Технологии строгой аутентификации

Сергей ПАНАСЕНКО,
директор по научной работе, компания «Актив»

Управление доступом пользователей к ресурсам информационных систем является одной из основных мер защиты как информации, хранящейся/обрабатываемой в системах, так и систем в целом. Конкретный пользователь системы авторизуется на доступ к определенному ресурсу в соответствии с заранее установленными правилами разграничения доступа по результатам идентификации и аутентификации пользователя. Можно утверждать, что качественное управление доступом возможно только в случае применения таких процедур аутентификации пользователей, которые не позволяют кому-либо выдать себя за легального пользователя системы, а также не позволяют легальному пользователю системы выдать себя за другого пользователя – например, обладающего бóльшими полномочиями. Обсудим в данной статье возможности и технологии строгой аутентификации пользователей, которая позволяет противодействовать подобным угрозам.

Возможные угрозы при использовании уязвимых методов аутентификации

В качестве примеров потенциально уязвимых методов аутентификации рассмотрим следующие:

- парольную аутентификацию;
- аутентификацию с использованием кода, хранящегося в носителе, не обеспечивающем защиту хранящихся в нем данных.

При использовании вышеперечисленных методов могут возникать следующие угрозы (рис. 1):

- перехват злоумышленником данных аутентификации (пароля или кода) при их вводе или передаче по сети;
- онлайн-подбор данных аутентификации (возможно как для пароля, так и для кода, но особенно актуально при использовании паролей: пароли могут быть слабыми, являться производными от логинов/имен пользователей и т. п.);
- офлайн-подбор данных аутентификации, например, путем анализа файла с хешированными паролями пользователей (при получении злоумышленником доступа к данному файлу);
- клонирование носителя, т. е. создание его копии/эмулятора, содержащей требуемый код (при получении злоумышленником доступа к носителю).

Реализация злоумышленником одной из данных угроз позволит ему аутентифицироваться под именем легального пользователя системы и в результате получить доступ ко всем ресурсам системы, к которым разрешен доступ данному пользователю.

Усиленная аутентификация – «противоядие» от подобных угроз

Усиленные методы аутентификации позволяют противодействовать описанным выше угрозам. Частным случаем таких методов является строгая аутентификация пользователей – аутентификация на основе криптографических протоколов. Основными свойствами строгой аутентификации, предопределяющими обеспечиваемый с ее помощью высокий уровень защиты, являются:

- в основе строгой аутентификации лежат криптографические механизмы; может использоваться аутентификация на основе стандартизованных криптографических протоколов с доказуемой стойкостью;

