ЯЗЫК PDA ДЛЯ ДИНАМИЧЕСКОГО АНАЛИЗА КРИПТОГРАФИЧЕСКИХ ПРОТОКОЛОВ

Бабенко Л.К.[1], Писарев И.А.[2]

Цель статьи: *разработка алгоритма динамического анализа исходных кодов криптографических протоколов с помощью языка PDA для возможности использования собственных моделей атак.*

Метод: *использован метод генерации исходного кода для моделирования стороны злоумышленника при передаче сообщений между легальными сторонами согласно модели Долева-Яо. Также использован метод ложного завершения, который используется в динамическом анализе и позволяет обнаруживать атаки при моделировании.*

Результаты:

В данной работе представлен язык PDA для динамического анализа исходных кодов криптографических протоколов. Описан подход к динамическому анализу, основанный на принципе ложного завершения. Приведен процесс моделирования активной атаки злоумышленника. Описаны элементы языка PDA и приведен пример описания тестового протокола на данном языке. Реализован тестовый протокол на языке программирования C#. Проведено тестирование эффективности работы динамического анализа с помощью моделирования атаки повтором. Проведена верификация безопасности тестового криптографического протокола с помощью известных средств верификации Scyther и Avispa. Проведено сравнение основных показателей известных средств и предлагаемого авторами динамического анализатора протоколов. Приведены основные преимущества предлагаемого авторами подхода. Описано дальнейшее направление работы.

Ключевые слова: *динамический анализ, криптографические протоколы, исходный код, анализ, dpa, моделирование, криптография, верификация.*

DOI:10.21681/2311-3456-2020-05-19-29

Введение

Верификация криптографических протоколов — актуальная и сложная задача. В связи с развитием различных сервисов электронной коммерции, а также мобильных роботов, беспилотных летательных аппаратов, Интернета вещей, вычислительные ресурсы в которых в большинстве случаев очень ограничены и не всегда возможно использовать хорошо известные протоколы. Любой разработанный протокол должен пройти процедуру верификации. Для оценки безопасности криптографического протокола используются в основном формальные методы проверки. Существуют верификаторы, которые позволяют проверить протокол на наличие уязвимостей и выдать список атак (Scyther [1-2], Avispa [3-4], ProVerif [5-6], CryptoVerif [7-8], Tamarin Prover [9]). Однако, проверка данными верификаторами не гарантирует безопасности реализации протокола на языке программирования. Существуют работы, в которых производится верификация реализаций известных протоколов [10-12]. Однако, в них не предлагается общий подход к применению данных подходов к любому протоколу. Существуют также работы, где такой подход предлагается [13-14]. Описанные там подходы позволяют извлечь структуру криптографических протоколов из исходного кода. В основном в данных работах осуществляется извлечение структуры протоколов, чаще всего на основе аннотаций, оставленных в исходном коде, после чего осуществляется трансляция описания протокола для использования верификатора ProVerif. Также могут быть использованы и другие верификаторы. Данный подход хоть и является более эффективным по сравнению с верификацией безопасности протокола до его реализации, но все же имеет ряд недостатков. Главным недостатком является то, что излеченная структура абстрагируется от некоторых деталей реализации, что влечет собой вероятность не обнаружения атак на протокол. Кроме того, и само ограничение функционала средств верификации не позволяет проверить все возможные воздействия на протокол, которые влекут собой появления атак. Есть ряд работ [15-16], в которых предлагается специальная IDE для создания протоколов. Протоколы проходят верификацию в реальном времени при их написании и после чего можно сгенерировать участки кода на выбранном языке программирования для их использования в системе. У данного подхода также есть недостатки в виде, как уже говорилось ранее, несовершенства средств формальной ве-

1 Бабенко Людмила Климентьевна, доктор технических наук, профессор, Южный Федеральный Университет «ЮФУ», Институт компьютерных технологий и информационной безопасности, г. Таганрог, Россия. E-mail: lkbabenko@sfedu.ru.

2 Писарев Илья Александрович, аспирант, Южный Федеральный Университет «ЮФУ», Институт компьютерных технологий и информационной безопасности, г. Таганрог, Россия.
E-mail: ilua.pisar@gmail.com.

рификации, а также то, что сгенерированные участки кода необходимо встраивать в разрабатываемую систему, что также может привести к ошибкам и уязвимостям в протоколе. Таким образом, проверка исходных кодов криптографических протоколов актуальна и необходим метод, который позволил бы анализировать непосредственно реализации криптографических протоколов. В данной работе предлагается метод динамического анализа, основанный на применении языка PDA и который работает непосредственно с уже реализованным криптографическим протоколом в системе.

1. Динамический анализ

PDA language (protocols dynamic analysis) — язык спецификации для проведения динамического анализа исходных кодов криптографических протоколов. С его помощью можно смоделировать любую возможную атаку и проверить ее на практике в исследуемой системе. Язык представляет собой секции, в которых описывается необходимая для моделирования информация.

Согласно модели Долева-Яо [17] через злоумышленника проходят все передаваемые между легальными сторонами сообщения. В основе динамического анализа лежит принцип ложного завершения. При реализации протоколов в случае нарушения штатного исполнения, например, подмена данных, может быть использован либо повторный запрос передаваемых данных, либо разрыв текущей сессии. В большинстве случаев используется именно разрыв текущей сессии и повторный запуск протокола сначала. Принцип ложного завершения состоит в том, что злоумышленник будет осуществлять модификацию данных и в случае, если протокол корректно исполнится на всех легальных сторонах, то это будет говорить об атаке на данный протокол. Корректным исполнением протокола будут служить метки успешного завершения, которые были описаны в предыдущем пункте. Схема активной атаки злоумышленника представлена на Рис. 1.

