

Jet

INFO

МАТЕРИАЛ
НОМЕРА

Сервер аутентификации **Kerberos**



А ТАКЖЕ:

- **НОВОСТИ INTERNET**
- **ХРОНИКА ОДНОГО ИНЦИДЕНТА**
- **ОБОБЩЕННЫЙ ПРИКЛАДНОЙ ПРОГРАММНЫЙ ИНТЕРФЕЙС СЛУЖБЫ БЕЗОПАСНОСТИ**
- **РУКОВОДИТЕЛЬ – РУКОВОДИТЕЛЮ**
- **ПАРТНЕРЫ JET INFOSYSTEMS**

12 / 13 1996



JAVA ПОМОГАЕТ БОРЬБЫ С ПРЕСТУПНОСТЬЮ

Компания PSI International, стратегический партнер Sun Microsystems Federal (подразделение Sun Microsystems), объявила о создании одного из первых продуктов на основе Java™, ориентированного на борьбу с преступностью.

Система получила название Internet in Blue™ ("Интернет в синем" — под цвет полицейских мундиров). В этом решении сочетаются мощь технологии Java, Интернет-серверов Netra™, среды разработки Java WorkShop™ (все это продукты Sun) и реляционной СУБД компании PSI. Предлагаемая система позволяет максимально полно использовать ресурсы Internet для целей правоохранительных органов, что достигается применением революционной технологии Java, обеспечивающей простоту, быстроту и независимость от операционной системы доступ к Интернет.

Система Internet in Blue, которую будет поставлять компания PSI International, предоставит полицейским участкам и другим правоохранительным органам средства для использования новейших технологий Интернет в борьбе с преступностью. С другой стороны, система позволит многим гражданам участвовать в этой борьбе. Они получат доступ к местному полицейскому участку средствами Интернет, смогут оперативно сообщать о возникающих проблемах, играя более активную роль в охране правопорядка в своем районе.

Помимо доступа к Интернет, система даст возможность полицейским участкам построить внутренний Интранет. Это позволит существенно улучшить внутреннюю информационную сеть, ускорить доступ к внутренним базам данных, которые со-

держат данные о местной преступности, в том числе схемы преступлений, состав преступных группировок, данные о подозреваемых и задержанных. До последнего времени зачастую приходилось хранить подобную информацию в бумажном виде, а это неэффективно.

"Правоохранительные органы очень быстро осознают значение Интернет и Интранет как стратегических средств в борьбе с преступностью, — сказал John Marselle, президент Sun Microsystems Federal, — Основанная на технологии Sun, система Internet in Blue использует преимущества средств безопасности Java, независимость Java от аппаратно-программной платформы и имеющиеся в Java механизмы доступа к базам данных. Система облегчит полицейским участкам и другим правоохранительным органам получение оперативного доступа и увеличит возможности их компьютерных сетей."

Java WorkShop, визуальная среда разработки для Java, может быть включена как составная часть в систему Internet in Blue. Используя ее в рамках полицейского участка, можно достаточно просто и быстро создавать, тестировать и развертывать Интернет- и Интранет-приложения на языке Java. Java WorkShop работает в среде Sun Solaris™ и Windows 95/NT. Посредством Java, СУБД PSIBase обеспечивает доступ к данным для навигаторов, работающих на различных аппаратных платформах.

"Интернет — это новый уровень в разработке информационных систем для правоохранительных органов, — сказал Paul Wormeli, директор программы создания систем для правоохранительных органов компании PSI, — Наша система Internet in Blue обеспечит им быстрейший доступ к таким

новейшим технологиям, как Java."

Internet in Blue включает:

- Начальный набор: сервер Sun Netra Internet и программные продукты и услуги PSI, необходимые для создания узла World Wide Web, а также техническую помощь в разработке программных продуктов и обучении персонала.
- Полный набор дополнительно включает прикладные средства для создания полицейского Интранет, обеспечивающего безопасность внутреннего доступа к определенной информации о преступности, стандартные полицейские процедуры и другие сервисы инфраструктуры. Частью системы является интерактивная, основанная на сообщениях коммуникационная среда для передачи в правоохранительные органы информации криминального характера.
- В качестве частичной компенсации стоимости системы, правоохранительные органы могут предоставлять отчетную информацию о преступлениях, инцидентах и т.п. внешним организациям, например юридическим фирмам и страховым компаниям.

Продукт Internet in Blue будет доступен, начиная с июля 1996 г.

Справка. PSI International, Inc. в течение 14 лет ведет работы как системный интегратор для правительственных учреждений на всех уровнях — от федерального до местного. Компания специализируется на решениях для правоохранительных органов и на продуктах в области систем общественной безопасности. В ее составе свыше 250 профессионалов, владеющих практически всеми современными технологиями.

JAVA ДЛЯ КЛИЕНТОВ БАНКОВ

Согласно неофициальной информации, компания NCR переписала на языке Java клиентскую часть своего программного обеспечения для обработки транзакций в сети банкоматов. Речь идет о системе Top End. Официальное объявление ожидается в середине июня.

После переписывания клиентская часть превратилась в

JAVA-УСКОРЕНИЕ

В конце мая компания Asymetrix Corp. объявила о выпуске среды визуального программирования SuperCede для языков Java и C++. Важнейшим элементом этой среды является Flash Compiler — компилятор, способный транслировать программы "на лету". Утверждается, что он ускоряет выполнение Java-программ по сравнению с обычной интерпретацией в 50 раз и делает в 5 раз более быстрые программы по сравнению с другими компиляторами, работающими "на лету".

Важным технологическим элементом является виртуальная Java-машина (SuperCede VM), во многом определяющая эффективность работы Java-программ.

Предполагается, что основными потребителями среды SuperCede станут изготовители Java-приложений (таких, напри-

мерно Java-апплет размером 50 Кб, который можно загрузить с помощью Web-навигатора, однако затем он (апплет) начинает общаться с сервером Top End напрямую. Это сделано не только по соображениям безопасности; сообщается, что продукт NCR, по сравнению с навигатором Netscape, обеспечивает увеличение скорости обработки транзакций в 5 — 10 раз. Что касается собст-

венно информационной безопасности, то утверждается, что новый продукт соответствует стандартам, принятым в банковской индустрии.

Переписывание на Java сделало клиентскую часть доступной практически для всех аппаратно-программных платформ. Пользователи будут получать ее бесплатно.

мер, как Web-навигаторы), которым новый компилятор и виртуальная машина позволяют резко увеличить эффективность. Как заявил Эрик Шмидт (Sun Microsystems), "SuperCede снимает вопросы, связанные с эффективностью Java-систем, и тем самым укрепляет позиции Java как стандартного языка для создания Интернет-приложений".