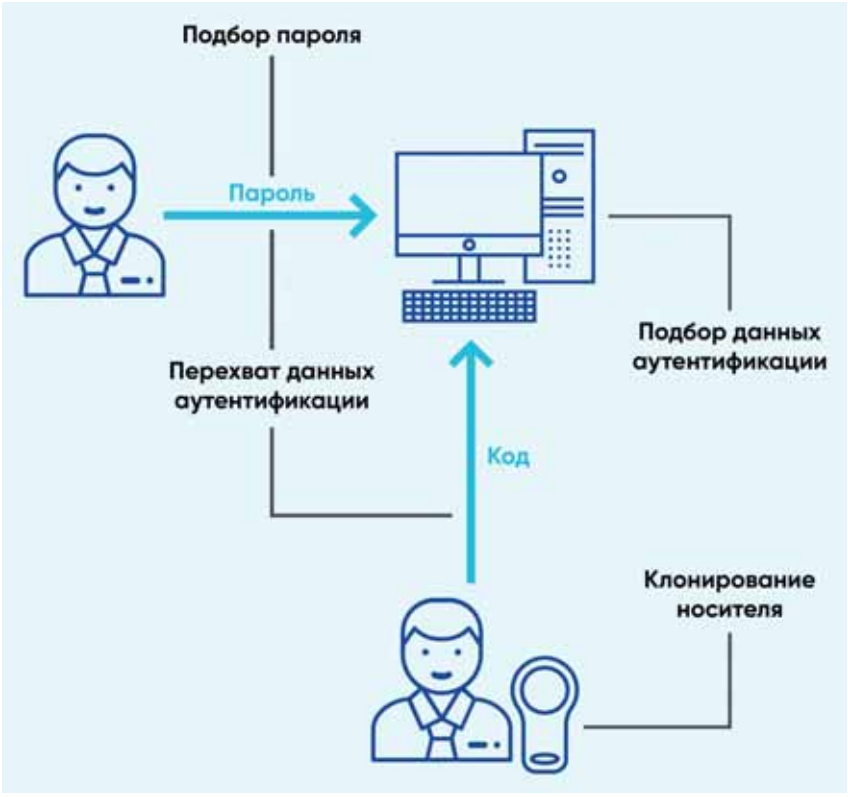

Рис. 1. *Угрозы при использовании аутентификации на основе пароля или клонируемого носителя*

- криптографические ключи, с помощью которых производится строгая аутентификация, могут храниться в защищенном от клонирования носителе (могут даже быть не извлекаемыми из носителя – такой ключ генерируется непосредственно носителем и никогда не покидает его), снабженном криптографическими функциями, что позволяет носителю не просто передавать какой-либо код в систему, а быть активным участником выполняемого протокола аутентификации;
- возможно выполнение взаимной аутентификации (например, между носителем и удаленным сервером), в результате которой вырабатывается общий криптографический ключ – такой ключ впоследствии может быть использован для защиты канала связи между прошедшими взаимную аутентификацию сущностями.

Носителем с описанными выше свойствами может быть, например, смарт-карта или криптографический токен. Рассмотрим далее примеры протоколов строгой аутентификации, которые могут выполняться данными носителями.

Пример строгой аутентификации с помощью смарт-карт

Микропроцессорные смарт-карты могут быть оснащены достаточно широким набором криптографических функций; как было сказано выше, такие карты могут являться носителем аутентификационных данных, активно участвующим в процессе аутентификации.

В качестве примера действий по аутентификации пользователей, которые могут выполнять смарт-карты, рассмотрим режим MUTUAL AUTHENTICATE команд EXTERNAL AUTHENTICATE и GENERAL AUTHENTICATE, которые являются стандартными командами для смарт-карт; они описаны, в частности, в стандарте ГОСТ Р ИСО/МЭК 7816-4-2013.

Данный режим позволяет выполнить строгую взаимную аутентификацию смарт-карты (т. е. пользователя-владельца смарт-карты; до выполнения описанной далее последовательности действий по аутентификации пользователь должен сам аутентифицироваться смарт-картой – например, ввести корректный PIN-код) и второго участника аутентификации (например, терминала) с выработкой общего ключа. Аутентификация в этом случае может выполняться следующим образом (рис. 2):

Шаг 1. Поскольку смарт-карта не может выступать инициатором выполнения протокола аутентификации (смарт-карта является ведомым устройством, выполняющим команды терминала и отвечающим на них), выполнение протокола начинает терминал (например, после получения соответствующей команды от пользователя), для чего он посылает на смарт-карту команду GET CHALLENGE, запрашивающую у смарт-карты задачу на выполнение аутентификации терминала.

Шаг 2. В ответ на эту команду смарт-карта выдает терминалу задачу, которая обычно включает в себя некое случайное число (обозначим его R_C).

Шаг 3. Терминал решает данную задачу и отправляет на смарт-карту команду EXTERNAL AUTHENTICATE (с указанием режима MUTUAL AUTHENTICATE), в данных которой передает смарт-карте решение ее задачи и свою задачу. Решением задачи может быть, например, зашифрование случайного числа R_C алгоритмом симметричного шифрования на некоем заранее известном смарт-карте ключе K (это доказывает карте знание терминалом данного

секретного ключа, но ключ K должен быть загружен на карту заранее в доверенной среде, например, в процессе регистрации пользователя в системе), а задачей терминала может быть еще одно случайное число R_T. Т. е. передаваемые терминалом смарт-карте данные команды EXTERNAL AUTHENTICATE можно представить в виде следующей криптограммы:
$C_T = E_K(R_C, R_T)$,
где E – используемый алгоритм шифрования.

