УДК 621.382.26

О ПРОБЛЕМАХ ПРОТОКОЛОВ ВЗАИМОДЕЙСТВИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

К. Р. Большаков,
ассистент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье рассматриваются проблемы разработки протоколов уровня приложений и причины возникновения этих проблем. Предлагается подход к механизму конструирования таких протоколов, нацеленный на преодоление указанных проблем.

This paper outlines current problems of development of application-level protocols in distributed systems. A review of issues and their corresponding causes is given. An approach to protocol construction is outlined that is aimed at overcoming these issues.

В условиях разнообразия известных и новых протоколов взаимодействия распределенных систем методы атаки на такие системы часто основываются не только на несовершенстве моделей безопасности самих систем, но и на недостатках протоколов взаимодействия, влекущих за собой зависимость от человеческого фактора. К таким недостаткам можно отнести различную интерпретацию данных, поступающих из сети, на разных уровнях стеков протоколов, отсутствие строгой типизации данных в сообщениях протоколов, синтаксические структуры без явно указанных длин. Использование конструкций сообщений, удобных для прочтения человеком (как, например, в протоколах, ведущих свое происхождение от HTTP [1]), приводит к неудобству их интерпретации на машинном уровне, что, в свою очередь, провоцирует ошибки программирования, влекущие появление уязвимостей. Реализации многих стеков протоколов имеют дефекты, не свойственные выбранному методу реализации и имеющие причиной ошибки системного программиста, сколько заложенным в саму структуру протоколов и системных библиотек недостатками и нечеткими спецификациями. Снижение производительности протоколов часто происходит за счет неоптимального метода кодирования данных.

Разработка протоколов и стеков протоколов уровня приложений обычно включает в себя этапы спецификации, реализации, тестирования и эксплуатации построенных на основе этих протоколов распределенных систем. Рассмотрим каждый из этапов, отметив, что основное внимание будет сконцентрировано на этапах спецификации и реализации.

На сегодняшний день активно развивающиеся протоколы часто разрабатываются не в закрытом режиме (одной стандартизационной организацией), а комитетами, куда входят поставщики программного обеспечения, либо сообществами, в которых выполняется работа по анализу и изменению черновых вариантов протоколов. Примером таких организаций могут служить рабочие группы IETF (Internet Engineering Task Force), разрабатывающие значительное число современных протоколов уровня приложений (но не ограничиваясь им). Как правило, и независимые разработчики (которые редко являются таковыми – даже университетские исследования финансируются корпорациями), и производители программного и аппаратного обеспечения, а также зависящие от них производители услуг в области телекоммуникаций весьма заинтересованы в скорейшем появлении на рынке продуктов, соответствующих новым стандартам. Не вдаваясь в рассмотрение мотивации, отметим, что одним из инструментов достижения цели широчайшего распространения новых протоколов является использование каких-либо старых протоколов в качестве базовых если не по семантике, то по синтаксису.

Примером такого заимствования синтаксиса и, отчасти, семантики некоторых операций может служить протокол SIP (Session Initiation Protocol) [2],

многое перенявший у протокола HTTP (Hypertext Transfer Protocol). Даже спецификация SIP содержит значительное число нормативных ссылок на спецификацию HTTP. Использование известных протоколов в качестве базы позволяет надеяться, во-первых, на ускоренную адаптацию разработчиков ПО, а во-вторых, на возможность повторного использования программных компонент, применяемых для работы с базовыми протоколами. Дополнительным критерием «простоты» часто служит «читаемость» двоичного представления нового протокола: возможность человека без использования специальных средств визуально воспринимать сообщения протокола. Заметим, что такой же подход свойственен и недавно принятому стандарту в области представления данных – XML [3].

Такой подход страдает несколькими недостатками. Во-первых, не принимаются во внимание многие недостатки базовых протоколов. Например, и у XML, и у HTTP наблюдается значительная избыточность представления как в структуре, так и в содержании информационных полей. В IETF даже была разработана группа стандартов SIGCOMP – стандарты по сжатию сообщений протоколов с неэкономным представлением сообщений. Во-вторых, отсутствие явной типизации полей, их переменная длина без ее явного указания и ориентированность на «читаемость» приводят к необходимости неоднократных преобразований содержимого информационных полей протоколов при их кодировании и декодировании (этот процесс называется «экранированием» – escaping). В-третьих, протоколы часто наследуют и нестройную структуру. В качестве примера можно привести принцип группирования заголовков в протоколе SIP: все заголовки, внутренняя структура которых представима в виде списка, могут группироваться через запятую внутри одного заголовка с тем же именем, и лишь четыре заголовка такого вида группироваться не могут. В этом примере видно, что протокол не только содержит правило, обоснование для которого существует лишь для меньшей части заголовков, но и содержит исключения из этого правила. В-четвертых, спецификация протокола декларирует цель уменьшения размера сообщений и, в связи с этим, вводит дополнительные однобуквенные обозначения для имен заголовков протокола, что приводит к вопросу о необходимости других представлений имен заголовков.

Кроме приведенных недостатков, синтаксическая структура с полями переменной длины без явного ее указания и вложенные синтаксические конструкции провоцируют ошибки при реализации протоколов, что было убедительно показано в работе [4]. Типичными ошибками для реализаций являются не только переполнения буферов и стеков [5], когда синтаксическая структура по вложенности, длине или повторяемости элементов превышает предусмотренные разработчиком пределы, но и переполнение памяти, что имеет место при использовании устройств с ограниченным ее объемом.