Пишущие на C++ благодаря новой среде получают возможность в диалоговом режиме во время выполнения изменять разрабатываемые программы без потери эффективности исполняемых кодов. Таким образом, программисты смогут работать с такой скоростью, с какой они думают — среда разработки перестанет их тормозить.

В настоящее время свободно доступна бета-версия вирту-

альной Java-машины для платформ Windows 95/NT (см. <http://www.asymetrix.com>). Предполагается, что ее промышленная версия станет доступной в июле 1996 года. Бета-версия SuperCede for Java ожидается в июле, а финальная версия — в октябре 1996 года. Версия SuperCede for C++ станет доступной в 1997 году.

Компания Asymetrix Corp. планирует в дальнейшем выпустить на рынок средства разработки Интернет-приложений в трехуровневой архитектуре, а также монитор распределенных транзакций, координирующих работу СУБД, входящих в состав Web-серверов или являющихся самостоятельными элементами корпоративных сетей. Очевидно, это укрепит позиции сторонников технологии Интранет.

JAVA-АПЛЕТЫ СМОГУТ УПРАВЛЯТЬ СЕТЬЮ

Согласно заявлению представителей компании SunSoft, ими разработана среда Solstice Workshop, предназначенная для создания Java-апплетов, управляющих корпоративными сетями. В основе новой среды лежит программный инструментальный Java Workshop и прикладной программный интерфейс Java Management, спроектированный в сотрудничестве с компаниями Cisco Systems и Novell. Идея состоит в том, чтобы сделать сетевые управляющие компоненты мобильными и доступными для многократного использования.

Компания SunSoft планирует в течение следующего года выполнить "явизацию" своего продукта Solstice Domain Manager, преобразовав консольный компонент для поддержки так называемого навигационного интерфейса (BUI — Browser User Interface). Несомненно, навигационный интерфейс является естественным не только при разработке программ, но и при управлении сетями.

Продукт Solstice Domain Manager является средним звеном в трехуровневой архитектуре систем управления корпоративными сетями, предлагаемой компанией

SunSoft. Он позволяет объединить в одну область управления до 10 тысяч узлов.

Предполагается, что программный интерфейс Java Management поддержат такие компании, как AutoTrol, Bay Networks, BGS, BMC, Century Design Systems, Computer Associates, Compuware, Landmark Technologies, Legato Systems, OpenVision, Platinum Technologies, Tivoli Systems, 3Com.

Более подробную информацию можно найти по адресу <http://www.sun.com/solstice/products/workshop.html>



Хроника одного инцидента

13 мая в 14:38 была зарегистрирована попытка получения списка имен всех компьютеров в домене jet.msk.su с компьютера 194.XX.XX.94 (jumbo.XX.XX.ru)*. Такие попытки происходят регулярно, в большинстве случаев они не преследуют целей несанкционированного доступа к информации. Попытки получить такой список производят всякого рода статистические роботы, исследующие Интернет, любознательные пользователи, роботы, проверяющие правильность конфигурации DNS и другие.

Подобные попытки фиксируются в системных журналах и хранятся одну неделю, но не считаются достаточно важными, чтобы специально уведомлять системных администраторов (протокол 1).

* — Поскольку инцидент был улажен на уровне системных администраторов, и формальные действия, необходимые в подобных случаях (информирование CERT — Computer Emergency Response Team, группы реагирования на нарушения информационной безопасности) не предпринимались, мы решили не указывать подлинных имен компьютеров и пользователей.

Наш сервер DNS сконфигурирован таким образом, что передача имен компьютеров запрещена. Это сделано для того, чтобы минимизировать выдаваемую наружу информацию об устройстве внутренней сети Компании. Поэтому попытка получить имена не удалась. По всей видимости, это заинтересовало того, кто исследовал нашу систему, и он решил пойти дальше. Определить, что из Интернет доступна машина relay1.jet.msk.su, можно разными способами. Этот компьютер выполняет функции сервера DNS для наших доменов, принимает (и отправляет) электронную почту. Он же является межсетевым экраном (firewall).

Следующим шагом была попытка идентифицировать систему, установленную на этой машине. Часто это можно сделать, открыв соединение по протоколу telnet. Большинство операционных систем настроены изготовителем таким образом, что они сообщают название компьютера и операционной системы, например UNIX(r) System V Release 4.0 (gandalf) login:

Или таким:
SunOS UNIX (bilbo)
login:

Было установлено соединение по протоколу telnet. В ответ на такую попытку наша машина выдает приглашение:

Username:

не передавая никакой иной информации.

На данном этапе все эти действия не являлись атакой — не было попытки получить доступ в чужую компьютерную систему. Однако дальше была зарегистрирована попытка получить такой доступ под широко распространенными идентификаторами пользователей (протокол 2).

В ответ на все эти имена система выдавала диагностику:

Userid nonexistent
(несуществующий идентификатор пользователя), так как доступ из Интернет разрешен только пользователям, зарегистрированным в специальной базе данных.

Атакующий решил использовать SMTP — протокол обмена почтой — для поиска правильных идентификаторов пользователей (протокол 3).

①

May 13 14:38:16 relay1 named[86]: unapproved AXFR from [194.XX.XX.94].34887 for jet.msk.su

②

May 13 14:41:44 localhost tn-gw[12970]: permit host=jumbo.XX.XX.ru/194.XX.XX.94 use of gateway
May 13 14:41:50 localhost authsrv[12971]: BADAUTH guest (tn-gw jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:41:56 localhost authsrv[12971]: BADAUTH root (tn-gw jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:42:04 localhost authsrv[12971]: BADAUTH bin (tn-gw jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:42:41 localhost tn-gw[12970]: exit host=jumbo.XX.XX.ru/194.XX.XX.94. 94 no auth

③

May 13 14:42:44 localhost smap[12977]: connect host=jumbo.dist.XX.XX.ru/194.XX.XX.94
May 13 14:42:56 localhost smap[12977]: expn root (jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:43:01 localhost smap[12977]: expn bin (jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:43:09 localhost smap[12977]: expn postmaster (jumbo.XX.XX.ru/194.XX.XX.94)
May 13 14:43:15 localhost smap[12977]: expn info (jumbo.XX.XX.ru/194.XX.XX.94)

Команда **exrn** позволяет проверить, зарегистрирован ли на данной машине пользователь с соответствующим идентификатором, и узнать, что происходит с почтой, адресованной ему. Например, с помощью команды **exrn postmaster** обычно можно определить реальный идентификатор пользователя, отвечающего за работоспособность почтовой системы. В установленной почтовой системе команда **exrn** подтверждает существование любого пользователя — мы не заинтересованы в том, чтобы кто угодно мог получить информацию о наших пользователях (протокол 4).