Шаг 4. Получив такую команду, смарт-карта проверяет криптограмму – расшифровывает ее и убеждается, что она содержит отправленное ранее терминалу случайное число R_C. После этого смарт-карта генерирует общий симметричный ключ, создает ответную криптограмму, содержащую случайное число R_T, например, $C_C = E_K(R_T)$ и отправляет ее на терминал. Общий симметричный ключ K_S может генерироваться на основе известного обоим сторонам ключа K и одного или обоих случайных чисел R_C и R_T с помощью какой-либо функции генерации производного ключа.

Шаг 5. После получения ответа на команду EXTERNAL AUTHENTICATE, содержащего криптограмму C_C, терминал расшифровывает данную криптограмму и убеждается, что она содержит отправленное им ранее случайное число R_T, что доказывает терминалу знание смарт-картой секретного ключа K. После этого терминал идентичным смарт-карте образом генерирует общий симметричный ключ K_S.

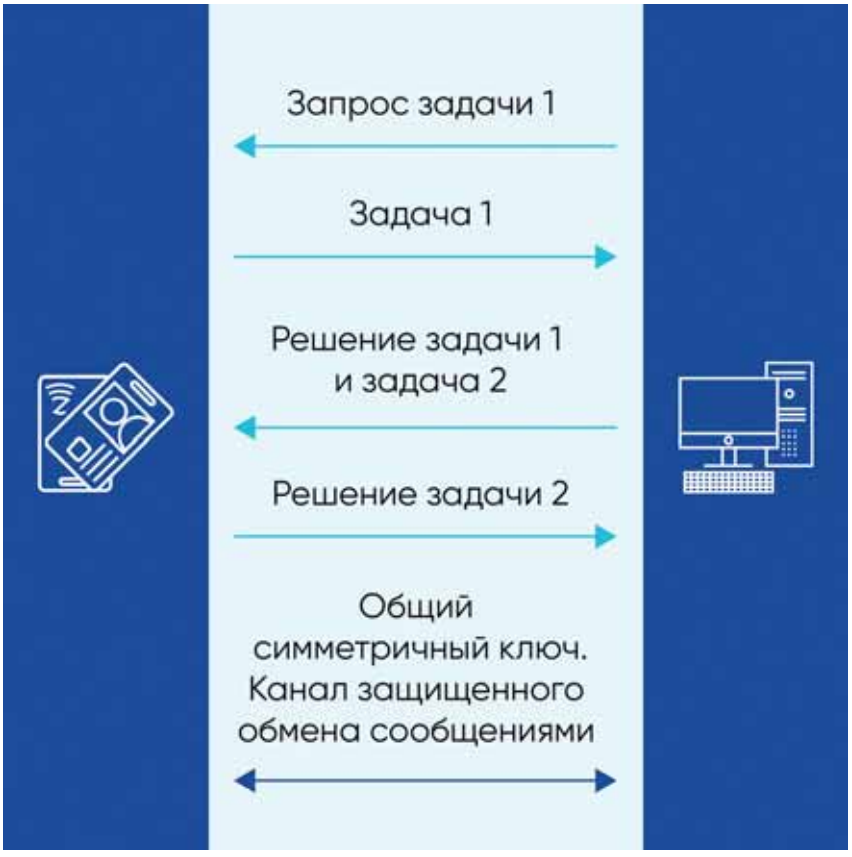

Рис. 2. *Пример строгой взаимной аутентификации с помощью смарт-карты*

После первых трех шагов терминал является аутентифицированным смарт-картой, а после пяти смарт-карта также аутентифицирована терминалом (оба устройства доказали друг другу знание определенного секретного ключа K) плюс выработан общий симметричный ключ для дальнейшего защищенного обмена сообщениями в рамках текущей сессии взаимодействия смарт-карты и терминала.

Конкретные действия, которые должны выполнять данные устройства в рамках режима MUTUAL AUTHENTICATE, не стандартизованы; выше приведен лишь упрощенный пример реализации данного режима.