В данном случае протокол состоит из трех сообщений и предназначен для взаимодействия двух сторон. Злоумышленник подменяет сообщение 2 на аналогичное сообщение, полученное из предыдущей сессии. Но несмотря на подмену протокол корректно исполняется до конца на обеих сторонах и флаги успешного завершения устанавливаются в True. Данное сообщение может являться, как и частью аутентификации сторон, что, к примеру, позволит злоумышленнику аутентифицироваться вместо легальной стороны, так и содержать некоторые смысловые данные, которые будут использоваться в системе и впоследствии могут привести к некорректному ее поведению.

Запуск процесса динамического анализа на примере общения двух сторон выглядит следующим образом:

1. Процесс злоумышленника I запускает процессы клиента A и сервера B. На клиенте A используется циклическая попытка подключиться к серверу с локальным IP адресом и некоторым портом port1. Сервер B открывает соединение с локальным IP адресом и некоторым портом port2 и ожидает подключение клиента.

2. Далее моделируется атака человек по середине. Злоумышленник подключается как клиент к локальному адресу и порту port2 и начинает общение с сервером B, а также открывает соединение с локальным адресом и портом port1 и ждет пока клиент A подключится. По умолчанию на стороне злоумышленника все пересылаемые через него сообщения пересылаются в таком же виде.

3. Согласное модели атаки осуществляется замена элементов сообщений, порядка пересылки, блокирование каналов и т.д.

4. По окончанию исполнения протокола проверяются флаги успешного завершения. Если они оба присутствуют, это будет свидетельствовать об успешности текущей моделируемой атаки. О защищенности от атаки будет свидетельствовать отсутствие одного из флагов или обоих флагов успешного завершения. В зависимости от реализации конкретной системы активные атаки злоумышленника могут вызывать необработанные исключения. В таком случае при отсутствии одного или обоих флагов успешного завершения будет считаться, что моделируемая атака не прошла успешно, но будет запомнена соответствующая информация о том, что данная атака вызывает нестабильное поведение системы в целом.

2. Язык DPA

PDA language (protocols dynamic analysis) — язык спецификации для проведения динамического анализа исходных кодов криптографических протоколов. С помощью него можно смоделировать любую возможную атаку и проверить ее на практике в исследуемой системе. Язык представляет собой секции, в которых описывается необходимая для моделирования информация.

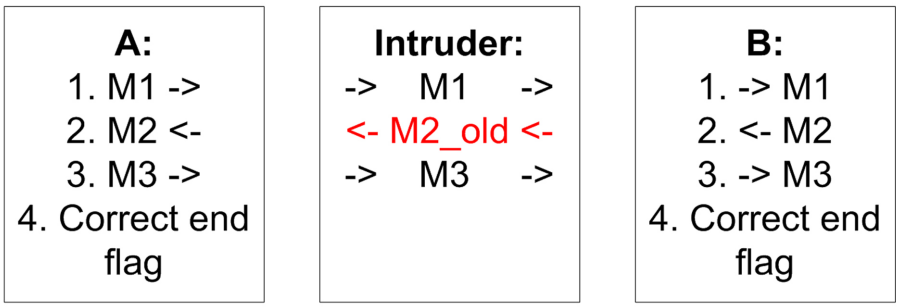

Рис. 1. Пример активной атаки злоумышленника

```
1   Projects
2   {
3    Info1: "project_client", "ProtocolCall(bool mode)", "123", "project_client/
4   protocol.cs"
5    Info2: "project_server", "ProtocolCall(bool mode)" , "223", "project_server/
6   protocol.cs"
7   }
8
9   Parameters
10  {
11   A: Info1
12   B: Info2
13   I: INTRUDER
14  }
15
16  MainProtocol1 //Please use this as base for models
17  {
18   A=>B
19   1 [A = "sertA.dat", Na = RAND, kb = "keyB.dat"] A->B: E_kB(Na,
20  A)
21   2 [B = "sertB.dat", Nb = RAND, ka = "keyA.dat", Na = Get_1_Na] B->A: E_kA(Na,
22  Nb, B)
23   3 [kb = "keyB.dat", Nb = Get_2_Nb] A->B: E_kB(Nb)
24  }
25
26  Model1
27  {
28   A=>I
29   I=>B
30   1 [A = "sertA.dat", Na = RAND, kI = "keyI.dat"] A->I: E_kI(Na,
31  A)
32   2 [A = Get_1_A, Na = Get_1_Na, kb = "keyB.dat"] I->B: E_kB(Na,A)
33   3 [B = "sertB.dat", Nb = RAND, kA = "keyA.dat", Na = Get_2_Na] B->I: E_kA(Na,
34  Nb, B)
35   4 [Mess4 = Get_3_Message] I->A: Mess4
36   5 [kI = "keyI.dat", Nb = Get_3_Nb] A->I: E_kI(Nb)
37   6 [kb = "keyB.dat", Nb = Get_4_Nb] I->B: E_kB(Nb)
38  }
39
40  Run
41  {
42   Model1
    }
```

