УДК 628.1.03

О.Кириловська, О.Чмут
Вінницький державний технічний університет

ІНФОРМАЦІЙНА НАДІЙНІСТЬ СИСТЕМ КЕРУВАННЯ

Найбільш універсальним є узагальнений прикладний програмний інтерфейс служби безпеки (Generic Security Service Application Program Interface- GSS-API) - набір специфікацій, схвалених співтовариством Internet, що поширює ідеологію відкритих систем на таку традиційно закриту область, як інформаційна безпека, що забезпечує аутентифікацію, цілісність, конфіденційність. Інтерфейс GSS-API, на відміну від більшості інших засобів безпеки, орієнтованих на локальне використання, має на меті захист комунікацій між компонентами програмних систем, побудованих в архітектурі клієнт-сервер. Він надає послуги із взаємної аутентифікації (перевірки істинності) партнерів, що спілкуються, і з контролю цілості і забезпечення конфіденційності повідомлень, що надсилаються.

При проектуванні розподілених систем керування виробничими комплексами одним із найскладніших завдань є забезпечення інформаційної безпеки обчислювальних мереж. На сьогодні існує ряд програмних і апаратних засобів захисту інформації, які на певному рівні надійності розв'язують це завдання. Одним із найбільш універсальних є узагальнений прикладний програмний інтерфейс служби безпеки (Generic Security Service Application Program Interface - GSS-API) - набір специфікацій, схвалених співтовариством Internet, що поширює ідеологію відкритих систем на таку традиційно закриту область, як інформаційна безпека, забезпечує аутентифікацію, цілість, конфіденційність.

Інтерфейс GSS-API, на відміну від більшості інших засобів безпеки, орієнтованих на локальне використання, має на меті захист комунікацій між компонентами програмних систем, побудованих в архітектурі клієнт-сервер. Він надає послуги із взаємної аутентифікації (перевірки істинності) партнерів, що спілкуються, і з контролю цілості і забезпечення конфіденційності повідомлень, що надсилаються. Користувачами інтерфейсу безпеки GSS-API є комунікаційні протоколи (звичайно прикладного рівня) або інші програмні системи, що самостійно виконують надсилання даних. GSS-API не залежить від конкретного мовного середовища і від механізму безпеки, що забезпечує реальний захист. Це дозволяє створювати додаткові локальні засоби захисту, що на рівні вихідного тексту мобільні щодо зміни механізму безпеки, чим реалізовується відкритість прикладних систем і відповідних засобів захисту.

Засоби захисту можуть бути істотно різними - від систем Kerberos, що базуються на симетричних методах шифрування, і до продуктів, що реалізують специфікації X.509, де явно обумовлюється використання відкритих ключів. На кожному комп'ютері, де передбачається використовувати інтерфейс безпеки GSS-API, треба встановити клієнтське програмне забезпечення відповідного механізму захисту. Додаток, що використовує GSS-API, локально викликає необхідні функції, одержуючи у відповідь так звані токени безпеки. Подібний токен може містити зашифроване посвідчення користувача, електронний підпис під повідомленням або ціле зашифроване повідомлення. Додатки обмінюються токенами безпеки, досягаючи цим аутентифікації, цілості і конфіденційності повідомлення. Оскільки комунікаційні аспекти винесені за межі узагальненого інтерфейсу безпеки GSS-API, він автоматично надається незалежно від мережних протоколів. Мережна мобільність повинна забезпечуватися інакше.

Базовими елементами GSS-API є поняття посвідчення, контексту безпеки, токена безпеки.

Посвідчення - це структура даних, що дозволяє його власникові доказувати партнерові спілкування або третій стороні, що він (власник) є саме тим, за кого себе видає. Тобто у GSS-API посвідчення є засобом аутентифікації. Природно, що служба

безпеки (наприклад, Kerberos) разом з операційною системою повинна забезпечити захист посвідчень від несанкціонованого використання. Процесам, що діють від імені користувачів, не дається права прямого доступу до посвідчень. Замість цього при необхідності процеси постачаються дескрипторами посвідчень. Дескриптори не містять секретної інформації і не потребують захисту. Нема рації красти дескриптори, оскільки їхня інтепретація для різних процесів - пред'явників буде різноманітною. Одному користувачеві можуть знадобитися посвідчення декількох видів для спілкування з різними партнерами.

