스 존재 여부를 확인하고 할당된 MAC 주소를 반환한다.	getMainIface

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
getIfacelp getIfaceIp(iface: Optional[str]=None) -> str L37-70	function	인터페이스 존재 여부를 확인하고 할당된 IPv4 주소를 반환한다.	getMainIface

8.9 validation.py

역할: IPv4 및 MAC 문자열 유효성 검사.

직접 import: ipaddress; scapy.all (mac2str)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
isValidIpv4 isValidIpv4(ip: str) -> bool L6-14	function	ipaddress 모듈로 IPv4 형식과 버전을 검증한다.	-
isValidMac isValidMac(mac: str) -> bool L16-26	function	Scapy mac2str 변환 결과가 6바이트인지 확인한다.	-

8.10 ATTACK.py

역할: 현재는 DPDK 플러그만 설정하는 미완성/레거시 공격 스텝.

직접 import: scapy.all (*)

심볼 / 시그니처 / 라인	종류	기능	주요 호출 의존
tcpSynFlood tcpSynFlood() L4-5	function	Scapy의 DPDK 사용 플러그만 활성화하는 빈 스텝이다. 실제 flood 로직은 없다.	-

9. 비활성 코드와 외부 의존성

sender.py의 triple-quoted 문자열 내부에는 off0sFirewall()와 signal_handler() 형태의 예전 코드가 남아 있으나 Python AST에는 함수로 정의되지 않으며 실행되지 않습니다. 실제 RST 차단은 TCP_pac._0sRstGuard가 담당합니다.

의존성	사용 목적	필수 조건/주의
Scapy	패킷 객체, send/sendp/sr, AsyncSniffer, route와 인터페이스 조회	requirements.txt: scapy>=2.5,<3
libpcap 또는 Npcap	AsyncSniffer 실제 캡처 백엔드	운영체제별 캡처 권한 필요
root/Administrator	Raw 패킷 송수신과 RST 방화벽 규칙	Linux는 iptables/nft, Windows는 PowerShell firewall
표준 라이브러리	queue, threading, gzip/zlib, http.cookies, ipaddress, socket 등	별도 설치 없음

수정 패키지에 포함된 보조 파일

파일	용도
CHANGELOG_SEQ_ACK.md	원인, 수정 범위와 테스트 명령
VALIDATION_SEQ_ACK.txt	재현 수치와 검증 결과
tests/test_seq_ack_regression.py	네트워크 없이 piggyback ACK 상태를 재현
inventory.json	모든 함수/클래스/메서드/호출 관계의 AST 추출 결과
requirements.txt	Scapy 의존성