2.1. Секция Projects

В данной секции (строки 1-7) описывается информация о реализациях протоколов. Изначально нашему динамическому анализатору требуется только информация о путях к проектам сторон. После чего автоматически заполняется данная секция. Вид данной секции, следующий:

```
[Info name]: [project path], [function with protocol], [string number], [file path with protocol]
```

В зависимости от сложности проекта в одном проекте может быть реализовано несколько протоколов. В

данном случае секция будет выглядеть следующим образом в случае, если в первом проекте два протокола, во втором два и в третьем один:

```
      Projects
{
        Info1:         "project_client",
"ProtocolCall(bool     mode)",     "123",
"project_client/protocol.cs"
        Info2:         "project_client",
"ProtocolCall(bool    mode)"    ,    "234",
"project_client /protocol.cs"
        Info3:         "project_server",
"ProtocolCall(bool    mode)"    ,    "223",
"project_server/protocol.cs"
        Info4:         "project_server",
"ProtocolCall(bool    mode)"    ,    "345",
"project_server /protocol.cs"
 Info5: "project_c", "ProtocolCall(bool
mode)" , "445", "project_c/protocol.cs"
      }
```

2.2. Секция Parameters

В данной секции описываются стороны. В параметрах у стороны указывается информация о проекте. Здесь (строки 9-14) указывается 2 проекта легальных сторон A и B и информация о стороне злоумышленнике I.

2.3. Секция MainProtocol1

В данной секции (строки 16-24) описывается основной протокол для системы. Данных секций может быть несколько в случае, если протоколов в системе несколько. Данная секция помогает при составлении модели атаки, поскольку модель должна включать в себя как минимум один основной протокол, дополненный атакой. Протоколы на легальных сторонах неизменны. Единственное что может меняться это передаваемые данные. Их можно менять с помощью загрузки из файлов. В секции изначально описывается порядок соединения сторон (строка 18), после чего описывается порядок передачи сообщений и их содержимое (строки 19-23). Сообщение задается следующим форматом:

```
    Message index [message element =
load path, …] Side1->Side2: Message
```

2.4. Секция Model1

Секция, в которой описывается моделируемая атака. Как говорилось ранее, в моделируемой атаке должен присутствовать полностью основной протокол с самой атакой. В данной секции (строки 28-36) приведена классическая атака Lowe на протокол Нидхема-Шредера [18]. Стоит отметить, что при каждом моделировании создаются копии проектов легальных сторон с замененными загрузками данных в элементы сообщений согласно спецификации модели и метки успешного завершения, а также проект со злоумышленником, в котором генерируются сообщения и загрузка элементов из файлов также согласно спецификации, заданные в текущей секции. Атака считается успешной в случае, если после завершения ее моделирования протокол корректно и до конца исполнился на всех легальных сторонах. Моделей может быть несколько. Все они проверяются по очереди. Возможно также использование параллельных сессий, а также сессий после сессий. Все это можно описать в теле модели. Для этого в секции Parameters нужно создать копии ролей на основе проектов, а также описать порядок передачи сообщений.

```
      Parameters
{
 A1: Info1
 B1: Info2
 A2: Info1
 B2: Info2
 I: INTRUDER
      }
```

В случае если злоумышленник не знает ключа и ему требуется переслать данные просто другой стороне, то для этого используется следующая конструкция:

```
    [Mess4 = Get_3_Message] I->A: Mess4
```

2.5. Секция Run

Запускает моделирование согласно телу области Model.

В результате моделирования выдается результат в виде прошла ли успешно атака или нет:
Run model1— Attack is working
Run model2— Secure

При Run:

Генерируется роль злоумышленника и его исходный код. Пересылка сообщений согласно телу области Model. Урезанной системы больше нет. В местах, которые участвуют в протоколе и требуют ввода динамических данных (загрузка из файла, ввод с клавиатуры) эти места заменяются на загрузку данных из файлов.

3. Тестирование

Для тестирования был взят следующий протокол, основанный на протоколе Нидхема-Шредера:

1. $A \rightarrow B: E_{pkB}(A, Na)$
2. $B \rightarrow A: E_{pkA}(Na, Nb, B)$
3. $A \rightarrow B: E_{pkB}(Nb, k)$
4. $B \rightarrow A: E_{k}(Mess1, Na)$
5. $A \rightarrow B: E_{k}(Mess2, Nb)$

В данном протоколе осуществляется взаимная аутентификация двух сторон, после чего сторона А отсылает сессионный ключ k для дальнейшего использования симметричного шифрования в сообщениях (4) и (5). Данный протокол был реализован на языке программирования C# в виде общения двух компонентов сервер и клиент. Серверная часть выступала в роли стороны B, клиентская— A. Для протокола была составлена следующая модель атаки:

1. $A \rightarrow B: E_{pkB}(A, Na)$
2. $B \rightarrow A: E_{pkA}(Na, Nb, B)$
3. $A \rightarrow B: E_{pkB}(Nb, k)$
4. $B \rightarrow A: E_k(Mess1, Na)$
5. $A \rightarrow I: E_k(Mess2, Nb)$
6. $I \rightarrow B: E_k(Mess1, Na)$

Полное описание модели для проведения динамического анализа на языке PDA представлено ниже:

В результате динамического анализа выяснено, что

```
Projects
{
  Info1: "D:\project_client", "ClientCall()", "145", "D:\project_client\
project_client\Protocol.cs"
  Info2: "D:\project_server", "ServerCall()" , "156", "D:\project_server\
project_server\Protocol.cs"
}

Parameters
{
 A: Info1
 B: Info2
 I: INTRUDER
}

MainProtocol1 //Please use this as base for models
{
 A=>B
 1 [A = "sertA.dat", Na = RAND, kb = "keyB.dat"] A->B: E_kB(Na,
A)
 2 [B = "sertB.dat", Nb = RAND, ka = "keyA.dat", Na = Get_1_Na] B->A:
E_kA(Na, Nb, B)
 3 [kb = "keyB.dat", Nb = Get_2_Nb, k = "key.dat"] A->B:
E_kB(Nb,k)
 4 [k = Get_3_k, M1 = "text1.txt", Na = Get_3_Na] B->A:
E_k(M1,Na)
 5 [k = "key.dat", M2 = "text2.txt", Nb = Get_4_Nb] A->B:
E_k(M2,Nb)
}

Model1
{
 A=>I
 I=>B
 1 [A = "sertA.dat", Na = RAND, kb = "keyB.dat"] A->I: E_kB(Na,
A)
 2 [Mess2 = Get_1_Message] I->B: Mess2
```

DOI:10.21681/2311-3456-2020-05-19-29

```
38   3 [B = "sertB.dat", Nb = RAND, ka = "keyA.dat", Na = Get_2_Na]
39 B->I: E_kA(Na, Nb, B)
40   4 [Mess4 = Get_3_Message] I->A: Mess4
41   5 [kb = "keyB.dat", Nb = Get_4_Nb, k = "key.dat"] A->I:
42 E_kB(Nb,k)
43   6 [Mess6 = Get_5_Message] I->B:
44   7 [k = Get_6_k, M1 = "text1.txt", Na = Get_6_Na] B->I:
45 E_k(M1,Na)
46   8 [Mess8 = Get_7_Message] I->A: Mess8
47   9 [k = "key.dat", M2 = "text2.txt", Nb = Get_8_Nb] A->I:
48 E_k(M2,Nb)
49   10 [Mess10 = Get_7_Message] I->B: Mess8
50 }
51
52 Run
53 {
54   Model1
55 }
56
```

моделируемая атака на протокол прошла успешно. При рассмотрении исходного кода было определено, что в коде отсутствовало действие при проверке возвращаемого случайного числа Na и Nb в сообщениях (4) и (5). В результате этого протокол исполнялся до конца, но при этом вместо требуемых данных в сообщении M2 сторона получила данные M1.

4. Сравнение

Для сравнения предлагаемого подхода были взяты наиболее известные существующие инструменты для проведения верификации безопасности протоколов Scyther и Avispa.

В инструменте Scyther был описан тестируемый протокол и проведена верификация безопасности всех свойств. Описание протокола представлено на Рис. 2.

В результате верификации было установлено, что протокол является безопасным (см. Рис. 3).