Для аутентификации и выработки общего ключа в данном режиме могут использоваться и асимметричные криптоалгоритмы: в этом случае на карте может находиться неизвлекаемый приватный ключ, а терминал должен заранее считать соответствующий публичный ключ с карты и загрузить на карту собственный публичный ключ. Тогда общий симметричный ключ может быть вычислен каждой сущностью на основе собственного приватного ключа и публичного ключа другой стороны аутентификации с помощью протокола ECDH (Elliptic Curve Diffie-Hellman – протокол Диффи-Хеллмана на эллиптических кривых) или аналогичного.

Пример строгой аутентификации с помощью криптографических токенов

Еще одним примером протокола строгой взаимной аутентификации с выработкой общего симметричного ключа является протокол SESPAKE (Security Evaluated Standardized Password-Authenticated Key Exchange) – протокол выработки общего ключа с аутентификацией на основе пароля, который может быть реализован в криптографическом токене.

Данный протокол разработан коллективом криптографов отечественной компании «Крипто-Про» и стандартизован в рекомендациях по стандартизации Р 50.1.115-2016. Упрощенная последовательность выполнения протокола SESPAKE сторонами A (криптографический токен) и B (сервер) выглядит следующим образом (об упрощениях см. далее):

Шаг 1. Сторона A отправляет стороне B запрос на проведение аутентификации, содержащий идентификатор стороны A ID_A. Отправка и дальнейшее использование в рамках протокола идентификатора ID_A являются опциональными (за исключением ряда случаев, где его использование обязательно); если данный идентификатор не используется, то он принимается равным пустой строке (строке нулевого размера).

Шаг 2. Сторона B отправляет стороне A следующие значения:
- индекс стартовой точки используемой эллиптической кривой ind;
- соль (используемая для рандомизации ряда значений случайная величина) salt;
- идентификатор используемой эллиптической кривой ID_{alg};
- идентификатор стороны B ID_B.

Протокол предполагает, что в рамках его выполнения может использоваться несколько различных эллиптических кривых (конкретные эллиптические кривые для обязательного использования в рамках протокола не установлены). Используемая в конкретном случае выполнения данного протокола эллиптическая кривая определяется идентификатором ID_{alg}. Все дальнейшие вычисления в рамках протокола обе стороны проводят в группе точек данной эллиптической кривой.

На используемой эллиптической кривой должны быть определены N точек (N является параметром протокола) $\{Q_1, Q_2, \ldots, Q_N\}$, относящихся к циклической подгруппе порядка q группы точек эллиптической кривой.

Точка Q_{ind}, относительно которой осуществляются дальнейшие вычисления в рамках протокола SESPAKE, определяется ее индексом ind, ind $\in$ $\{1, \ldots, N\}$.

Значение salt выбирается из диапазона $\{1, \ldots, 2^{128}-1\}$.

Идентификатор ID_B является опциональным аналогично ID_A.

Параметры ind, salt, ID_A и ID_B могут быть согласованы сторонами A и B заранее. В этом случае передача данных значений в рамках выполнения шагов 1 и 2 не требуется, но эти значения должны использоваться в дальнейших вычислениях в рамках протокола. Параметр ID_{alg} также может быть зафиксирован или согласован заранее.

Шаг 3. Сторона A вычисляет точку эллиптической кривой Q^A_{PW}:

$Q^A_{PW} = int(F(PW, salt, 2000)) * Q_{ind}$,

где:
- PW – пароль, на основе которого выполняется аутентификация;
- int – функция преобразования байтовой строки в целое число;
- F – функция вычисления ключа на основе пароля.В качестве функции F(PW, salt, n) используется определенная в рекомендациях по стандартизации Р 50.1.111-2016 функции PBKDF2(PW, salt, n, len), где значение параметра len равно:
  - 256, если $2^{254} < q < 2^{256}$;
  - 512, если $2^{508} < q < 2^{512}$.