По всей видимости, сейчас это понял и атакующий.

В это же время система безопасности обнаружила три неудачные попытки идентификации подряд и отправила соответствующий фрагмент журнала по электронной почте системному администратору. Я прочитал это сообщение только в 15:14, соединился с машиной, подвергавшейся атаке, и попробовал получить список активных пользователей на том компьютере, с кото-

рого производилась атака (протокол 5).

Тем временем атакующий повторно запустил telnet и перепробовал еще несколько идентификаторов:

```
alex, admyn, serg, admin,
boris, ivan, fuck
```

Последний из них, по всей видимости, свидетельствовал о том, что атакующий понял, что у него ничего не получится, и прекратил атаку.

Атака на этом действительно закончилась, но мы приняли решение найти виновника. Сначала решили определить, что такое XX.ru.

Домены второго уровня *.ru регистрируются Российским Институтом Развития Общественных Сетей — РОСНИИ-РОС. Поиск такого домена по его базе данных (протокол 6) позволил определить название организации, на которую зарегистрирован домен XX.ru, и имя контактной персоны.

На адреса:

```
hostmaster@XX.ru
postmaster@XX.ru
root@XX.ru
```

```
postmaster@dist.XX.ru
hostmaster@dist.XX.ru
root@dist.XX.ru
```

было отправлено письмо 1.

Вечером того же дня было получено послание, которое можно рассматривать, как ответ (письмо 2).

Обратите внимание на обратный адрес — intruder@jet.msk.su. Адрес подделан, заголовки письма показывают, что оно было отправлено с машины ns.cs.msu.su. Подделка адреса производится достаточно легко. Такое действие можно рассматривать как отдельную атаку, но мы решили ее проигнорировать.

На следующий день ответственные лица прислали свои извинения, а также информацию о пользователе, чьи действия были зарегистрированы нашей системой (письмо 3).

В ответ мы указали, что попытка получить доступ в систему — несколько экстравагантный способ понять ее тип (письмо 4).

Ответ (письмо 5) нас удовлетворил:

На этом дело было закрыто.

④

```
May 13 14:43:20 localhost smap[12977]: exrn f (jumbo.XX.XX.ru/194.XX.XX.94
```

⑤

```
relay1% date
Mon May 13 15:14:41 GMT+0400 1996
relay1% finger @jumbo.XX.XX.ru
[jumbo.XX.XX.ru]
Login      Name          TTY      Idle   When   Where
ruslan     Ruslan ..... pts/0    15 Sun 16:46 mason
boris      Boris .....   pts/1    1:51 Sun 17:17 valkiria
root       0000-Admin(0000) pts/2    1:51 Sun 17:18 valkiria
bsu        Serge .....   pts/3    55 Sun 17:51 sirius
vadim      ..... Vadim   pts/4    1 Sun 18:18 banshie
```

⑥

```
% whois -h whois.ripn.net XX.ru
domain:      <Domain name>
descr:      <Company name>
descr:      Russia
admin-c:     <person name>
tech-c:     <person name>
zone-c:     hostmaster@XX.ru
nserver:    ns.demos.su 194.87.0.8 194.87.0.9
nserver:    ns.elvis.ru 192.153.171.36
sub-dom:    pub lab soft pro cda
source:     RIPN
```

>From avk Mon May 13 17:33:32 1996
 Subject: attack from jumbo.XX.XX.ru
 To: hostmaster@XX.XX.ru, hostmaster@XX.ru, root@jumbo.XX.XX.ru,
 root@XX.ru, postmaster@jumbo.XX.XX.ru, postmaster@XX.XX.ru,
 postmaster@XX.ru
 Date: Mon, 13 May 1996 17:33:32 +0400 (MSD)



Господа:
 13.05.96 в интервале с 14:41 до 15:08 MSD с машины jumbo.XX.XX.ru (194.XX.XX.94) были произ-
 ведены попытки неавторизованного доступа к компьютерной системе "Инфосистемы Джет". Зарегистриро-
 ваны попытки получить информацию через SMTP EXPN и получить доступ к системе по протоколу TELNET.
 При доступе через TELNET использовались имена:

root, bin, alex, admyn, serg, admin, boris, ivan, fuck.

В 15:14 с машины relay1.jet.msk.su была дана команда finger@jumbo.XX.XX.ru. Выдача команды
 finger следует:

```
[jumbo.XX.XX.ru]
Login      Name                TTY      Idle   When   Where
ruslan    Ruslan .....          pts/0    15 Sun 16:46 meson
boris     Boris .....           pts/1    1:51 Sun 17:17 valkiria
root      0000-Admin(0000)       pts/2    1:51 Sun 17:18 valkiria
bsu       Serge .....           pts/3    55 Sun 17:51 sirius
vadim     ..... Vadim           pts/4    1 Sun 18:18 banshie
```

Прошу произвести анализ этого инцидента и держать меня (hostmaster@jet.msk.su) в курсе.

Спасибо
 Andrew V. Kovalev +7 095 973 4849

>From jet.msk.su!intruder Mon May 13 19:57:00 1996
 Received: from jet.msk.su by jet.msk.su ; Mon, 13 May 96 19:57 MSD
 Received: from jet.msk.su by jet.msk.su ; Mon, 13 May 96 19:55 EET DST
 Received: by jet.msk.su; Mon, 13 May 96 19:55 GMT+4:00
 Message-Id: <m0uIzxy-0004R4C@jet.msk.su>
 Date: Mon, 13 May 96 19:55 GMT+4:00
 From: intruder@jet.msk.su
 To: root@jet.msk.su
 Received: from ns.cs.msu.su(158.250.10.2) by relay1.jet.msk.su
 id sma015665; Mon May 13 19:49:58 1996
 Status: 0



Hi
 u said about intrusion - it's wrong, nothing criminal, just exploring your site simply i have
 not ever met such system , don't bother Vadim
 P.S. Last "fuck" - i was tired and lived ;)

Андрей!

Завещаю, что с нашей стороны не было злых намерений. Просто один из наших сотрудников пытался
 понять тип вашей ОС (выдает Username как VMS или cisco) ну и сходил через SMTP - она обычно
 что-нибудь говорит о системе

Борис.



Насчет smtp - оно, конечно, может быть, хотя я не очень понимаю, как может команда expn root
 показать, что за система.
 Но вот при чем тут попытки зайти telnet'ом под root, bin и др? Это какой-то новый способ
 понять тип ОС.

avk



Ну это уже было озорство. Во всяком случае, мы сделали ему внушение о недопустимости таких
 действий и обязали извиниться.
 Надеюсь впредь такого не повторится.