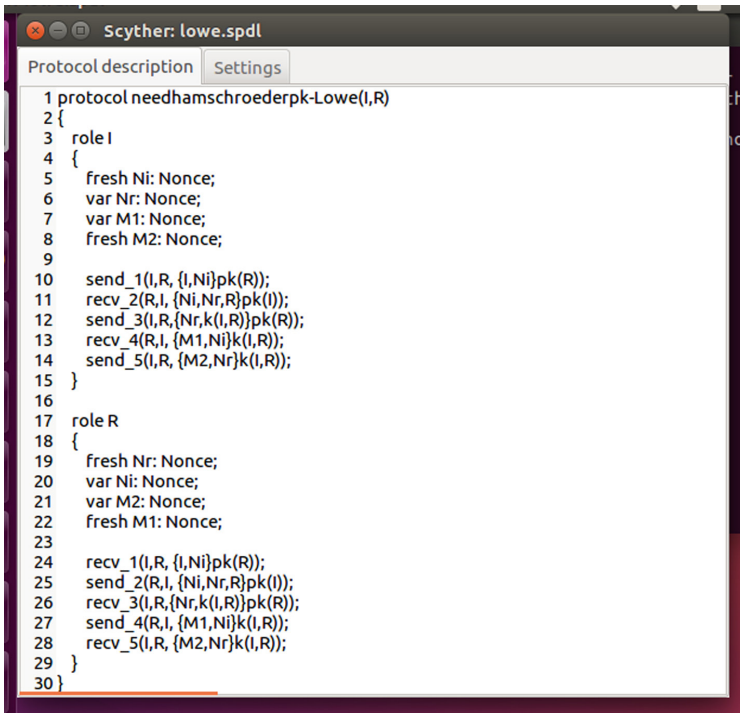

Рис. 2. Описание протокола

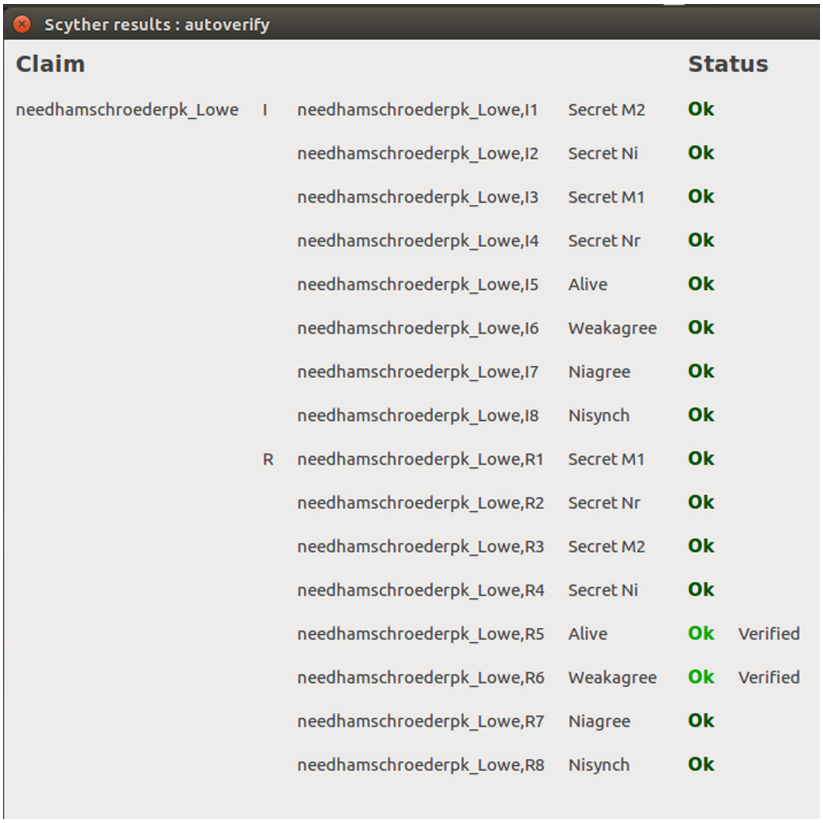

Рис. 3. Результат верификации

Тестовый протокол также был описан на языке CAS+ [19] и транслирован в язык HLPSL [20] для использования в верификаторе Avispa. Avispa— автоматизированный анализатор протоколов безопасности. Поддерживает спецификацию протоколов и свойств безопасности с помощью модульного языка спецификаций. Интегрирует различные back-end, реализующие различные современные методы автоматического анализа для верификации протокола (путем поиска атаки на входной протокол). Методы проверки на основе абстракции как для конечного, так и бесконечного числа сеансов. В ходе анализа использовался режим OFMC [21]. Описание протокола на языке CAS+ представлено на Рис. 4.

В результате верификации было установлено, что протокол является безопасным (см. Рис. 5)

