2022/11/09~2022/11/16

Openssl 속도 측정 도구 개발 (AES vs Aria)

SESLab 이성진

차례

1. 서론

한국 전력에서 새롭게 네트워크 망을 설치하면서 기존에 사용하던 Aria 암호화 모듈의 사용이 불가할 것 같다는 의견이 제시되었다. 이에 따라 Openssl에서 지원하는 AES 암호화 모듈을 사용하여 Server/Client 간의 데이터 송수신 시간을 측정했을 때, Aria보다 사용 효율이 높을 것인가에 대해 알아본다.

2. 설치 및 빌드

01

설치

실험을 위한 2개의 가상머신을 생성한다. 지원하는 OS는 “kali 2022.03”, “Ubuntu20.04” 2개의 환경에서 지원한다(윈도우는 따로 테스트해보지 않았음). 함께 첨부한 파일 2개를 다운로드 받고 README.md 문서를 참고하여 환경 설정을 완료한다(README.md를 확인할 수 있는 프로그램이 없다면 아래 표의 URL을 참고).

02

빌드

opensocket 파일은 README.md에 기술한 대로 빌드하고, tlstimer의 경우 tcpclient, tcpserver, tlsclient, tlsserver 4개의 디렉토리에 들어가 make 명령어를 입력하여 빌드한다.

프로그램 이름	설명	URL
opensocket	TCP, UDP, TLS(SSL) 과 관련된 모든 소켓 프로그래밍에 사용하는 외부 라이브러리로 이전에 직접 개발함	https://github.com/wifievent/opensocket/tree/develop
tlstimer	TLS 및 TCP 환경에서 server/client를 생성하여 테스트할 수 있는 파일로 인증서 파일은 모두 임의로 생성해놓음 사용 가능한 AES 및 ARIA 버전은 README.md 참고	https://github.com/maro5397/tlstimer

3. 사용 방법

tlstimer를 사용하여 TLS환경을 테스트해보고 싶다면 tlsserver/tlsclient를 빌드하여 사용하면 되고, TCP환경을 테스트해보고 싶다면 tcpserver/tcpclient를 빌드하여 사용하면 된다.
명령어는 아래와 같다. 해당 명령어는 tlstimer의 README.md를 참고하거나 [2. 설치 및 빌드]의 URL을 참고하라.

01

인증서 생성 방법

1) 아래의 명령어를 사용하여 암호화에 사용될 키를 생성한다.

$ openssl genrsa [-"암호화알고리즘"] -out ["키이름"] ["키길이"]

- 암호화 알고리즘: -aes128, -aes192, -aes256, -aria128, -aria192, -aria256, -camellia128, -camellia192, -camellia256, -des, -des3, -idea

ex)

$ openssl genrsa -aes128 -out server.key 4096
$ openssl genrsa -aes128 -out server.key 2048

2) 키를 생성했다면 아래 명령어를 입력하여 인증서를 생성한다.

$ openssl req -new -key server.key -out server.csr
$ cp server.key server.key.origin
$ openssl x509 -req -days 3650 -in server.csr -signkey server.key -out server.crt
$ openssl rsa -in server.key -text] key.pem
$ openssl x509 -inform PEM -in server.crt] crt.pem
$ openssl pkcs12 -export -in server.crt -inkey server.key -out cert.p12
$ openssl pkcs12 -in cert.p12 -nodes -out cert.pem

3. 사용 방법

tlstimer를 사용하여 TLS환경을 테스트해보고 싶다면 tlsserver/tlsclient를 빌드하여 사용하면 되고, TCP환경을 테스트해보고 싶다면 tcpserver/tcpclient를 빌드하여 사용하면 된다.
명령어는 아래와 같다. 해당 명령어는 tlstimer의 README.md를 참고하거나 [2. 설치 및 빌드]의 URL을 참고하라.

02

tlsserver/tlsclient 사용방법

1) tlsserver 사용 명령어
$ ./main [port] [ciphersuite] [symmetric type] [certificate type] [view] [save]
ex)
$ main 8080 AES128-GCM-SHA256 aes128 2048 0 0
$ main 8080 AES128-GCM-SHA256 aes128 4096 0 0

2) tlsclient 사용 명령어
$ ./main [ip] [port] [ciphersuite] [filename] [count] [view] [delay]
ex)
$ main 127.0.0.1 8080 AES128-GCM-SHA256 testdata.txt 1 0 0
$ main 127.0.0.1 8080 AES128-GCM-SHA256 testdata.txt 1 0 0

reference)

- ip: 서버 아이피
- port: 서버 포트
- ciphersuite: 사용할 cipher
  - ARIA128-GCM-SHA256
  - ARIA256-GCM-SHA384
  - AES256-SHA256
  - AES128-SHA256
  - AES128-GCM-SHA256
  - AES128-CCM8
  - AES128-CCM
  - AES256-GCM-SHA384
  - AES256-CCM8
  - AES256-CCM
- symmectric type: 암호화 알고리즘
- certificate type: 키길이
- view: 송수신 데이터 출력(0: 출력X, 1: 출력O)
- save: 송수신 데이터 저장(0: 저장X, 1: 저장O)
- filename: 송신할 파일 이름
- count: 파일 송신 횟수
- delay: 소켓이 송신 후 대기 시간

3. 사용 방법

tlstimer를 사용하여 TLS환경을 테스트해보고 싶다면 tlsserver/tlsclient를 빌드하여 사용하면 되고, TCP환경을 테스트해보고 싶다면 tcpserver/tcpclient를 빌드하여 사용하면 된다.
명령어는 아래와 같다. 해당 명령어는 tlstimer의 README.md를 참고하거나 [2. 설치 및 빌드]의 URL을 참고하라.