Сервер аутентификации Kerberos

Содержание

1. Введение	7
2. От чего защищает и от чего не защищает Kerberos	7
3. Неформальное изложение возможностей Kerberos ...	8
4. Флаги, используемые в билетах	11
5. Форматы данных в системе Kerberos	13
6. Типы пересылаемых сообщений.	14
7. База данных Kerberos. .	15
8. Предлагаемые направления развития системы Kerberos.	16
Приложение 1. Терминологический словарь	16

1. Введение

Идентификация и проверка подлинности пользователей (аутентификация) — это основное средство защиты информационных систем от одной из главных угроз — постороннего вмешательства. Если у злоумышленника нет средств для нелегального доступа, возможности информационного нападения существенно уменьшаются.

Под идентификацией мы будем понимать процедуру, посредством которой пользователь или компьютерный процесс сообщает сведения о себе (называет себя). Проверка подлинности, или аутентификация — это процедура проверки достоверности предъявленных данных.

Современные информационные системы представляют собой совокупность разнородных (реализованных на различных аппаратно-программных платформах), территориально разнесенных компонентов. Как правило, большинству пользователей требуется доступ ко

многим сервисам, предоставляемым системами. В то же время ни им, ни системным администраторам не хочется размножать регистрационную информацию и особым образом входить в каждый сервер.

В соответствии с технологией клиент/сервер, выход заключается в создании специального сервера проверки подлинности, услугами которого будут пользоваться другие серверы и клиенты информационной системы. На сегодняшний день на роль фактического стандарта сервера аутентификации имеется только один реальный претендент — Kerberos, продукт, разработанный в середине 1980-х годов в Массачусетском технологическом институте и претерпевший с тех пор ряд принципиальных изменений. Клиентские компоненты Kerberos присутствуют в большинстве современных операционных систем, в такие системы, как Solaris, Kerberos проник весьма глубоко, а концерн OSF сделал Kerberos частью своей распределенной компьютерной среды (DCE).

Выделение особого сервера аутентификации позволяет реализовывать собственные прикладные системы со своей системой понятий, но со стандартной процедурой проверки подлинности, что существенно облегчает управление правами доступа пользователей. Kerberos (точнее, его версия, реализованная в MIT) является свободно распространяемым программным продуктом, доступным в исходных текстах (кроме, быть может, криптографических компонентов). Таким образом, в руках разработчиков прикладных систем и лиц, отвечающих за информационную безопасность,

оказывается не "черный", а "прозрачный ящик", функционирующий понятным и проверяемым образом и доступный для "подгонки" с учетом нужд конкретной организации.

2. От чего защищает и от чего не защищает Kerberos

Kerberos предоставляет средства проверки подлинности субъектов, работающих в открытой (незащищенной) сети, то есть сети, не исключающей возможности перехвата, модификации и пополнения пересылаемой информации. Kerberos не полагается на средства аутентификации, реализованные в операционных системах сетевых компьютеров, на подлинность сетевых адресов, на физическую защищенность сетевых компьютеров (кроме тех, на которых работает сервер Kerberos).

С точки зрения реализации Kerberos — это один или несколько серверов проверки подлинности, функционирующих на физически защищенных компьютерах. Серверы поддерживают базу данных субъектов и их секретных ключей. В данном документе не рассматривается протокол администрирования, позволяющий безопасным образом изменять базу данных субъектов, а также протокол репликации этой базы.

Kerberos не защищает от:

- атак на доступность. Отражение подобных атак (как и реакция на "нормальные" отказы) возлагается на пользователей и администраторов.
- кражи секретных ключей. Субъекты должны сами заботиться о секретности своих ключей.

- угадывания паролей. Плохо выбранный пароль может не устоять против подбора с помощью словаря, вычисления по паролю секретного ключа с последующей попыткой расшифровать полученную от Kerberos информацию. Соответствующим образом измененная программа ввода пользовательского пароля ("тroyанский конь") позволит злоумышленнику узнать пароли многих пользователей. Таким образом, Kerberos все же предполагает некоторый уровень защищенности обслуживаемых компьютеров.
- повторного использования идентификаторов субъектов. В принципе возможна ситуация, когда новый субъект Kerberos получит тот же идентификатор, что был у ранее выбывшего субъекта. Возможно, что этот идентификатор остался в списках управления доступом какой-либо системы в сети. В таком случае новый субъект унаследует права доступа выбывшего.
- рассогласования часов на компьютерах. Kerberos полагается на приблизительную согласованность часов сетевых компьютеров. Подобное предположение упрощает обнаружение воспроизведения (дублирования) информации. Допустимая погрешность часов может устанавливаться индивидуально для каждого сервера. Если синхронизация часов выполняется сетевыми средствами, соответствующий протокол должен сам заботиться о безопасности.

3. Неформальное изложение возможностей Kerberos

Система Kerberos предназначена для решения следующей задачи. Имеется открытая (незащищенная) сеть, в узлах

которой сосредоточены субъекты — пользователи, а также клиентские и серверные программные системы. Каждый субъект обладает секретным ключом. Чтобы субъект *S* мог доказать свою подлинность субъекту *S* (без этого *S* не станет обслуживать *C*), он должен не только назвать себя, но и продемонстрировать знание своего секретного ключа. *C* не может просто послать *S* свой секретный ключ, во-первых, потому, что сеть открыта (доступна для пассивного и активного прослушивания), а, во-вторых, потому, что *S* не знает (и не должен) знать секретный ключ *C*. Требуется менее прямолинейный способ демонстрации знания секретного ключа.

Система Kerberos представляет собой надежную третью сторону (то есть сторону, которой доверяют все), владеющую секретными ключами обслуживаемых субъектов и помогающую им в попарной проверке подлинности.

Чтобы с помощью Kerberos получить доступ к *S* (обычно это сервер), *C* (как правило — клиент) посылает Kerberos'у запрос, содержащий сведения о нем (клиенте) и о запрашиваемой услуге. В ответ Kerberos возвращает две порции информации: так называемый билет, зашифрованный секретным ключом сервера, и копию части информации из билета, зашифрованную секретным ключом клиента. Клиент должен расшифровать вторую порцию данных и переслать ее вместе с билетом серверу. Сервер, расшифровав билет, может сравнить его (билета) содержимое с дополнительной информацией, присланной клиентом. Совпадение свидетельствует о том, что клиент смог расшифровать предназначенные ему данные (ведь содержимое билета никому, кроме сервера и Kerberos, недоступно), то есть продемонстрировал знание своего секрет-

ного ключа. Значит, клиент — именно тот субъект, за кого себя выдает. Подчеркнем, что секретные ключи в процессе проверки подлинности не передавались по сети (даже в зашифрованном виде) — они только использовались для шифрования. Как организован первоначальный обмен ключами между Kerberos и субъектами и как субъекты хранят свои секретные ключи — вопрос отдельный.