Шаг 4. Сторона A случайным образом выбирает значение α из диапазона $\{1, \ldots, q-1\}$ и вычисляет точку u_1 на используемой эллиптической кривой:

$u_1 = \alpha * P - Q^A_{PW}$,

где P – порождающий элемент циклической подгруппы порядка q группы точек используемой эллиптической кривой.

Шаг 5. Сторона A отправляет вычисленную точку u_1 стороне B.

Шаг 6. Сторона B проверяет, принадлежит ли полученная от стороны A точка u_1 используемой эллиптической кривой. Если это не так, то аутентификация считается неуспешной и работа протокола завершается.

Шаг 7. Сторона B вычисляет точку Q_B на используемой эллиптической кривой:

$Q_B = u_1 + Q_{PW}$,

где точка Q_{PW} вычисляется на основе пароля следующим образом (она может быть вычислена заранее):

$Q_{PW} = int(F(PW, salt, 2000)) * Q_{ind}$.

Шаг 8. Сторона B случайным образом выбирает значение β из диапазона $\{1, \ldots, q-1\}$ и вычисляет значение K_B:

$K_B = Hash(Bytes(((m / q) * \beta \bmod q) * Q_B))$,

где:
- m – порядок группы точек используемой эллиптической кривой;
- Hash – стандартная функция хеширования, определенная в ГОСТ Р 34.11-2012, с 256-битным выходным значением;
- Bytes – функция, преобразующая точку эллиптической кривой в байтовое представление.

Шаг 9. Сторона B вычисляет точку u_2 на используемой эллиптической кривой: $u_2 = \beta * P + Q_{PW}$,

Шаг 10. Сторона B отправляет вычисленную точку u_2 стороне A.

Шаг 11. Сторона A проверяет, принадлежит ли полученная от стороны B точка u_2 используемой эллиптической кривой. Если это не так, то аутентификация считается неуспешной и работа протокола завершается.

Шаг 12. Сторона A вычисляет точку Q_A на используемой эллиптической кривой: $Q_A = u_2 - Q^A_{PW}$.

Шаг 13. Сторона A вычисляет значение K_A: $K_A = Hash(Bytes(((m / q) * \alpha \bmod q) * Q_A))$.

Шаг 14. Сторона A вычисляет значение MAC_A: $MAC_A = HMAC(K_A, 0x01 \| ID_A \| ind \| salt \| u_1 \| u_2 \| DATA_A)$,

где:

- HMAC(K, M) – функция вычисления HMAC (Hash-based Message Authentication Code – код аутентификации сообщения на основе хеширования) для сообщения M на ключе K, определенная в рекомендациях по стандартизации Р 50.1.113-2016, с 256-битным выходным значением;
- || – операция конкатенации;
- $DATA_A$ – опциональные аутентифицируемые данные, передаваемые стороной A стороне B в рамках выполнения протокола; если передача аутентифицируемых данных не требуется, то значением $DATA_A$ считается пустая строка.

Шаг 15. Сторона A отправляет стороне B значение $DATA_A \| MAC_A$.

Шаг 16. Сторона B проверяет равенство значения MAC_A и следующего значения: $HMAC(K_B, 0x01 \| ID_A \| ind \| salt \| u_1 \| u_2 \| DATA_A)$.
Если данные значения не равны, то аутентификация считается неуспешной и работа протокола завершается.

Шаг 17. Сторона B вычисляет значение MAC_B: $MAC_B = HMAC(K_B, 0x02 \| ID_B \| ind \| salt \| u_1 \| u_2 \| DATA_A \| DATA_B)$, где $DATA_B$ – опциональные аутентифицируемые данные, передаваемые стороной B стороне A в рамках выполнения протокола; если передача аутентифицируемых данных не требуется, то значением $DATA_B$ считается пустая строка.

Шаг 18. Сторона B отправляет стороне A значение $DATA_B \| MAC_B$.