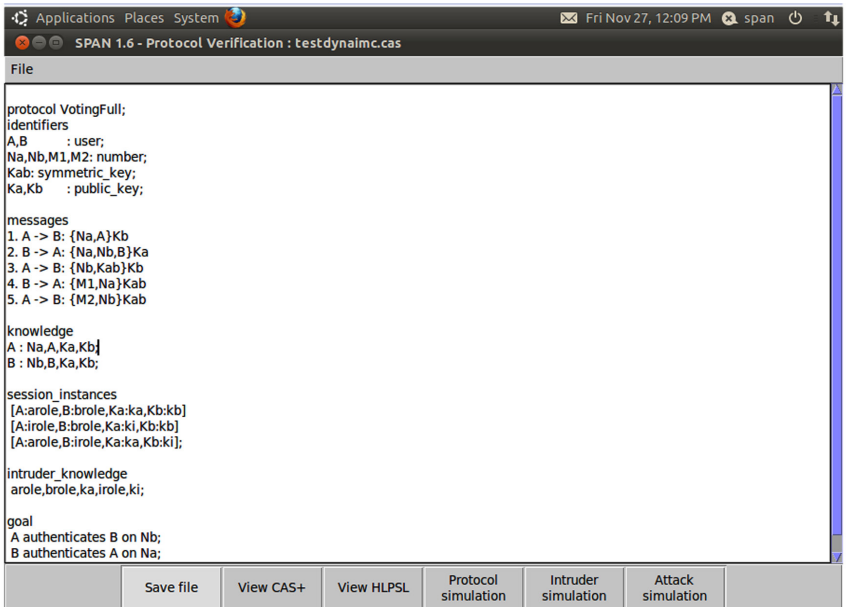

Рис. 4. Описание протокола

DOI:10.21681/2311-3456-2020-05-19-29

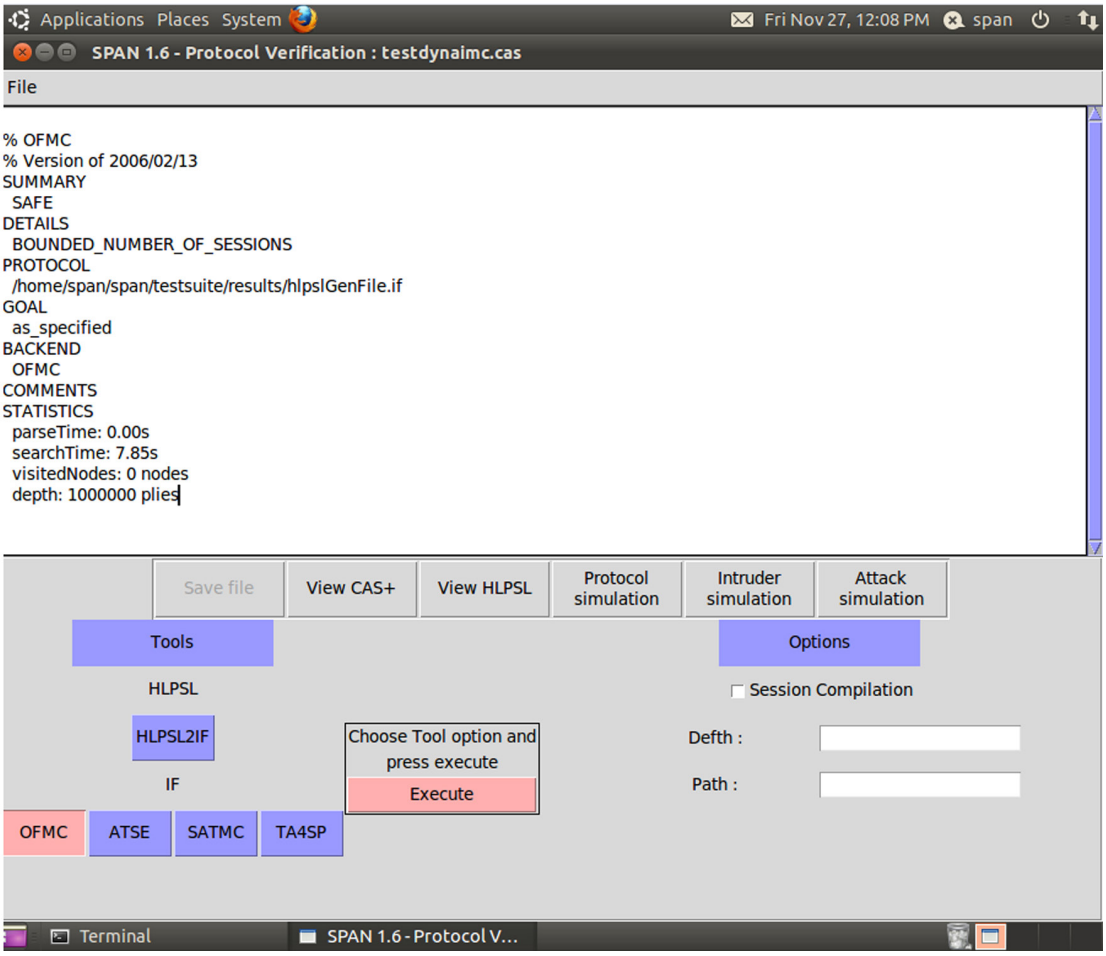

Рис. 5. Результат верификации

Как видно из результатов верификации существующие инструменты не обнаружили уязвимость протокола к атаке повтором, поскольку они не анализируют исходный код. Сам протокол изначально безопасен, однако, отсутствие ключевых проверок в исходном коде делают протокол уязвимым на практике, что получилось обнаружить благодаря предлагаемому авторами подходу.

Было проведено сравнение основных показателей верификаторов Scyther, Avispa и разработанного авторами динамического анализатора PDA. В Табл. 1 приведено сравнение основных показателей. Главными преимуществами предлагаемого авторами динамического анализа с помощью языка PDA являются возможность использования собственных моделей и анализ непосредственно исходного кода криптографического протокола. Также преимуществом является поддержка всех возможных криптографических примитивов и функций, поскольку анализируется непосредственно реализация протокола. Недостатками являются низкая скорость работы даже одной модели, не говоря о том, что для достаточного уровня анализа протокола необходимо большое количество моделей, а также невозможность автоматизированного обнаружения атак, поскольку модели необходимо описывать самостоятельно. Скорость работы в будущем может быть улучшена посредством использования технологий распараллеливания обработки на процессоре, а также путем оптимизации существующих алгоритмов. Касательно автоматизации обнаружения, то планируется создать автоматизированный генератор шаблонов моделей для проверки различных атак.

Таблица

Сравнение основных показателей

Показатель	Scyther	Avispa	PDA
Использование собственных моделей	нет	нет	да
Анализ исходного кода	нет	нет	да

Показатель	Scyther	Avispa	PDA
Скорость работы (на приведенном примере)	1.2 сек	7.8 сек	15 сек (одна модель)
Автоматизированное обнаружение атак	да	да	нет
Поддержка криптографических примитивов и функций	Временные метки, симметричные и асимметричные ключи, подпись, хеширование.	Симметричные и асимметричные ключи, неатомные ключи, ключевые таблицы, ключевое соглашение Диффи-Хеллмана, хэш-функции, алгебраические функции, типизированные и нетипизированные данные, подпись.	Любые

Вывод

В работе представлен язык PDA для динамического анализа исходных кодов криптографических протоколов. Описан подход к динамическому анализу, основанный на принципе ложного завершения. Описаны элементы языка PDA, приведен и проанализирован тестовый протокол. В реализации тестового протокола была обнаружена уязвимость, которая заключалась в отсутствии действия при проверке случайного числа последних сообщений, что приводило к атаке повтором. Проведено сравнение существующих верификаторов с динамическим анализатором, представленным авторами. Существующие верификаторы показали безопасность тестового протокола. Сравнение показало, что предлагаемый авторами подход является наиболее эффективным с точки зрения обнаружения всех возможных потенциальных атак, поскольку осуществляется анализ исходных кодов криптографических протоколов по заданной модели. Также благодаря анализу исходных кодов возможна поддержка всех возможных криптографических функций и примитивов, что позволяет обнаруживать уязвимости в протоколах с их использованием. Однако, время работы относительно других верификаторов дольше. Кроме этого, на данном этапе исследований не представляется возможным автоматизированная верификация исходных кодов, поскольку требуется описывать собственные модели проверки. В дальнейшем авторами планируется уменьшение скорости работы анализатора, а также поддержка возможности автоматизированного применения заранее сгенерированных шаблонов моделей для проверки безопасности криптографических протоколов.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-37-90151.