Структура посвідчень залежить від механізму безпеки, що стоїть за узагальненим інтерфейсом. Особливою уявою повинні бути влаштовані так звані посвідчення, що делегуються, призначені для передання прав на виконання визначених дій від імені деякого користувача. Є й інші чинники, що впливають на структуру посвідчень. У процесі входу користувача до системи можуть формуватися посвідчення стандартного виду, що видаються за умовчанням (автоматично).

Контекст безпеки - це пара структур даних (по одній локально збереженій структурі для кожного партнера спілкування), в яких утримується роздільна інформація про процес спілкування, необхідна для захисту повідомлень, що надсилаються. Як і посвідчення, контексти безпеки зберігаються внутрішнім для служби безпеки способом - прикладні процеси постачаються лише дескрипторами контекстів. Контексти формуються на підставі локально виданих або делегованих посвідчень. Партнери спілкування можуть по черзі або одночасно використовувати декілька контекстів, якщо необхідно підтримувати інформаційні потоки різного рівня захищеності.

Інтерфейс GSS-API не залежить від використовуваного мережного протоколу. Тому формування контексту безпеки не пов'язане з встановленням з'єднання в мережному змісті. Мало того, для GSS-API байдуже, використовується протокол із встановленням з'єднання або без нього. Організація потоку повідомлень, а також виділення з вхідного потоку даних, що генеруються в рамках GSS-API, це обов'язок прикладних систем.

Токени безпеки - це елементи даних, що курсують між користувачами інтерфейсу GSS-API для підтримки працездатності цього інтерфейсу і захисту прикладної інформації. Токени підрозділяються на два класи. Контекстний клас призначений для встановлення контекстів безпеки і для виконання керуючих дій над ними. У рамках вже встановленого контексту для захисту повідомлень використовуються токени повідомлень.

Для додатка структура токенів безпеки є закритою. Токени генеруються і контролюються виняткова функціями GSS-API. Справа додатка - надіслати їх і передати відповідним функціям для опрацювання. Природно, служба безпеки виявить спроби додатка змінити токен.

На віддалених системах функціонує декілька різноманітних служб безпеки. Конкретна служба характеризується типом реалізованого механізму безпеки, що позначається структурою даних, так званого ідентифікатора об'єкта. На рівні узагальненого програмного інтерфейсу структура ідентифікаторів об'єктів не уточнюється. Якщо токени є структурою, закритою для додатків, то струтура імен, уживаних при формуванні контексту безпеки, закрита для функцій GSS-API. Імена розглядаються цими функціями як послідовності байтів, що інтепретується комунікаційними компонентами додатків. Передбачається наявність трьох типів імен - внутрішніх, друкарських і об'єктних. Як правило, аргументами функцій GSS-API служать внутрішні імена.

Кожна функція, що належить до узагальненого інтерфейсу безпеки GSS-API, повертає два коди відповіді - основний і додатковий. Набір основних кодів регламентується у рамках GSS-API, додаткові коди можуть бути специфічними для різноманітних служб безпеки. Робота додатка, що спирається на інтерфейс GSS-API,

повинна починатися з одержання посвідчення про те, що додаток має право працювати від імені визначеного суб'єкта. Створення контексту безпеки передує всім операціям з обміну повідомленнями між потенційними партнерами.

Деякі служби безпеки можуть на вибір надавати різноманітну якість захисту (Quality Of Protection - QOP). Вибір потрібної якості важливий для додатка з погляду розумної витрати ресурсів. Загальна схема взаємодії віддалених партнерів під захистом GSS-API досить проста, проте практична реалізація всіх необхідних перевірок потребує акуратності і точності.