Проиллюстрируем описанную процедуру рисунком 1.

Изложенный вариант 1 — крайне упрощенная версия реальной процедуры проверки подлинности. Прежде всего, необходимо уточнить структуру пересылаемых запросов и возвращаемых ответов. И билет, и вторая порция информации, предоставляемая со стороны Kerberos, содержат случайный ключ, пригодный для шифрования сообщений, пересылаемых между клиентом и сервером. Этот ключ называется сеансовым и представляет собой общую секретную информацию, разделяемую *C* и *S*. Появление подобной общей информации и возможность организации конфиденциального взаимодействия — важный побочный продукт аутентификации средствами Kerberos.

Билет, который выдает Kerberos, имеет ограниченный срок годности. Естественно распространить этот срок и на сеансовый ключ. Когда срок годности истекает, необходимо проведение повторной проверки подлинности. Продолжительность стандартного срока годности зависит от политики безопасности, проводимой организацией. Типичный срок годности — 8 часов.

Вполне вероятно, что клиент *C* захочет убедиться в подлинности обслуживающего его сервера *S*. Чтобы это было возможным, клиент шифрует дополнительную информацию, пе-

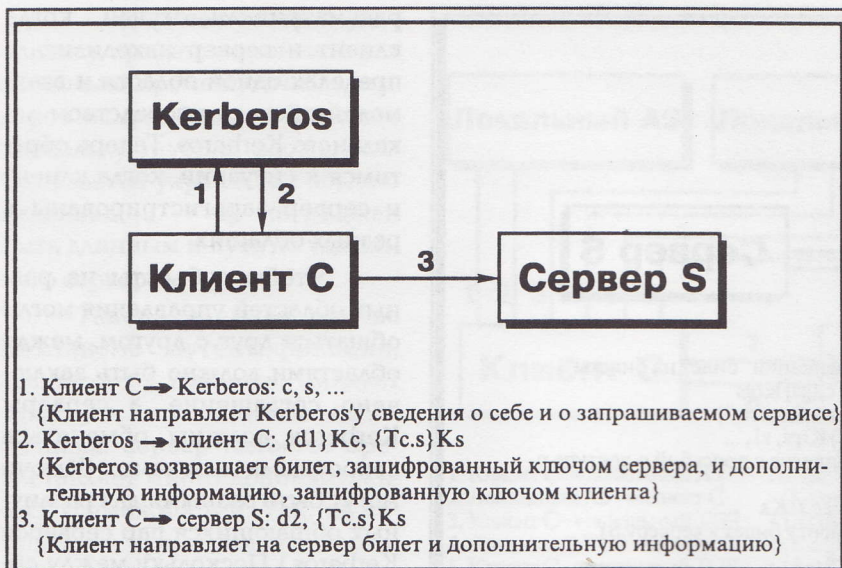


Рис. 1. Проверка сервером S подлинности клиента C (вариант 1). Здесь: c и s—сведения (например, имя), соответственно, о клиенте и сервере, d1 и d2—дополнительная (по отношению к билету) информация, Tc.s—билет для клиента C на обслуживание у сервера S, Kc и Ks—секретные ключи клиента и сервера, {info}K—информация info, зашифрованная ключом K.

редаваемую серверу, с помощью сеансового ключа. Сервер может узнать сеансовый ключ, только расшифровав билет. Если это случится и сервер вернет клиенту свидетельство того, что он сумел прочитать дополнительную информацию, C вправе считать подлинность сервера S установленной.

Далее, необходимо защитить информацию, пересылаемую по сети, от воспроизведения. Если этого не сделать, злоумышленник сможет послать серверу дубликаты билета и дополнительной информации и успешно выдать себя за "второго C". Стандартный способ защиты от воспроизведения — включение в нее временных штампов и/или так называемых одноразовых номеров, позволяющих удостовериться в "свежести" сообщений и в их неповторяемости (и, более того, в целостности последовательности сообщений). Одноразовые номера позволяют также ассоциировать ответы с предыдущими запросами.

Отобразим упомянутые уточнения на новом рисунке (рис. 2).

Отдельного пояснения заслуживает обозначение s1 в первом запросе. Вообще говоря, клиенту нужен не конкретный сервер, а определенный сервис, о чем он и просит Kerberos. В ответ клиент получает адрес сервера

