Maude를 활용한 TLS 소프트웨어 모델 기반 테스트 프레임워크

포항공과대학교
Software Verification Lab
이재훈

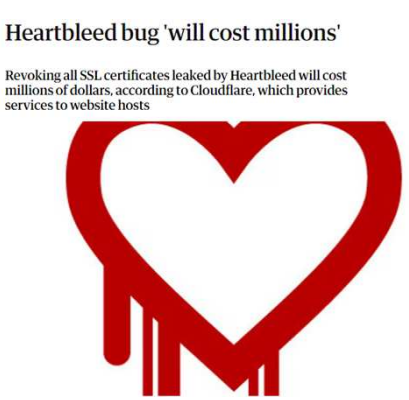
TLS(Transport Layer Security) Protocol

- 두 당사자 간 데이터 보호, 무결성 및 인증을 보장하기 위한 보안 네트워크 프로토콜

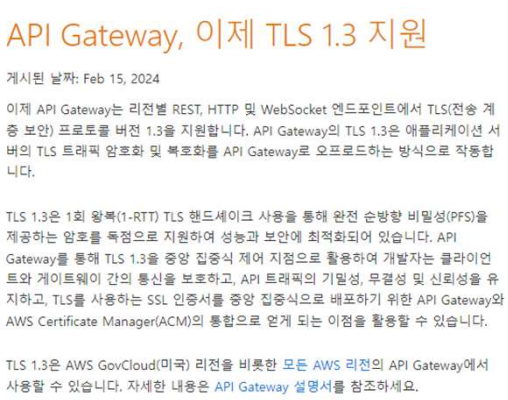
TLS(Transport Layer Security) Protocol

- 두 당사자 간 데이터 보호, 무결성 및 인증을 보장하기 위한 보안 네트워크 프로토콜

- TLS 프로토콜을 기반으로 HTTPS, SSH, SFTP와 같은 많은 보안 응용 프로그램들이 개발되었기 때문에, **검증**이 필요하다.



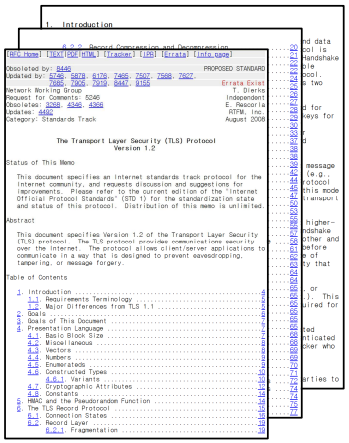
The Guardian



Amazon Web Service

Microsoft

Existing TLS [Verification/Testing]

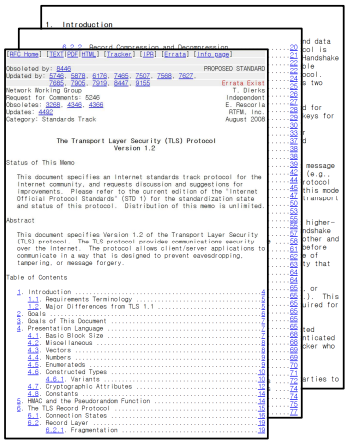


[RFC 5246, RFC 8446]

TLS Protocol Spec에 대한 검증

- TAMARIN, PROVERIF, Maude-NPA, …

Existing TLS [Verification/Testing]



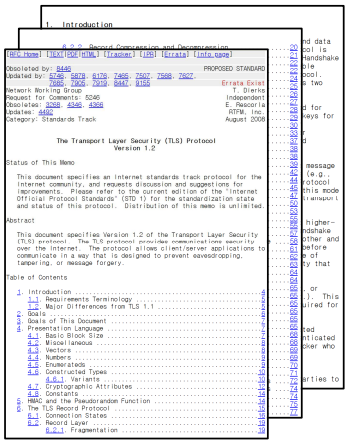
[RFC 5246, RFC 8446]

TLS Protocol Spec에 대한 검증

- TAMARIN, PROVERIF, Maude-NPA, …

TLS Spec에서 검증되었지만, Spec과 Code사이 Gap으로 인해 TLS Code에서 버그 및 취약점이 발생할 수 있음.

Existing TLS [Verification/Testing]



[RFC 5246, RFC 8446]

TLS Protocol Spec에 대한 검증
- TAMARIN, PROVERIF, Maude-NPA, …

TLS Spec에서 검증되었지만, Spec과 Code사이 Gap으로 인해 TLS Code에서 버그 및 취약점이 발생할 수 있음.