Зараз питанням безпеки даних у розподілених комп'ютерних системах приділяється дуже велика увага. Розроблено багато засобів для забезпечення інформаційної безпеки, призначених для використання на різноманітних комп'ютерах із різними ОС. Як одним з напрямків можна виділити міжмережеві екрани (firewalls), покликані контролювати доступ до інформації з боку користувачів зовнішніх мереж. На прикладі пакета Solstice Firewall-1 можна розглянути декілька типових випадків використання таких систем, особливо щодо питань забезпечення безпеки Internet-підключень. Деякі унікальні особливості Solstice Firewall-1 дозволяють говорити про його лідерство у цьому класі додатків. Екран (firewall) - це засіб розмежування доступу клієнтів з однієї множини систем до інформації, що зберігається на серверах в іншій множині. Екран виконує свої функції, контролюючи всі інформаційні потоки між цими двома множинами інформаційних систем, працюючи як певна “інформаційна мембрана”. У цьому значенні екран можна уявляти як набір фільтрів, що аналізують плинну через них інформацію. Така система може реєструвати події, пов'язані з процесами розмежування доступу. Найважливішим прикладом потенційно ворожої зовнішньої мережі є Internet.

Роботи із захисту інформації починаються із моменту появи комп'ютерних систем. На етапі слабко розвинутих комунікацій це були окремі алгоритми, такі, як блоковий шифр DES (the Data Encryption Standard), що реалізовує симетричну схему із секретним ключем, затверджений урядом США як державний стандарт в 1977 році. RSA - криптосистема з відкритим ключем, що використовується як для шифрування, так і для аутентифікації (1977р.). Алгоритм криптографічного перетворення ДЕРЖСТАНДАРТ 28147-89 прийнятий у 1989 році і реалізовує блоковий алгоритм шифрування з імітозахистом.

З цього моменту алгоритми криптографічного захисту інформації з'являються регулярно. Блокові шифри IDEA (International Data Encryption Algorithm), RC-2, шифри RC-4, RC-5, що є удосконаленням шифру RC-2 на збільшення швидкості опрацювання.

Наступним етапом було об'єднання існуючих окремих алгоритмів в один. Так з'явився PGP, що використовує RSA для безпечного обміну ключами, IDEA - для шифрування повідомлень, RSA - для цифрового підпису і хеш-функця - MD5. Більш потужним засобом, що об'єднує в собі весь спектр технологій захисту інформації, є вітчизняний проект «Верба», що реалізує ДЕРЖСТАНДАРТ 28147-89, ДЕРЖСТАНДАРТ Р 34. 10-94 і ДЕРЖСТАНДАРТ Р 34. 11-94 відповідно у шифруванні, електронно-цифровому підписі і функції хешування.

З появою технологій типу клієнт-сервер виникла необхідність захисту on-line - сервісів, авторизації сеансів, записів у розподіліні бази даних та ін. Багато значних фірм, що не бажають втрачати своїх потенційних клієнтів, почали фінансувати проекти, що гарантують безпеку власним корпоративним ресурсам. Серед них - спільний проект фірм Microsoft Corporation і Visa International - технології PCT (Private Communication Technology), проект фірми Netscape - SSL (Secure Sockets Layer), спрямовані на безпеку комунікацій, вироблення безпечного протоколу S-HTTP (Secure Hypertext Transfer Protocol) фірми Enterprise Integration Technologies та ін.

В умовах, коли на ринку засобів захисту є широкий вибір продуктів захисту віддалених ресурсів, потенційному споживачеві важко визначити, який з них

надійніший. При виборі необхідно враховувати надійність фірми-виробника, рівень безпеки, інтеграцію продукту з власними інформаційними технологіями і ресурсами споживача, локалізацію технологій захисту до рівня робочих місць користувачів.

Таким чином, ми підійшли до необхідності створення єдиного інтегрованого інтерфейсу захисту інформації з повним спектром механізмів безпеки. Основними завданнями такого інтерфейсу є криптографічний захист інформації, переданої йому користувальним додатком, аутентифікація процесів встановлення з'єднання типу клієнт-сервер, аутентифікація власне переданих даних, конфіденційність і цілість переданої мережею загального користування інформації, неможливість одержувача відмовитися від факту одержання, захист від нав'язування інформації. При цьому додаток може не мати жодного уявлення про суть перетворень, виконуваних з даними.