Литература

1. El Madhoun N., Guenane F., Pujolle G. An online security protocol for NFC payment: Formally analyzed by the scyther tool / 2016 Second International Conference on Mobile and Secure Services (MobiSecServ). IEEE, 2016. Pp. 1-7. DOI:10.1109/MOBISECSERV.2016.7440225
2. Yang H., Oleshchuk V. A., Prinz A. Verifying Group Authentication Protocols by Scyther. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 2016. T. 7. No. 2. Pp. 3-19.
3. Mir O., van der Weide T., Lee C. C. A secure user anonymity and authentication scheme using AVISPA for telecare medical information systems // Journal of medical systems. 2015. T. 39. No. 9. Pp. 89. DOI:10.1007/s10916-015-0265-8
4. Amin R. et al. Design of an enhanced authentication protocol and its verification using AVISPA. 2016 3rd International Conference on Recent Advances in Information Technology (RAIT). IEEE, 2016. Pp. 404-409. DOI:10.1109/RAIT.2016.7507899
5. Cheval V., Cortier V., Turuani M. A little more conversation, a little less action, a lot more satisfaction: Global states in ProVerif. 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018. Pp. 344-358.
6. Chothia T., Smyth B., Staite C. Automatically checking commitment protocols in proverif without false attacks. International Conference on Principles of Security and Trust // Springer, Berlin, Heidelberg, 2015. Pp. 137-155. DOI:10.1007/978-3-662-46666-7-8
7. Blanchet B. Composition theorems for cryptoverif and application to TLS 1.3. 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018. Pp. 16-30. DOI:10.1109/CSF.2018.00009
8. Eswaraiah G., Vishwanathan R., Nedza D. Automated proofs of signatures using bilinear pairings. 2018 16th Annual Conference on Privacy, Security and Trust (PST). IEEE, 2018. Pp. 1-10. DOI:10.1109/PST.2018.8514201
9. Cremers C. Symbolic security analysis using the tamarin prover. 2017 Formal Methods in Computer Aided Design (FMCAD). IEEE, 2017. Pp. 5-5.
10. De Ruiter J., Poll E. Protocol State Fuzzing of {TLS} Implementations. 24th {USENIX} Security Symposium ({USENIX} Security 15). 2015. Pp. 193-206.
11. Cohn-Gordon K. et al. A formal security analysis of the signal messaging protocol. 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017. Pp. 451-466.

12. Beringer L. et al. Verified Correctness and Security of OpenSSL {HMAC}. 24th {USENIX} Security Symposium ({USENIX} Security 15). 2015. Pp. 207-221.
13. Kobeissi N., Bhargavan K., Blanchet B. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017. Pp. 435-450. DOI:10.1109/EuroSP.2017.38
14. Dowling B. et al. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015. Pp. 1197-1210.
15. Garcia R., Modesti P. An IDE for the design, verification and implementation of security protocols. 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2017. Pp. 157-163. DOI:10.1109/ISSREW.2017.69
16. Modesti P. AnBx: Automatic generation and verification of security protocols implementations. International Symposium on Foundations and Practice of Security. Springer, Cham, 2015. Pp. 156-173. DOI:10.1007/978-3-319-30303-1-10
17. Baskar A., Ramanujam R., Suresh S. P. Dolev-Yao theory with associative blindpair operators. International Conference on Implementation and Application of Automata. Springer, Cham, 2019. Pp. 58-69.
18. Szymoniak S., Siedlecka-Lamch O., Kurkowski M. SAT-based verification of NSPK protocol including delays in the network // 2017 IEEE 14th International Scientific Conference on Informatics. IEEE, 2017. Pp. 388-393.
19. Babenko L., Pisarev I. Translation of Cryptographic Protocols Description from Alice-Bob Format to CAS+ Specification Language // International Conference on Intelligent Information Technologies for Industry. Springer, Cham, 2019. Pp. 309-318.
20. Chandre P. R., Mahalle P. N., Shinde G. R. Machine learning based novel approach for intrusion detection and prevention system: A tool based verification // 2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN). IEEE, 2018. Pp. 135-140.
21. Bloemen V., van de Pol J. Multi-core SCC-based LTL model checking //Haifa Verification Conference. Springer, Cham, 2016. Pp. 18-33.