OpenSSL Cryptography and SSL/TLS Toolkit

GnuTLS

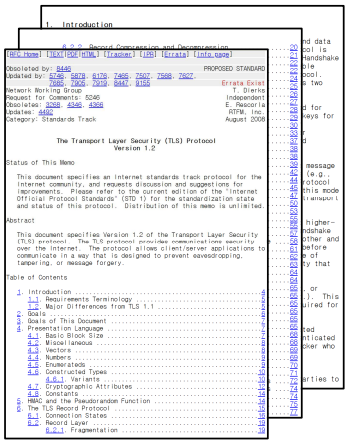
wolfSSL

BoringSSL

[TLS Software Libraries]

TLS Protocol Implementations에 대한 테스트
- Fuzzing/Differential/Combinatorial Tests
- Active/Passive Learning Tests…

Existing TLS [Verification/Testing]



[RFC 5246, RFC 8446]

[TLS Software Libraries]

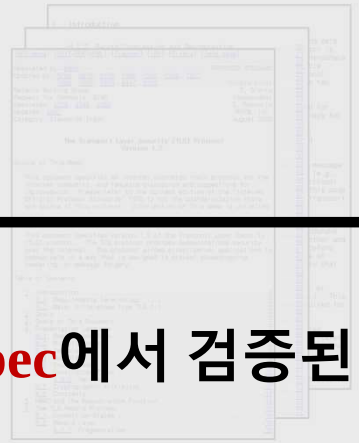
TLS Protocol Spec에 대한 검증
- TAMARIN, PROVERIF, Maude-NPA, …

TLS Spec에서 검증되었지만, Spec과 Code사이 Gap으로 인해 TLS Code에서 버그 및 취약점이 발생할 수 있음.

TLS Protocol Implementations에 대한 테스트
- Fuzzing/Differential/Combinatorial Tests
- Active/Passive Learning Tests…

커버리지를 높일 수 있지만, 기능/보안 요구사항을 확인하는 테스트케이스를 자동으로 생성하는데 어려움이 있음.

Existing TLS [Verification/Testing]



[RFC 5246, RFC 8446]

OpenSSL Cryptography and SSL/TLS Toolkit

GnuTLS

wolfSSL

BoringSSL

[TLS Software Libraries]

- **TLS Spec**에서 검증된 (기능/보안) 요구사항들이 **TLS Code**에서 만족되는지 확인
- 요구사항을 검증할 **자동화된** 테스트 케이스의 생성 및 실행

TLS Protocol Spec에 대한 검증

- TAMARIN, PROVERIF, Maude-NPA

TLS Protocol Implementations에 대한 테스트

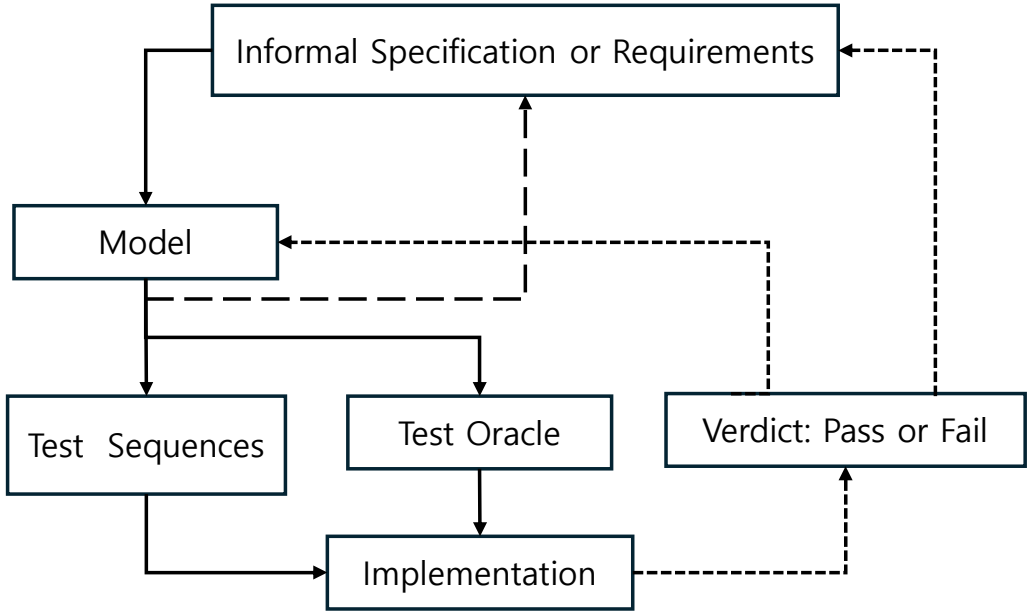
- Fuzzing/Differential/Combinatorial Tests
- Active/Passive Learning Tests

TLS Spec에서 검증되었지만, Spec과 Code사이 Gap으로 인해 TLS Code에서 버그 및 취약점이 발생할 수 있음.