На этапе реализации проявляются как ошибки, упомянутые выше, так и нечеткость спецификации самих протоколов. Спецификация того же протокола SIP требует от читателя значительных усилий при своей интерпретации; в помощь тем, кто занимается реализацией этого протокола, даже организована рассылка, в которой создатели спецификации протокола помогают разработчикам программного обеспечения в интерпретации спецификации.

При реализации также проявляются проблемы системных и прикладных библиотек, например, работы со строками или выделением памяти. Некоторые стандартные библиотеки содержат функции, неосторожное применение которых может привести к перезаписыванию сторонних областей памяти.

Тестирование протоколов перекликается с реализацией. Авторами исследования [4] были разработаны наборы тестов для некоторых распространенных протоколов, которые показали, что даже программное обеспечение, произведенное известными компаниями, страдает от проблем реализации и подвержено как атакам отказа в сервисе, так и захвату устройств и узлов, на которых выполняется это программное обеспечение. Некоторые из реализаций криптографически защищенных протоколов содержали уязвимости, связанные с недостаточной криптостойкостью использованных генераторов псевдослучайных чисел [6].

Эксплуатация протоколов предусматривает наличие процесса как введения поправок в уже существующие спецификации, так и добавления расширений. К сожалению, такой процесс часто начинается только после выпуска нескольких расширений, которые неортогональны друг другу, что сильно затрудняет их реализацию, и только после этого создаются рекомендации по разработке расширений [7].

Таким образом, требуется введение ряда мер, позволяющих разрабатывать безопасные приложения на основе существующих стандартов и обеспечивающих отсутствие указанных недостатков в спецификациях будущих протоколов, что важно для создания защищенных телекоммуникационных систем.

Для решения первой задачи существуют следующие методы защиты:

стандартизация и сертификация библиотек криптографических примитивов, что позволяет гарантировать свойства реализаций криптографических примитивов, используемых в реализациях стеков протоколов;

отказ от использования распространенных небезопасных библиотечных вызовов. В этом случае разработчик использует безопасные методы, что уменьшает число ошибок, происходящих из-за небрежного программирования;

использование специальных средств защиты программного кода, например, компиляторов, проверяющих фрейм стека при возврате из вызова [8].

В некоторых случаях на этапе дизайна следует прибегать к формальной верификации проектируе-

мых реализаций с помощью, например, средств автоматической проверки моделей [9].

Вторая задача требует продуманности решения не только с точки зрения решаемых протоколом актуальных вопросов, но и с точки зрения последующих расширений. Одним из проработанных в смысле архитектуры подходов является расширяемое семейство протоколов на основе Абстрактной Синтаксической Нотации (АСН.1 [10]), адаптированных в Российской Федерации как ГОСТ Р ИСО/МЭК 8824-(1,2,3)-2001. Это семейство основано на формальных правилах преобразования в битовый поток последовательностей, описанных с помощью формальных синтаксических правил. Протокол не позволяет вводить новые типы данных, но использует тот же подход, что и многие языки программирования: новые типы данных конструируются из старых. Это позволяет однократные проектирование и реализацию модуля генерации и разбора сообщений протокола. Протокол использует строго типизированные данные и позволяет легко конструировать на своей основе новые протоколы, в том числе, с возможностью расширения. Двоичное представление сообщений протокола содержит либо указание на тип данных, имеющий фиксированную длину, либо длину элемента, варьирующегося в размере. Упаковка данных из внутреннего представления в двоичное производится достаточно экономным образом. Повсеместное введение АСН.1 как базы для разрабатываемых протоколов уровня приложений обеспечит как облегчение разработки будущих расширений, так и повышение качества реализаций стеков протоколов за счет унификации модулей работы с данными.

Литература

1. **Fielding R., Gettys J., Mogul J., Frystyk H.** and oth. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999 (http://ietf.org/rfc/rfc2616.txt).
2. **Rosenberg J., Schulzrinne H., Camarillo G.** and oth. SIP: Session Initiation Protocol. RFC 3261, June 2002 (http://ietf.org/rfc/rfc3261.txt).
3. **Yergeau F., Cowan J., Bray T.** and oth. XML 1.1. W3C Recommendation, February 2004. (http://www.w3.org/TR/xml11).
4. **Röning J., Laakso M., Takanen A., Kaksonen R.** PROTOS – systematic approach to eliminate software vulnerabilities. Invited presentation at Microsoft Research, Seattle, USA. May 6, 2002. (http://www.ee.oulu.fi/research/ouspg/protos/sota/MSR2002-protos/index.html).
5. **One Aleph.** Smashing the stack for fun and profit. Phrack Magazine. – 49. – Vol. 7. – issue 49. – file 14 (http://www.phrack.org/phrack/49/P49-14).
6. **Goldberg I., Wagner D.** Randomness and the netscape browser. Dr. Dobb's Journal, January 1996. (http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html).
7. **Mankin A., Bradner S., Mahy R.** and oth. Change Process for the Session Initiation Protocol (SIP). RFC 3427, December 2002 (http://ietf.org/rfc/rfc3427.txt).
8. **Cowan C., Pu C., Maier D.** and oth. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. // Proceedings of 7th USENIX Security Conference. – 1998. – Texas, San Atonio. – P. 63–78.
9. **Holzmann G. J.** The Model Checker SPIN // IEEE Trans. on Software Engineering. – 1997. – 23(5) – P. 279–295.
10. **ГОСТ** Р ИСО/МЭК 8824-(1,2,3)-2001. Абстрактная синтаксическая нотация версии один (АСН.1). Спецификация основной нотации. Спецификация информационного объекта. Спецификация ограничения.