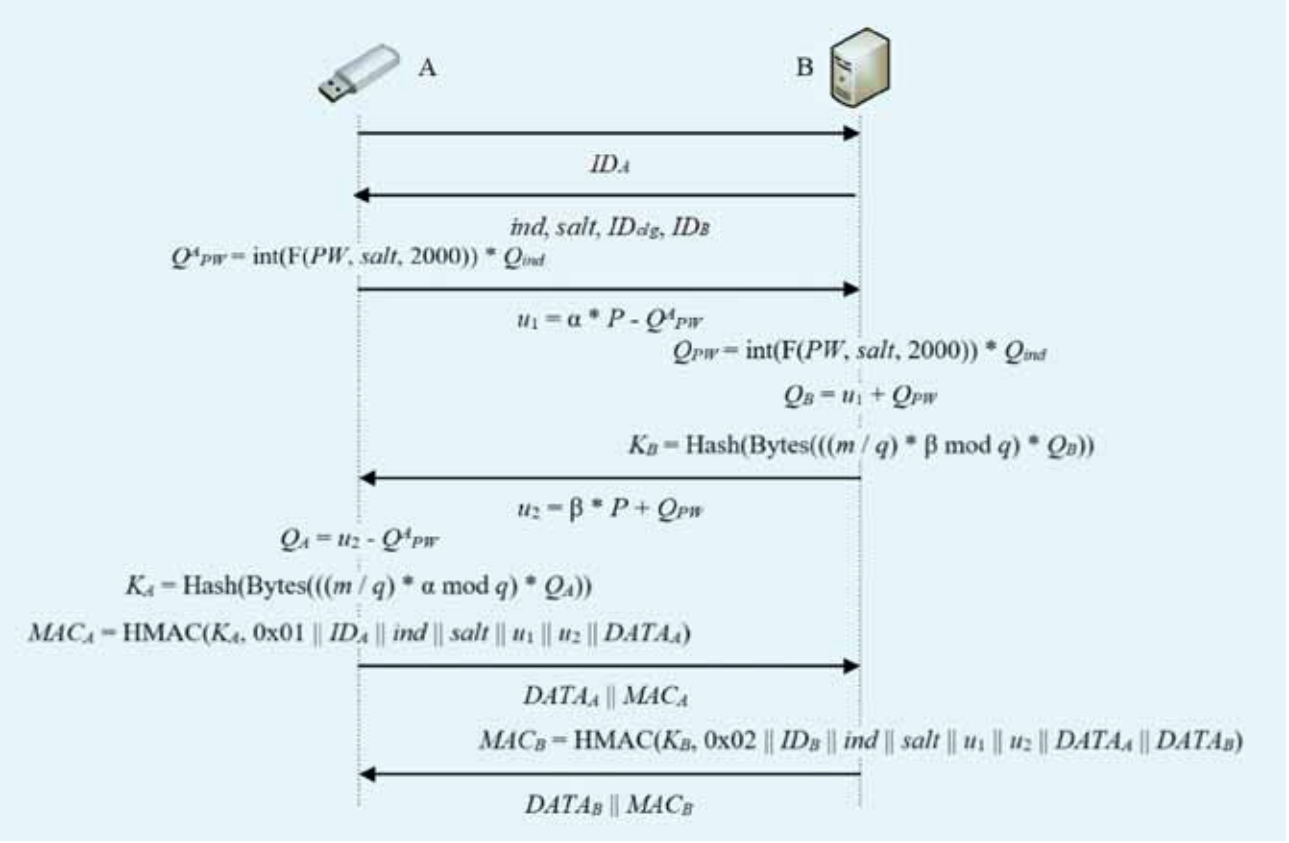

Рис. 3. *Упрощенная схема протокола SESPAKE*

Шаг 19. Сторона A проверяет равенство значения MAC_B и следующего значения: $HMAC(K_A, 0x02 \| ID_B \| ind \| salt \| u_1 \| u_2 \| DATA_A \| DATA_B)$. Если данные значения не равны, то аутентификация считается неуспешной и работа протокола завершается. В противном случае аутентификация считается успешно выполненной; работа протокола также завершается.

Схема основных действий, выполняемых сторонами в рамках протокола, приведена на рис. 3.