Інтерфейс кореспондентської частини додатку сам визначить характер перетворення й виконає необхідні дії (розшифрує дані, перевірить ЕЦП та ін.). Основною перевагою створення такого уніфікованого інтерфейсу є його спрямування на зручне вмонтування в користувальні, комунікаційні й інші додатки. Споживач, не змінюючи власної технології опрацювання інформації, одержує потужний механізм криптографічного захисту інформації, що реалізовує вітчизняні стандарти. Монтаж може виконувати як сам споживач, так і сторонні організації. Динамічність захисту визначається наявністю в інтерфейсному модулі всіх необхідних сервісів захисту інформації. Вибір необхідного набору досягається простою операцією конфігурації інтерфейсу в процесі встановлення.

Для розв’язання цієї проблеми в 1993 році був вироблений і прийнятий стандарт RFC-1508 “Generic Security Service Application Program Interface”.

GSS-API став стандартом для додатків, що потребують захисту основних одиниць даних (таких, як файли або повідомлення) засобом, що не залежить від інших одиниць даних і не залежить від взаємодії з їхнім одержувачем. GSS-API підходить, наприклад, для таких додатків, як захищена електронна пошта. Захист, запропонований GSS, містить набір сервісів, таких, як справжня аутентифікація даних із їхньою цілістю, конфіденційність даних із їхньою цілістю, підтримка сервісів, що не дозволяють одержувачеві відмовитися від прийому. Надалі дані, що потребують захисту, одержувач може трансформувати (передати далі) або надіслати до архіву, де інформацію можна розсекретити через певний проміжок часу (день, рік та ін.).

Запровадимо поняття IDU - незалежна одиниця даних. IDU може бути будь-якої довжини. Додаток може при необхідності розбивати IDU на частини і застосовувати алгоритми криптографічного перетворення до кожної конкретної частини в різноманітні моменти, але підсумковий механізм захисту застосовується до цілого IDU. Проте IDU характерна тим, що є окремою одиницею даних, чий захист зовсім не залежить від інших одиниць даних. Якщо додаток захищає декілька IDU і відправляє всі їхньому звичайному одержувачеві, то захист із IDU може бути знятий цим одержувачем у будь-якому порядку протягом певного інтервалу часу, при цьому не допускається жодний логічний зв'язок між частинами при знятті з них захисту.

Як і в RFC-1508, визначення GSS-API забезпечує безпечні сервіси для клієнтів в єдиному стилі, підтримуваному рядом механізмів і технологій нижнього рівня і тому забезпечує локальним засобам захисту високий рівень можливості застосування у різноманітних середовищах.

Програма, що викликає - це будь-який додаток, що працює з IDU і викликає GSS-API з метою захисту цього IDU за допомогою таких сервісів, як DOA (data origin authentication) - аутентифікація даних із контролем цілісті, CONF (confidentiality) - конфіденційність із цілістю і/або підтримка неможливості відмови (наприклад, генерація ознаки, де ознака - інформація, що сама або разом з іншою інформацією використовується для дії) [1]. Програма, що викликає, впускає IDU всередину і

приймає захищений текст ззовні (прийнятий GSS-API захищений текст ззовні називається P-IDU (Protected)). Сервіси безпеки, доступні через механізм GSS-API, виконані через ряд механізмів нижчого рівня, заснованих на криптографії із секретним і/або відкритим ключем. Зауважимо, що GSS-API може перетворити вхідний потік інформації на масив інформації довільного розміру до застосування криптографічних сервісів захисту.

У процесі операції захисту вхідний буфер IDU (буфер даних між програмою, що викликає, і GSS-API) можна модифікувати (наприклад, дані можна зашифрувати або закодувати будь-яким засобом) або залишити не змінними. В будь-якому випадку результат називається M-IDU (Modified) із метою відрізнити його від початкового IDU. Залежно від бажання програми, що викликає, і можливостей механізму нижчого рівня шифртекст, зроблений механізмом захисту, може інкапсулювати, а може і не інкапсулювати M-IDU. Таким чином, P-IDU може бути просто шифртекстом (якщо інкапсуляція виконана) або може бути логічним з'єднанням шифртексту і M-IDU (якщо інкапсуляції не було). У другому випадку додаток, що забезпечує безпека, може вибирати будь-який метод з'єднання і комбінації шифртексту M-IDU у P-IDU за умови, що додаток, відповідальний за зняття захисту, повинен до виклику механізму GSS-API знати про те, як розподілити P-IDU обернено на його компоненти.