PDA LANGUAGE FOR DYNAMIC ANALYSIS OF CRYPTOGRAPHIC PROTOCOLS

Babenko L.K.[3], Pisarev I.A.[4]

Purpose of the article: *development of an algorithm for dynamic analysis of the source codes of cryptographic protocols using the PDA language for the possibility of using your own attack models.*

Method: *a source code generation method was used to simulate the attacker's side when transmitting messages between legal parties according to the Dolev-Yao model. The method of false termination is also used, which is used in dynamic analysis and allows detecting attacks during simulation.*

Results: *this paper presents the PDA language for dynamic analysis of the source codes of cryptographic protocols. An approach to dynamic analysis based on the principle of false termination is described. The process of modeling an active attack by an intruder is presented. The elements of the PDA language are described and an example of the description of the test protocol in this language is given. A test protocol in the C# programming language has been implemented. The effectiveness of the dynamic analysis was tested by simulating a replay attack. The security verification of the test cryptographic protocol was carried out using the well-known verification tools Scyther and Avispa. The comparison of the main indicators of the known means and the dynamic protocol analyzer proposed by the authors is carried out. The main advantages of the approach proposed by the authors are presented. The further direction of work is described.*

Keywords: *dynamic analysis, cryptographic protocols, source code, analysis, dpa, modeling, cryptography, verification.*

References

1. El Madhoun N., Guenane F., Pujolle G. An online security protocol for NFC payment: Formally analyzed by the scyther tool / 2016 Second International Conference on Mobile and Secure Services (MobiSecServ). IEEE, 2016. Pp. 1-7. DOI:10.1109/MOBISECSERV.2016.7440225
2. Yang H., Oleshchuk V. A., Prinz A. Verifying Group Authentication Protocols by Scyther. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 2016. T. 7. No. 2. Pp. 3-19.

3 Liudmila Babenko, Dr.Sc., Professor, Southern Federal University "SFedU", Institute of Computer Technologies and Information Security, Taganrog, Russia. E-mail: lkbabenko@sfedu.ru.

4 Ilya Pisarev, postgraduate student, Southern Federal University "SFedU", Institute of Computer Technologies and Information Security, Taganrog, Russia. E-mail: ilua.pisar@gmail.com.

3. Mir O., van der Weide T., Lee C. C. A secure user anonymity and authentication scheme using AVISPA for telecare medical information systems // Journal of medical systems. 2015. T. 39. No. 9. Pp. 89. DOI:10.1007/s10916-015-0265-8
4. Amin R. et al. Design of an enhanced authentication protocol and its verification using AVISPA. 2016 3rd International Conference on Recent Advances in Information Technology (RAIT). IEEE, 2016. Pp. 404-409. DOI:10.1109/RAIT.2016.7507899
5. Cheval V., Cortier V., Turuani M. A little more conversation, a little less action, a lot more satisfaction: Global states in ProVerif. 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018. Pp. 344-358.
6. Chothia T., Smyth B., Staite C. Automatically checking commitment protocols in proverif without false attacks. International Conference on Principles of Security and Trust // Springer, Berlin, Heidelberg, 2015. Pp. 137-155. DOI:10.1007/978-3-662-46666-7-8
7. Blanchet B. Composition theorems for cryptoverif and application to TLS 1.3. 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018. Pp. 16-30. DOI:10.1109/CSF.2018.00009
8. Eswaraiah G., Vishwanathan R., Nedza D. Automated proofs of signatures using bilinear pairings. 2018 16th Annual Conference on Privacy, Security and Trust (PST). IEEE, 2018. Pp. 1-10. DOI:10.1109/PST.2018.8514201
9. Cremers C. Symbolic security analysis using the tamarin prover. 2017 Formal Methods in Computer Aided Design (FMCAD). IEEE, 2017. Pp. 5-5.
10. De Ruiter J., Poll E. Protocol State Fuzzing of {TLS} Implementations. 24th {USENIX} Security Symposium ({USENIX} Security 15). 2015. Pp. 193-206.
11. Cohn-Gordon K. et al. A formal security analysis of the signal messaging protocol. 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017. Pp. 451-466.
12. Beringer L. et al. Verified Correctness and Security of OpenSSL {HMAC}. 24th {USENIX} Security Symposium ({USENIX} Security 15). 2015. Pp. 207-221.
13. Kobeissi N., Bhargavan K., Blanchet B. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017. Pp. 435-450. DOI:10.1109/EuroSP.2017.38
14. Dowling B. et al. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015. Pp. 1197-1210.
15. Garcia R., Modesti P. An IDE for the design, verification and implementation of security protocols. 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2017. Pp. 157-163. DOI:10.1109/ISSREW.2017.69
16. Modesti P. AnBx: Automatic generation and verification of security protocols implementations. International Symposium on Foundations and Practice of Security. Springer, Cham, 2015. Pp. 156-173. DOI:10.1007/978-3-319-30303-1-10
17. Baskar A., Ramanujam R., Suresh S. P. Dolev-Yao theory with associative blindpair operators. International Conference on Implementation and Application of Automata. Springer, Cham, 2019. Pp. 58-69.
18. Szymoniak S., Siedlecka-Lamch O., Kurkowski M. SAT-based verification of NSPK protocol including delays in the network // 2017 IEEE 14th International Scientific Conference on Informatics. IEEE, 2017. Pp. 388-393.
19. Babenko L., Pisarev I. Translation of Cryptographic Protocols Description from Alice-Bob Format to CAS+ Specification Language // International Conference on Intelligent Information Technologies for Industry. Springer, Cham, 2019. Pp. 309-318.
20. Chandre P. R., Mahalle P. N., Shinde G. R. Machine learning based novel approach for intrusion detection and prevention system: A tool based verification // 2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN). IEEE, 2018. Pp. 135-140.
21. Bloemen V., van de Pol J. Multi-core SCC-based LTL model checking //Haifa Verification Conference. Springer, Cham, 2016. Pp. 18-33.