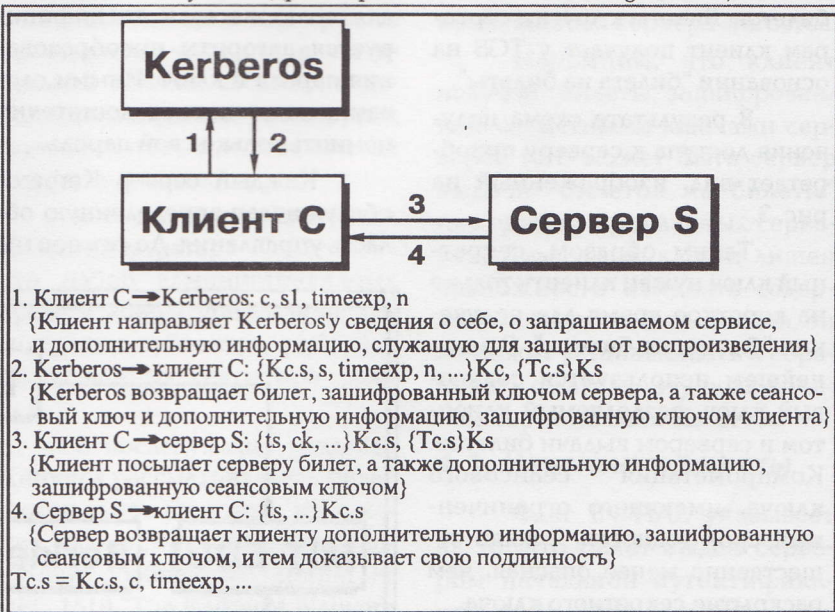


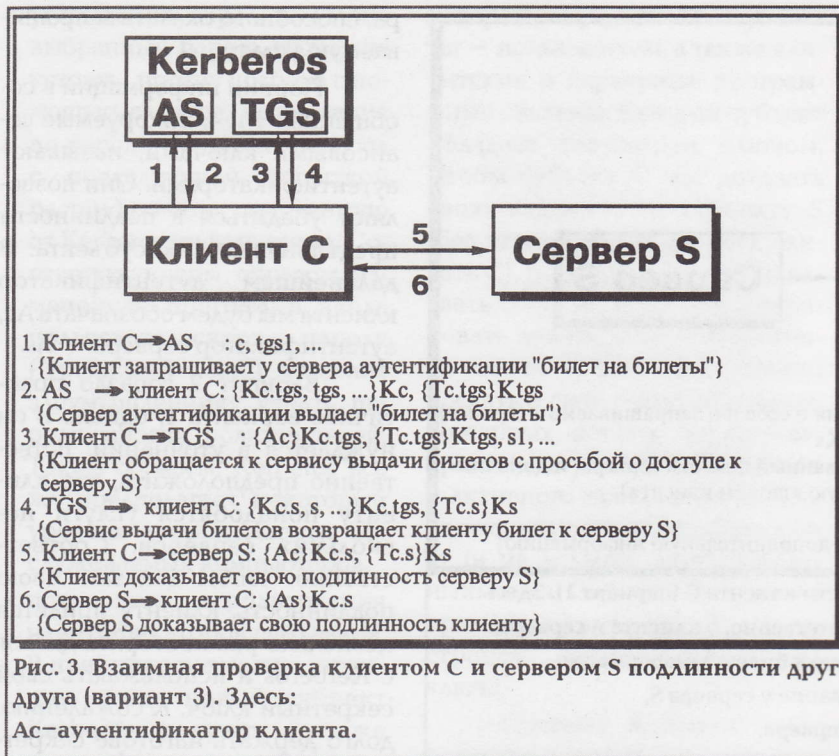
Рис. 2. Взаимная проверка клиентом C и сервером S подлинности друг друга (вариант 2). Здесь:

timeexp—срок годности билета,
n—одноразовый номер,
Kc.s—сеансовый ключ,
ts—временной штамп,
ck—контрольная сумма.

ра, способного оказать запрашиваемую услугу.

Порции информации в сообщениях 3 и 4, шифруемые сеансовыми ключами, называют аутентификаторами. Они позволяют убедиться в подлинности предъявившего их субъекта. В дальнейшем аутентификатор клиента мы будем обозначать Ac, аутентификатор сервера — As.

Вариант 2 гораздо практичнее варианта 1, однако и он нуждается в уточнении. Естественно предположить, что клиенту понадобятся услуги нескольких серверов. Соответственно, чтобы доказать свою подлинность, клиенту придется несколько раз повторять диалог с Kerberos и использовать свой секретный ключ. К сожалению, долго держать наготове секретный ключ опасно — его могут украсть. Чтобы справиться с указанной проблемой, сервер Kerberos "раздваивается" на сервер начальной аутентификации (AS — Authentication Server) и сервер выдачи билетов (TGS — Ticket Granting Server). Клиент



запрашивает у AS по схеме, изображенной на рис. 2, билет к TGS ("билет на получение билетов", "билет на билеты" - TGT, Ticket-Granting Ticket). TGT содержит сеансовый ключ для зашифрования общения между клиентом и сервером выдачи билетов. Билеты к другим серверам клиент получает у TGS на основании "билета на билеты".

В результате схема получения доступа к серверу приобретает вид, изображенный на рис. 3.

Таким образом, секретный ключ нужен клиенту только на короткое время для получения "билета на билеты". В дальнейшем используется сеансовый ключ, разделяемый клиентом и сервером выдачи билетов. Компрометация сеансового ключа, имеющего ограниченный срок годности, - вещь существенно менее опасная, чем раскрытие секретного ключа.

Любопытно отметить, что AS, вопреки своему названию, не проверяет подлинности обратившихся к нему субъектов. Билет на билеты может получить кто угодно, в том числе и злоумышленник, но воспользоваться

этим билетом он сможет, только если знает секретный ключ субъекта, от имени которого он выступает. Отметим также, что если для программных субъектов (клиентов и серверов) способ хранения секретного ключа не определен, то для пользователей специфицируется алгоритм преобразования пароля в ключ. Иными словами, пользователю достаточно помнить только свой пароль.

Каждый сервер Kerberos обслуживает определенную область управления. До сих пор мы

рассматривали случай, когда клиент и сервер находились в пределах одной области и взаимодействовали посредством локального Kerberos. Теперь обратимся к ситуации, когда клиент и сервер зарегистрированы в разных областях.

Чтобы субъекты из разных областей управления могли общаться друг с другом, между областями должно быть заключено соглашение, а серверы Kerberos должны обмениваться секретными ключами. (Ключи могут быть разными для различных общающихся пар серверов Kerberos.) Поскольку между областями управления Kerberos существуют стандартные иерархические отношения, каждой области достаточно заключить соглашение и обменяться ключами с родителем и всеми потомками. В принципе возможно заключение соглашения между двумя произвольными областями, субъекты которых активно общаются друг с другом. Рисунок 4 иллюстрирует отношения между областями управления.

При наличии договорных отношений локальный сервер выдачи билетов может выдать билет к удаленному TGS, который, в свою очередь, выдаст билет к удаленному серверу. Последовательность обмена сообщениями для этого случая показана на рис.5.

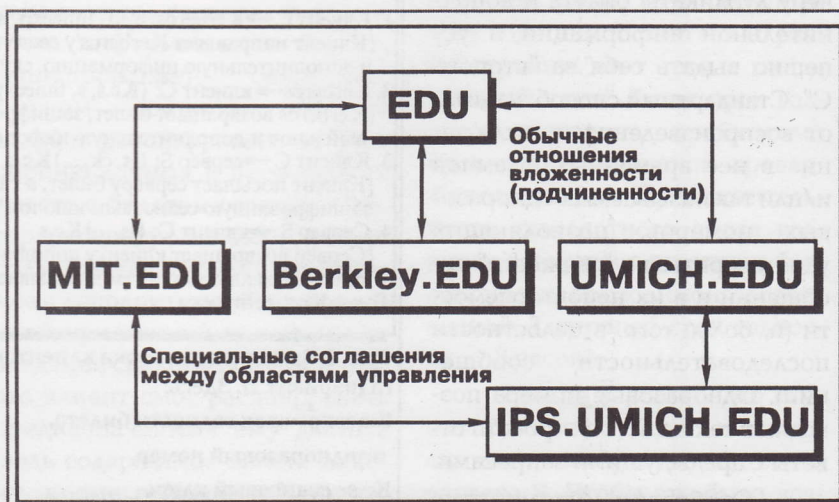


Рис. 4. Отношения между областями управления Kerberos.