커버리지를 높일 수 있지만, 기능/보안 요구사항을 확인하는 테스트케이스를 자동으로 생성하는데 어려움이 있음.

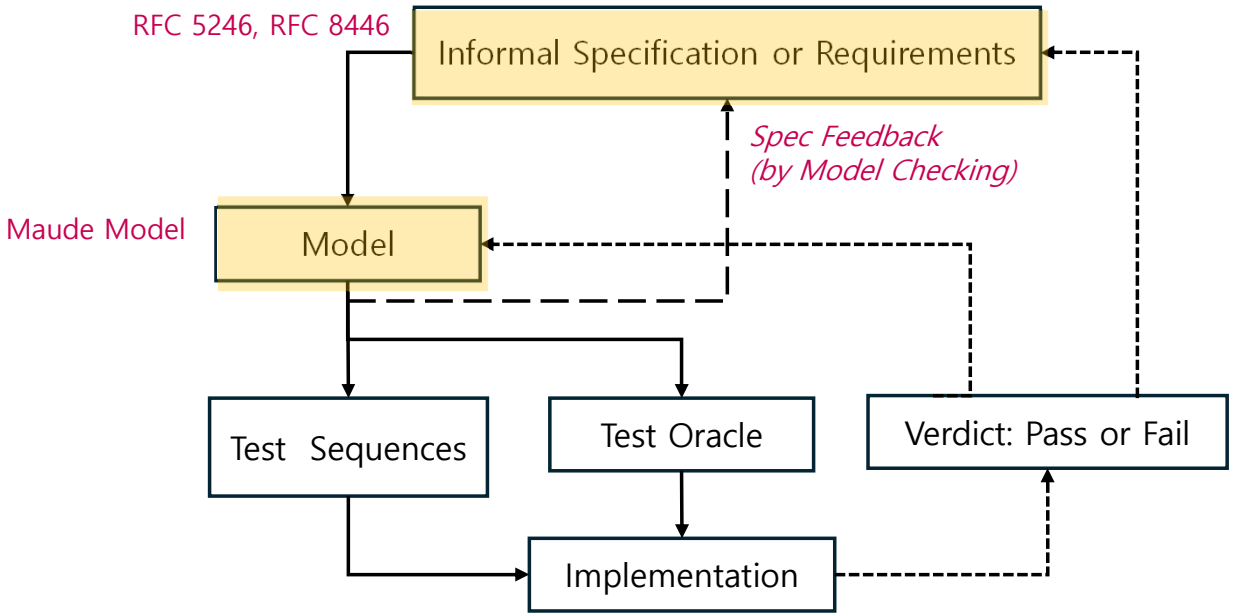
TLS Software Model Based Testing

- TLS Spec을 따르는 정형 모델 생성, 기능/보안 요구사항에 대한 정형검증 수행, 검증 결과로서 모델의 행동 시퀀스를 테스트 케이스로 생성 및 실행