GSS-API розподіляє операції ініціалізації механізму безпеки (функція Establish_Env (.)) і операції забезпечення захисту кожного конкретного IDU. Об'єднання цих механізмів є комплексом захисту. Завжди при роботі модуля GSS-API спочатку ініціалізуються механізми безпеки, і лише потім виконуються конкретні криптографічні перетворення з незалежними одиницями даних. Захист кожного IDU і зняття цього захисту забезпечуються за допомогою таких сервісів, як DOA, CONF, неможливість відмови й ін., залежно від запитів додатка, що викликає, і підтримки механізмів нижчого рівня.

Приклад. У цьому прикладі демонструється курсування даних при обміні між відправником і одержувачем P-IDU у вигляді, незалежному від алгоритмів перетворення. Допускається, що обмін мандатами вже зроблений обома сторонами. Приклад не розкриває всіх можливих операцій, доступних у механізмі захисту і його зняття.

Відправник спочатку викликає функцію Establish_Env(.) для встановлення безпечного оточення. Потім для IDU, що необхідно передати, відправник викликає функцію Start_Protect(), функцію Protect() для кожного буфера даних і потім End_Protect() для завершення захисту IDU, що утворився P-IDU, що може бути шифртекстом і конкатенацією шифртексту і M-IDU (залежно від обрана чи доступна інкапсуляція), готовий до передання адресатові. Відправник викликає функцію Abolish_Env() для доповнення службовою інформацією.

Одержувач викликає функцію Establish_Env() для встановлення безпечного оточення. Потім для отриманого P-IDU викликає Start_Unprotect(),Unprotect() для кожного буфера даних і End_Unprotect() для завершення процесу зняття захисту P-IDU. Наприкінці одержувач викликає функцію Abolish_Env() для доповнення службовою інформацією.

Важливо відзначити, що не потрібна і не передбачається жодна синхронізація між розмірами буфера даних для обміну між GSS-API відправника й одержувача, а також розмірів буферів даних для обміну між програмою, що викликає, і GSS-API. Всі ці розміри вважаються незалежними. Мало того, розміри буферів даних, що використовуються для викликів механізмів захисту, функція тільки локального середовища, де ці виклики відбуваються.

Конструкція GSS-API має такі основні цілі:

- Незалежність механізму. GSS-API визначає інтерфейс до сервісів криптографічного забезпечення безпеки на основному рівні, що не залежить від будь-яких механізмів нижчого рівня. Наприклад, сервіси GSS-API можуть забезпечувати механізми шифрування як із секретним, так і з відкритим ключем.
- Незалежність від протоколів транспортного рівня. GSS-API не залежить від протоколів транспортного рівня, які можна використати для передачі P-IDU, допускаючи використання широкого ряду протоколів.
- Незалежність від середовища. Конструкція механізму безпеки GSS-API не впливає на конструкцію взаємодії комунікаційних протоколів. Тому сервіси GSS-API можна застосовувати додатками у різноманітних середовищах.

Мандатна структура забезпечує попередні вимоги для встановлення безпечного контексту між кореспондентами.

Програма, що викликає, повинна визначити, що використовуються мандати по умовчанню у випадку, якщо немає точних вказівок щодо них.

Просту мандатну структуру можна використати для встановлення вихідного контексту і прийому вхідного. Програма, що викликає, змушена оперувати тільки в одному з цих режимів і визначити цей факт тільки після процесу доставки мандатів. Елементи мандатів можуть містити складові криптографічні ключі (наприклад, для можливості аутентифікації і шифрування повідомлень, виконуваних різноманітними алгоритмами).

Після того, як мандати GSS-API будуть встановлені, перенесення цих мандатів на інші процеси або аналогічні конструкції всередині системи - це питання локальної політики безпеки, не обумовлене GSS-API. Наприклад: локальна політика безпеки така, що мандати видаються в процесі login і надають користувачеві user account або делегування прав у account, до якого робиться доступ, та ін.