В более общем случае клиенту придется пройти через несколько промежуточных серверов выдачи билетов. Впрочем, поскольку глубина вложенности областей управления обычно невелика (3 – 4), не должен быть длинным и путь до удаленного сервера.

Такова идейная основа механизма аутентификации, принятого в Kerberos. Следует отметить, что, помимо изложенных, сервер Kerberos предоставляет много дополнительных возможностей. Так, на самом деле время жизни билета задается как пара (начальное время, конечное время). В результате можно получать билеты "на потом", например, для пакетных заданий, выполняющихся ночью. Далее, имеется механизм билетов с правом передачи, что позволяет серверам выполнять определенные действия от имени обратившихся к ним клиентов (скажем, серверу печати осуществлять доступ к файлам пользователя). Есть и другие средства, на которых, однако, мы сейчас останавливаться не будем.

На реализацию и использование Kerberos можно смотреть с разных точек зрения. Пользователь видит Kerberos как набор команд (например, kinit – начало Kerberos-сеанса, kdestroy – разрушение Kerberos-билетов и окончание сеанса, klist – выдача текущего списка билетов, kpasswd – смена Kerberos-пароля и т.п.) или не видит его вообще, если соответствующие вызовы встроены в утилиты и команды операционной системы, такие как login, logout, passwd.

Для программиста Kerberos представляет собой набор библиотек, весьма разумно структурированных, так что можно легко заменять разные компоненты, например, криптографические. Две функции основной библиотеки –

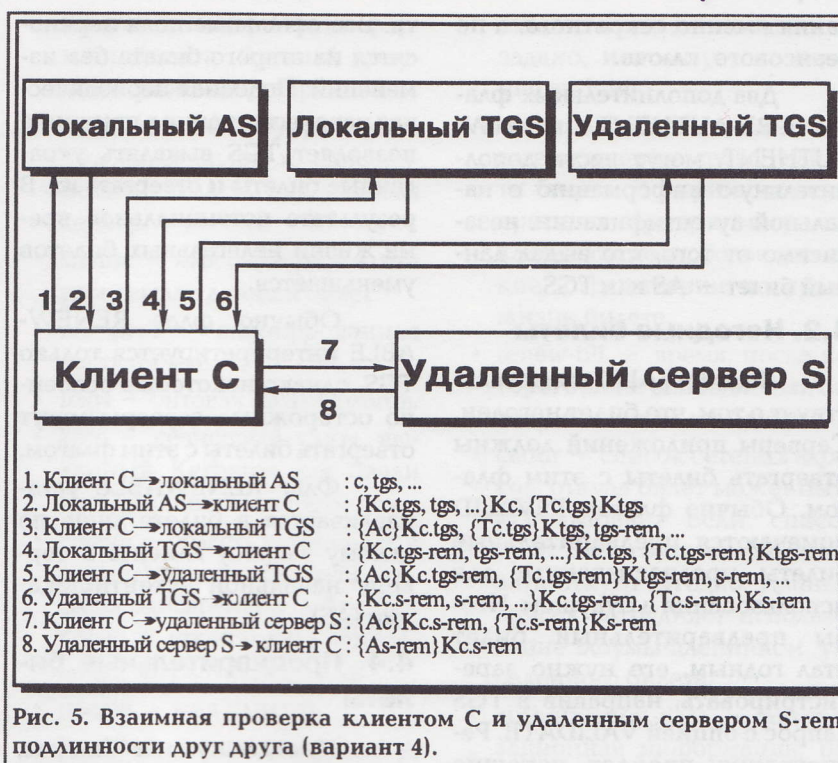


Рис. 5. Взаимная проверка клиентом С и удаленным сервером S-rem подлинности друг друга (вариант 4).

krb_mk_req и krb_rd_req, позволяют, соответственно, построить аутентификационный запрос к серверу и обработать его. При этом прикладной программист избавляется от рутинных аспектов общения с Kerberos. "Керберизация" приложений, то есть встраивание в них средств проверки подлинности, в значительной степени сводится к добавлению вызовов упомянутых функций, что, естественно, не слишком сложно.

С точки зрения системного администратора Kerberos – это набор административных средств конфигурирования, регистрации субъектов, работы с базой данных секретных ключей.

В последующих разделах Kerberos рассматривается более формально. Основой изложения является официальный документ сообщества Internet – RFC 1510 "The Kerberos Network Authentication Service (V5)".

4. Флаги, используемые в билетах

Каждый Kerberos-билет содержит набор флагов. Большинство из них запрашивается

клиентом перед получением билета. Некоторые флаги автоматически устанавливает и сбрасывает сам Kerberos. Рассмотрение флагов и способов их использования позволяет в полном объеме продемонстрировать аутентификационные возможности сервера Kerberos.

Напомним, что клиент получает билеты, зашифрованные секретными ключами серверов (это может быть сервер выдачи "билетов на билеты" или один из прикладных серверов). Тем самым клиент лишен возможности изменять содержимое билетов. В частности, он не может устанавливать и сбрасывать флаги – эту роль целиком берет на себя Kerberos.

4.1. Начальные билеты

Флаг INITIAL указывает на то, что билет выдан сервером начальной аутентификации (AS). Серверы приложений, которым нужно знать секретный ключ пользователя (пример – программа изменения пароля), могут принимать билеты только с установленным флагом INITIAL как свидетельство недавнего использо-

вания именно секретного, а не сеансового, ключа.

Два дополнительных флага, PRE-AUTHENT и HW-AUTHENT, могут нести дополнительную информацию о начальной аутентификации, независимо от того, кто выдал данный билет — AS или TGS.

4.2. Негодные билеты

Флаг INVALID свидетельствует о том, что билет негоден. Серверы приложений должны отвергать билеты с этим флагом. Обычно флагом INVALID помечаются предварительные билеты, предназначенные для использования в будущем. Чтобы предварительный билет стал годным, его нужно зарегистрировать, направив в TGS запрос с опцией VALIDATE. Регистрация пройдет успешно только после наступления начального времени годности билета. Процедура регистрации позволяет выявлять (и отвергать) украденные предварительные билеты.

4.3. Обновляемые билеты

Билеты с длительным сроком годности удобны, но небезопасны, поскольку кража содержащейся в них информации открывает злоумышленнику доступ к соответствующим серверам также на длительное время. С другой стороны, использование краткосрочных билетов заставляет постоянно обращаться к секретному ключу, что представляет еще большую опасность.