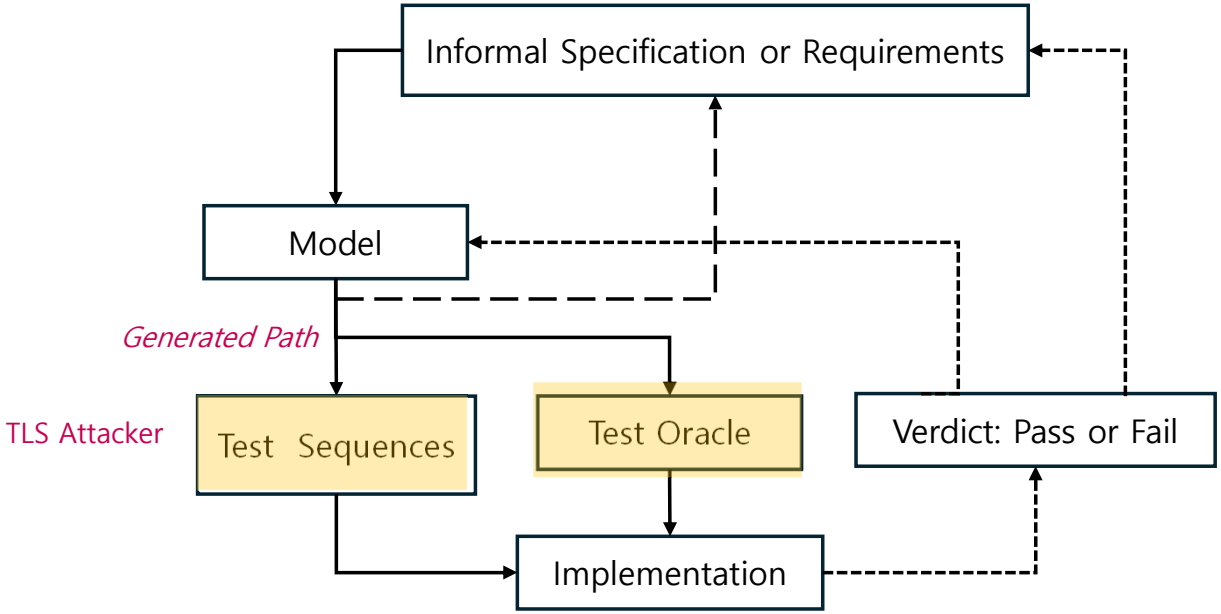
TLS Software Model Based Testing

- TLS Spec을 따르는 정형 모델 생성, 기능/보안 요구사항에 대한 정형검증 수행, 검증 결과로서 모델의 행동 시퀀스를 테스트 케이스로 생성 및 실행



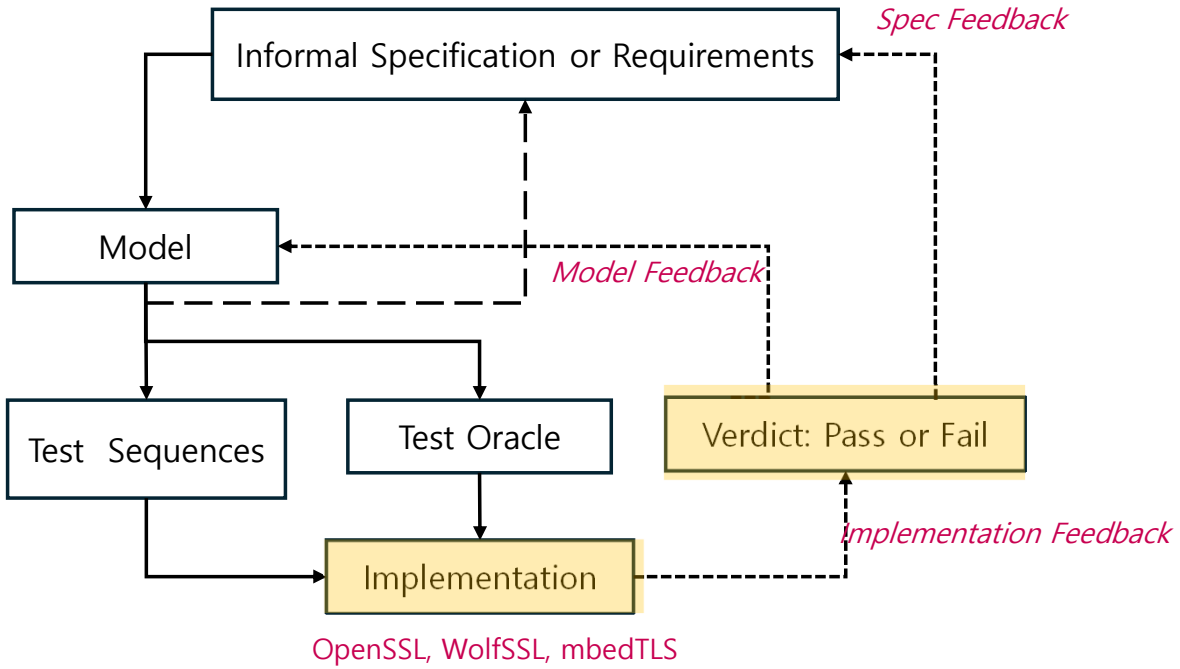
TLS Software Model Based Testing

- TLS Spec을 따르는 정형 모델 생성, 기능/보안 요구사항에 대한 정형검증 수행, 검증 결과로서 모델의 행동 시퀀스를 테스트 케이스로 생성 및 실행