Не декларується ніяке співвідношення (відповідність) між безпечним контекстом і комунікаційним протоколом. Цей поділ GSS-API можна використати у широкому ряді комунікаційних середовищ, він також спрощує послідовність індивідуальних викликів. У багатьох випадках (залежно від протоколу нижчого рівня, механізму взаємозв'язку і можливості кешування інформації), штатну інформацію, що потрібна для встановлення контексту, можна надіслати одночасно з уперше підписаними користувачем даними без додаткового обміну повідомленнями.

З метою встановлення безпечного контексту з обраною парою необхідно визначити відповідний механізм нижчого рівня (mech_type), що підтримує пара. Визначення механізму полягає не тільки у використанні особливого криптографічного перетворення (сполучення ряду криптографічних перетворень або вибір серед альтернативних), а також визначення синтаксису і семантики елементів даних для обміну, що служать для підтримки сервісів безпеки.

Рекомендується, щоб програми, що викликають і встановлюють контекст, визначали б mech_type по умовчанню, дозволяючи спеціальним системним функціям GSS-API або тим, що викликаються ними, вибирати відповідний mech_type. Проте програми, що викликають, можуть вказувати, що при необхідності буде застосований особливий mech_type.

GSS-API не створює проміжних імен для IDU. На випадок розподілу іменованого об'єкта на масив IDU, GSS-API передає ім'я об'єкта незмінним, виконуючи криптографічні перетворення лише з IDU. На цьому заснована одна з проблем перенесення GSS-API в різноманітних середовищах, оскільки він не має модуля

трансляції імен. Це питання легко розв'язується як організаційно, так і на рівні додатка, що викликає.

GSS-API містить концепцію канальної взаємодії (chan_binding) програм, що викликають, а це служить для встановлення безпечного контексту для передачі істотних характеристик (таких, як адреса, зміненого уявлення про ключі шифрування та ін.) і безпечного механізму передання інформації з цього каналу. Перевірка chan_binding інформації одною парою з двох, між якими встановлений контекст безпеки, дозволяє уникнути обом парам великої кількості активних атак.

Використання або невикористання можливостей GSS-API channel bindings - функція програми, що викликає, і GSS-API може підтримувати режим, у якому ця функція визначена як NULL.

Зауважимо, що GSS-API можна включити до різноманітних протоколів прикладного рівня типу клієнт-сервер, у т.ч. HTTP. Інтеграція HTTP і GSS-API відбувається на рівні мережного протоколу, за допомогою якого початкове з'єднання HTTP аутентифікується, а весь обмін захищений із використанням функцій GSS-API. При цьому захищені всі типи транзакцій, у т.ч. одержання document retrievals, form submissions і CGI results.

Інтеграція GSS-API ізолює протокол прикладного рівня і формат даних (HTML) від технології безпеки (наприклад, від формату криптографічного перетворення, процесу мандатного доступу та ін.). Такий підхід забезпечує ненав'язливу безпеку, в результаті якої Web-транзакції стають захищеними (інкапсульованими) без зміни конструкції стандарту HTTP/HTML.

Зараз вже існують комерційні реалізації GSS-API. Вони базуються на технологіях Kerberos, DCE, RSA. Ці реалізації містять інші мережні стандарти, такі як ASN. 1, X. 509, FIPS-113. [3]. Проте ці реалізації не задовольняють вітчизняних стандартів криптографічного захисту інформації.

Generic Security Service Application Program Interface (GSS-API) provides for a generic way of programming security systems and integrating them into the industrial processes. This report describes the basic aspects of GSS architecture and proposes its optimal application within modern automated industrial evironments.

Література

1. INTERNET-DRAFT <draft-ietf-cat-idup-gss-06. txt>, Nov. 26, 1996 *"Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)".*
2. RFC 1508**,** September 1993 *"Generic Security Service Application Program Interface".*
3. INTERNET-DRAFT <draft-ietf-wts-gssapi-00. txt> November 1995 *"Use of the GSS-API for Web Security".*

Одержано 18.06.2000 р.