Одно из основных упрощений приведенной выше последовательности обмена данными – отсутствие в ней счетчиков, которые ведут стороны при взаимодействии в рамках протокола SESPAKE. Целью счетчиков является противодействие атакам путем подбора и контроль непревышения максимально допустимого количества использований паролей. Кроме того, в протоколе предусмотрены дополнительные проверки на промежуточных этапах, которые опущены в приведенном выше описании.

В случае успешного выполнения протокола SESPAKE его результатами являются:

- успешно проведенная взаимная аутентификация сторон;
- вычисленный общий симметричный ключ $K = K_A = K_B$.

Протокол выглядит сложным, но его реализация в криптографическом токене позволяет скрыть все сложности протокола от пользователя. При этом протокол имеет массу достоинств:

- возможность генерации высокоэнтропийного общего ключа на основе пароля;
- внедренная в протокол защита от атак путем подбора и ряда других;
- использование в рамках протокола стандартизованных криптографических алгоритмов;
- доказанная криптографическая стойкость протокола.

Заключение

Разделение описанных выше протоколов на реализуемые смарт-картой и криптографическим токеном является условным, поскольку они могут быть реализованы и в том, и в другом типе аутентифицирующих носителей (данные протоколы являются лишь примерами, существуют и другие протоколы, с помощью которых можно выполнять строгую аутентификацию).

Подобные интеллектуальные аутентифицирующие носители, снабженные реализациями криптографических алгоритмов и протоколов строгой взаимной аутентификации, позволяют качественно защитить процедуры аутентификации даже от злоумышленников достаточно высокого уровня, что дает возможность обеспечить более высокий уровень защищенности информационных систем, особенно в части разграничения доступа к их ресурсам. ■

Аутентификация без трудностей

Рутокен MFA обеспечивает двухфакторную аутентификацию при входе в локальные учетные записи Ред ОС. Компании «Актив» и «Ред Софт» подтвердили совместимость пользовательских устройств новой линейки Рутокен MFA и отечественной операционной системы Ред ОС для строгой аутентификации при входе в локальные учетные записи. Работа устройств линейки Рутокен MFA основана на стеке технологий FIDO2 (CTAP2). Использование пакетов из репозитория Ред ОС позволяет аутентифицировать пользователя при входе в учетную запись по данному стандарту. После ввода логина и пароля пользователь должен ввести PIN-код от устройства (фактор знания) и подтвердить свое присутствие нажатием на сенсорную кнопку (фактор владения). Таким образом, заказчики получают возможность использовать один аутентификатор Рутокен MFA как для задач аутентификации в веб-сервисы, так и для входа в операционную систему. Ред ОС может применяться в организациях с высокими требованиями к информационной безопасности и сохранности персональных данных (до первого класса включительно). Безопасность работы Ред ОС обеспечивается средствами защиты информации от доверенных партнеров Ред Софт, встроенными инструментами мониторинга и реагирования на инциденты, присутствует сканер уязвимостей и средства безопасной разработки. Сценарии совместного использования продуктов Ред ОС и Рутокен MFA опубликованы на портале документации Рутокен; в базе знаний Ред ОС. Совместимость решений компаний «Актив» и Ред Софт подтверждены сертификатом. Ред ОС – российская операционная система семейства Linux для серверов и рабочих станций, предоставляющая универсальную среду для использования прикладного ПО. Ред ОС разрабатывается в закрытом контуре Ред Софт, исходные коды и пакеты находятся в собственном репозитории Ред ОС, расположенном на территории РФ. Этот составной программный продукт построен на пакетной базе RPM-формата, соответствующей требованиям POSIX и LSB 4.1 (Linux Standard Base). По словам Рустама Рустамова, заместителя генерального директора Ред Софт, Ред ОС создается как безопасная среда для работы. Задача компании – соответствовать самым высоким стандартам и требованиям к информационной безопасности, так как среди заказчиков Ред Софт много организаций и компаний, которые заботятся о сохранности своих данных. Такие продукты, как устройства Рутокен MFA, помогают расширить стек решений по информационной безопасности, доступных для пользователей Ред ОС.