TLS Software Model Based Testing

- TLS Spec을 따르는 정형 모델 생성, 기능/보안 요구사항에 대한 정형검증 수행, 검증 결과로서 모델의 행동 시퀀스를 테스트 케이스로 생성 및 실행



Formal Specification using Maude

- TLS Spec (RFC 5246, RFC 8446)와 TLS Software Library를 참조하여 서버와 클라이언트의 행동을 Maude 기반 정형 모델로 생성

```
Class Component | pVersion : ProtocolVersion,
  cipher : CipherSuite,      nonceCtr : Nat,
  pubKeys : Set{AsyncKey},   symKey : SymKey?,
  prvKeys : Set{AsyncKey},   peerRandom : Nonce?,
  preMasterSecret : Nonce?,  masterSecret : Nonce?,
  certs : Set{Certificate},  preferedCA : Set{Tid},
  cIBuf : TLSMsgList,        cOBuf : TLSMsgList,
  Owner : Tid,  peer : Tid,  currAPI : API? .
```

Client와 Server의 동일한 부분을 나타내는 상위 클래스인 Component 클래스 정의

```
rl [sendClientHello]:
   < CI : Client | connectState : V2C-INIT,
     nonceCtr : N, pVersion : PV, cipher : CS,
     clientRandom : noNonce,
     cOBuf : MSGS >
=> < CI : Client | connectState : V2C-CT-HLO,
     nonceCtr : s N,
     clientRandom : nonce(CI, N),
     cOBuf : MSGS ::
             (clientHello(PV, CS, nonce(CI,N))) >
```

TLS 소프트웨어 암호화 통신 중 ClientHello 메시지 송신 정의

Abstract Test Sequence Generation

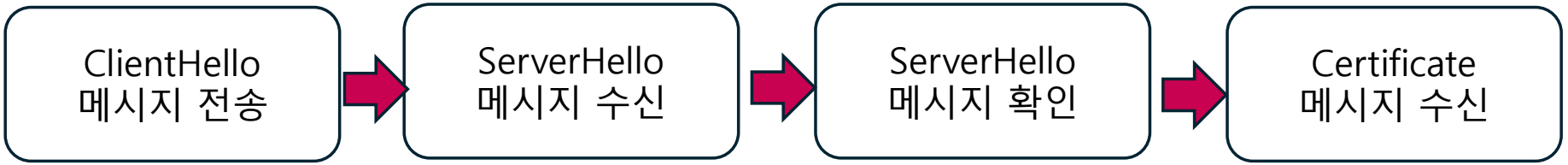
- Maude Search 명령어를 활용 기능/보안 요구사항에 대한 정형 검증 수행
  - CertificateRequest 메시지를 수신 받으면, Certificate 메시지를 송신
  - 공격자가 있는 상황에서 RSA-EXPORT와 같은 취약한 알고리즘을 사용 불가
  - 서버가 empty Certificate 메시지를 클라이언트에게 송신 불가

```
Maude> search [1] init =>* < C1 : Server | cipher : (RSA-EXPORT CS1), ATTS1 >
                            < C2 : Client | cipher : (RSA-EXPORT CS2), ATTS2 > CONF .

Solution 1 (state 819)
states: 820 rewrite : 5096 in 18092ms cpu (18092ms real) (21474 rewrites/second)
…
```

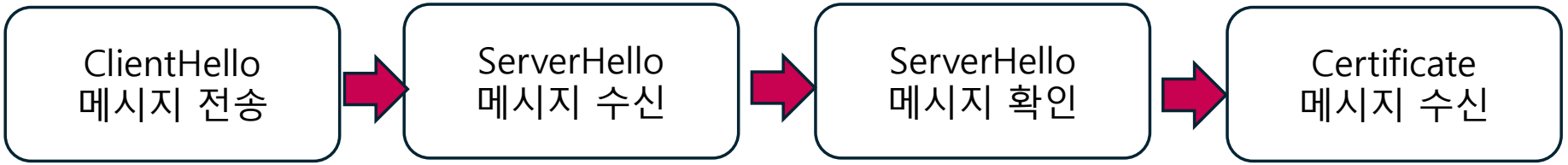
Test Concretization

- 검증 결과인 정형 모델의 행동 시퀀스를 코드 수준의 테스트 케이스로 생성 (TLS-Attacker 사용)