03

tcpserver/tcpclient 사용방법

1) tcpserver 사용 명령어
$ ./main [port] [view] [save]
ex)
$ main 8080 0 0

2) tcpclient 사용 명령어
$ ./main [ip] [port] [filename] [count] [view] [delay]
ex)
$ main 192.168.1.2 8080 testdata.txt 30 0 0

reference)

- ip: 서버 아이피
- port: 서버 포트
- view: 송수신 데이터 출력(0: 출력X, 1: 출력O)
- save: 송수신 데이터 저장(0: 저장X, 1: 저장O)
- filename: 송신할 파일 이름
- count: 파일 송신 횟수
- delay: 소켓이 송신 후 대기 시간

4. 테스트 실행 결과

[환경]

1. OS - kali 2022.03
2. 데이터 크기: 15,470,595bytes (15.47MB)
2. 동일한 가상머신에서 클라리언트 서버를 모두 열어둠
3. 30번 보냈을 때 평균 값임
4. 연결 성립 시간은 제외함

[명령어]

1) tlsserver

```
$ ./main 8080 AES128-GCM-SHA256 aes128 2048 0 0
$ ./main 8080 AES128-GCM-SHA256 aes128 4096 0 0
$ ./main 8080 AES256-GCM-SHA384 aes256 2048 0 0
$ ./main 8080 AES256-GCM-SHA384 aes256 4096 0 0
$ ./main 8080 ARIA128-GCM-SHA256 aria128 2048 0 0
$ ./main 8080 ARIA128-GCM-SHA256 aria128 4096 0 0
$ ./main 8080 ARIA256-GCM-SHA384 aria256 2048 0 0
$ ./main 8080 ARIA256-GCM-SHA384 aria256 4096 0 0
```

2) tlsclient

```
$ ./main 192.168.1.2 8080 AES128-GCM-SHA256 longdata.txt 30 0 0
$ ./main 192.168.1.2 8080 AES256-GCM-SHA384 longdata.txt 30 0 0
$ ./main 192.168.1.2 8080 ARIA128-GCM-SHA256 longdata.txt 30 0 0
$ ./main 192.168.1.2 8080 ARIA256-GCM-SHA384 longdata.txt 30 0 0
```

3) tcpserver

```
$ ./main 8080 0 0
```

4) tcpclient

```
$ ./main 192.168.1.2 8080 longdata.txt 30 0 0
```

4. 테스트 실행 결과

ARIA128-GCM-SHA256

- key length: 2048
  - 0.812383s
- key length: 4096
  - 0.826293s

ARIA256-GCM-SHA384

- key length: 2048
  - 0.86746s
- key length-4096
  - 0.878648s

AES128-GCM-SHA256

- key length: 2048
  - 0.687952s
- key length: 4096
  - 0.673881s

AES256-GCM-SHA384

- key length: 2048
  - 0.651215s
- key length: 4096
  - 0.656527s

None(TCP)

- 0.0411107s

5. 결론

테스트 결과, ARIA가 AES보다 0.2초 가량 늦는 것으로 측정된다. 즉, Openssl에서 제공하는 Aria 암호화 보다 Openssl에서 제공하는 AES 암호화의 속도가 좀 더 빠른 것을 확인하였다. 암호화 과정이 잘못 설정되었는지 확인해보기 위해 wireshark 툴을 사용하여 ciphersuite를 확인해본 결과 설정한 ciphersuite로 올바르게 송수신한 것을 확인할 수 있었다. 이에 대해 조사해본 결과 다음과 같은 이유로 위와 같은 결과가 도출될 수 있다는 점을 알 수 있었다.

1. 일반적으로 ARIA가 AES보다 빠르게 만들어졌다고 주장하지만 SW 구현 시 실제 구현 방법에 따라 성능이 예상 외로 나올 수 있다.
2. 환경에 따라 부채널 공격에 대한 카운터메저 삽입 등 변수가 있을 수 있고, 소켓 통신 부하 자체가 성능 측정에 포함된다면 암호 알고리즘의 성능에 영향을 주는 비율이 낮을 수 있는데, 이런 부분들을 확인해야 할 필요가 있다.
3. 제대로 된 측정을 위해서는 다른 환경적 변수들이 최대한 제거되어야 하고, 아주 대용량의 데이터 암호화 또는 많은 반복 횟수가 필요하다.
4. AES는 전 세계 구현자들을 통해 현재도 활발히 연구되고 가속기 등도 존재하기 때문에 AES를 고속 연산하는 방법이 존재한다.

2, 3번의 경우 동일한 환경에서 실험을 진행했으나, 반복 횟수를 30번으로 설정했기 때문에 더 많은 횟수로 반복하면 예측한 결과가 나올 수 있을 것이다. 그럼에도 불구하고 위 실험과 동일한 결과가 도출될 경우 1, 4번 이유에 따라 ARIA보다 AES의 속도가 더 빠르다고 결론 지을 수 있을 것이다.

감사의 인사

해당 프로젝트를 SESLab에 맡겨주셔서 감사드립니다. 특히 아래 언급된 분들에게 감사의 인사를 드립니다. 앞으로도 안전한 온라인 환경을 조성하기 위해 항상 노력하겠습니다. 앞으로도 SESLab과 긴밀한 연락을 통해 다른 여러 프로젝트들을 원활하게 수행할 수 있길 바랍니다.

세종대학교 송재승 교수님
한국전력 김영현 부장님
TTA 전숙현 책임님

연락처

SESLab Daeyang A.I Center, 7th Floor #728
209 Neungdong-ro Gwangjin-gu, Seoul, Republic of Korea
010-5397-5236
lsj000424@gmail.com