Test Concretization

- 검증 결과인 정형 모델의 행동 시퀀스를 코드 수준의 테스트 케이스로 생성 (TLS-Attacker 사용)



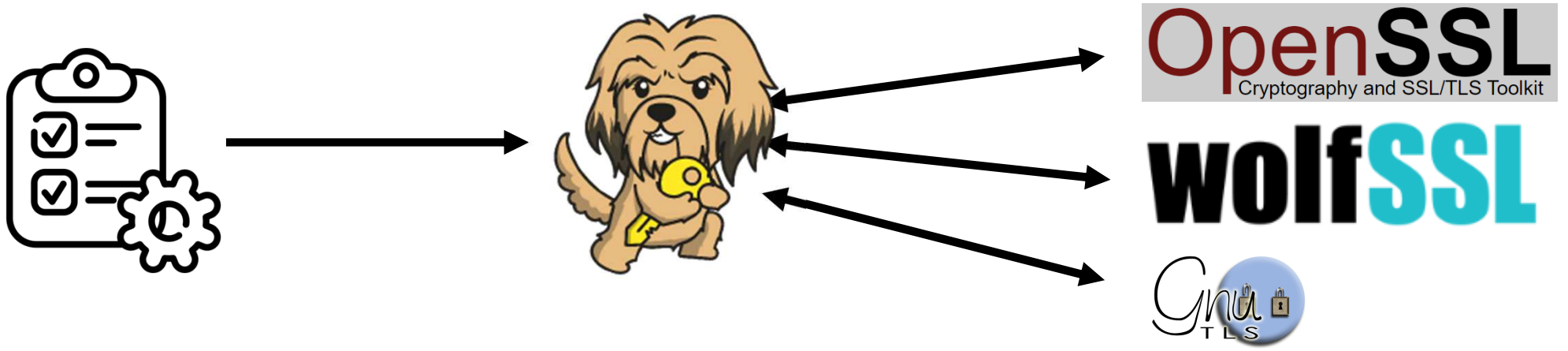

```
ProtocolMessage v0 = genClientHello(TLS12, ECDHE_ECDSA_AES, NONCE, NONCE, …)
send(v0);

ProtocolMessage v1 = recv();
AssertEquals(v1.handshakeType = HANDHSAKE);
AssertEquals(v1.server_version = TLS12);

ProtocolMessage v2 = recv();
```

Test Execution

- 생성된 테스트 케이스를 사용하여 TLS Software Libraries들과 통신하며 테스트를 진행



Evaluation Plan

- 정상 상황에서 TLS Spec 기능/보안 요구사항 자동 테스트

  TLS Spec에서 제시한 기능 요구사항과 기존 TLS Fuzzing 연구들에서 사용한 간단한 보안 요구사항들에 대한 테스트 진행

Evaluation Plan

- 정상 상황에서 TLS Spec 기능/보안 요구사항 자동 테스트

  TLS Spec에서 제시한 기능 요구사항과 기존 TLS Fuzzing 연구들에서 사용한 간단한 보안 요구사항들에 대한 테스트 진행

- MITM 상황에서 기능/보안 요구사항 자동 테스트

  네트워크 상에서 메시지를 생성/수정/삭제/지연시킬 수 있는 공격자가 있을 때, TLS의 기능/보안 요구사항 테스트 진행

Evaluation Plan

- 정상 상황에서 TLS Spec 기능/보안 요구사항 자동 테스트

  TLS Spec에서 제시한 기능 요구사항과 기존 TLS Fuzzing 연구들에서 사용한 간단한 보안 요구사항들에 대한 테스트 진행

- MITM 상황에서 기능/보안 요구사항 자동 테스트

  네트워크 상에서 메시지를 생성/수정/삭제/지연시킬 수 있는 공격자가 있을 때, TLS의 기능/보안 요구사항 테스트 진행

- 에러 상황에서 기능/보안 요구사항 자동 테스트

  서버 혹은 클라이언트가 잘못 행동했을 때, 연결 대상이 에러 처리를 제대로 처리하는지 테스트 진행

Maude를 활용한 TLS 소프트웨어 모델 기반 테스트 프레임워크

포항공과대학교
Software Verification Lab
이재훈