Обновляемые билеты позволяют смягчить последствия кражи информации. Они имеют два срока годности — промежуточный и окончательный. Клиент должен периодически, до окончания промежуточного срока, представлять обновляемый билет в TGS с опцией RENEW. В ответ TGS возвращает новый билет с другим сеансовым ключом и более поздним промежуточным сроком годнос-

ти. Все остальные поля переносятся из старого билета без изменений. Подобная периодическая перерегистрация в принципе позволяет TGS выявлять украденные билеты и отвергать их. В результате потенциальное время жизни нелегальных билетов уменьшается.

Обычно флаг RENEWABLE интерпретируется только TGS, однако некоторые особенно осторожные серверы могут отвергать билеты с этим флагом.

Флаг RENEWABLE устанавливается в билете лишь по явному запросу клиента к серверу начальной аутентификации (AS).

4.4. Предварительные билеты

Иногда нужны билеты, которые будут использоваться спустя довольно долгое время после выдачи. Например, от момента порождения пакетного задания до начала выполнения может проходить несколько часов. Поскольку держать годные билеты в пакетной очереди рискованно, Kerberos предлагает механизм предварительных билетов, получаемых в момент порождения задания, но "дремлющих" до момента активации и перерегистрации в TGS.

Флаг MAY-POSTDATE обычно интерпретируется только TGS в "билетах на билеты" и устанавливается лишь по явному запросу клиента к серверу начальной аутентификации (AS). По "билету на билеты" с флагом MAY-POSTDATE TGS выдает билет (к серверу) с флагом POSTDATED. Сервер может проверить, как давно был выдан предварительный билет и, возможно, отказать клиенту в обслуживании. Более того, билет с флагом POSTDATED первоначально содержит и флаг INVALID, так что клиент перед использованием должен перерегистрировать билет в TGS.

4.5. Доверенности

Когда возникает необходимость позволить серверу выполнить некоторые действия от имени клиента (например, получить доступ к файлам для их печати), можно воспользоваться имеющимся в Kerberos механизмом доверенностей.

Флаг PROXIABLE, установленный в "билете на билеты" (это происходит по умолчанию), дает право TGS выдать новый билет (но не "билет на билеты") с другим сетевым адресом. В новом билете устанавливается флаг PROXY. Клиент может передать серверу новый билет в качестве доверенности.

Чтобы затруднить использование украденных удостоверений, Kerberos-билеты обычно разрешается использовать только субъектам с сетевыми адресами, явно указанными в билете. В принципе сетевые адреса в билете можно не указывать, но это не рекомендуется. Вот почему клиент должен передать серверу доверенность — билет с его (сервера) сетевым адресом.

4.6. Билеты с правом передачи

Билеты с правом передачи позволяют развить механизм доверенностей и передавать серверу права на совершение любых действий от имени клиента. Подобное сильнодействующее средство полезно, например, когда пользователь, войдя в удаленную систему, хочет, чтобы проверка подлинности проходила так же, как и в случае локального входа.

Флаг FORWARDABLE трактуется почти так же, как PROXIABLE, однако дает право на получение новых "билетов на билеты" с другими сетевыми адресами и устанавливается лишь по явному запросу пользователя к серверу начальной аутентификации (AS) при получении первоначального "билета на билеты".

В принципе пользователь может получить новый "билет на билеты" с другими сетевыми адресами и без флага FORWARDABLE, обратившись непосредственно к AS, но тогда ему придется вводить свой пароль.

Флаг FORWARDED устанавливается в новом билете по явному запросу пользователя, когда он представляет TGS билет с флагом FORWARDABLE, задает опцию FORWARDED и указывает новый список сетевых адресов. Флаг FORWARDED наследуется при выдаче новых билетов.

5. Форматы данных в системе Kerberos

Опишем форматы двух важнейших объектов системы Kerberos — билета и аутентификатора. Описание дается в нотации ASN.1. Понимание этих форматов позволит лучше почувствовать тонкости механизма аутентификации. Листинг 1 содержит описание структуры билета, а листинг 2 — описание шифруемой (секретным ключом сервера) части билета.

Поясним смысл компонентов билета.

- tkt-vno — номер версии формата билета. Данный документ описывает версию 5.
- realm — область управления, выдавшая билет, и, одновременно, область, содержащая сервер.
- sname — имя сервера, к которому выдан данный билет.
- enc-part — зашифрованная часть билета.
- flags — битовая шкала флагов.
- key — сеансовый ключ, выданный Kerberos для связи между клиентом и сервером.
- crealm — область управления, в которой зарегистрирован клиент, и, одновременно, область, в которой происходила начальная проверка подлинности.
- sname — имя клиента.
- transited — список промежуточных областей управления, участвовавших в аутентификации клиента.
- authtime — время выдачи первоначального билета, "наследником" которого является данный билет. Осторожные серверы могут отвергать билеты, основанные на слишком давней аутентификации.
- starttime — время, отмечающее начало срока годности

билета. Если данное поле не задано, используется значение authtime.

- endtime — время, отмечающее окончание срока годности билета. В принципе конкретные серверы могут накладывать свои, более жесткие, ограничения на время жизни билета.
- renew-till — время, после которого даже обновляемый билет становится негодным.
- caddr — список сетевых адресов, откуда билет может быть использован. Если список пуст, билет можно использовать откуда угодно. Данный список затрудняет использование злоумышленником украденного билета.
- authorization-data — авторизационная информация, передаваемая владельцем билета серверу приложений. Предполагается, что это поле содержит имена обслуживаемых объектов и права доступа к ним. Поле authorization-data позволяет выдать доверенность на совершение определенных действий. Например, клиент, желающий напечатать файл, может передать серверу печати доверенность на чтение файла. Содержательная трактовка поля авторизации находится вне компетенции Kerberos; последний лишь переносит туда информацию, содержащуюся в запросе

```
Ticket ::= [APPLICATION 1] SEQUENCE {
    tkt-vno [0]      INTEGER,
    realm [1]       Realm,
    sname [2]       PrincipalName,
    enc-part [3]    EncryptedData
}
```

Листинг 1.

```
EncTicketPart ::= [APPLICATION 3] SEQUENCE {
    flags [0]       TicketFlags,
    key [1]         EncryptionKey,
    crealm [2]      Realm,
    sname [3]       PrincipalName,
    transited [4]   TransitedEncoding,
    authtime [5]    KerberosTime,
    starttime [6]   KerberosTime OPTIONAL,
    endtime [7]     KerberosTime,
    renew-till [8]  KerberosTime OPTIONAL,
    caddr [9]       HostAddresses OPTIONAL,
    authorization-data [10] AuthorizationData OPTIONAL
}
```

Листинг 2.

клиента на билет к серверу приложений.

Аутентификатор — это запись, посылаемая серверу вместе с билетом, удостоверяющая знание клиентом ключа шифрования, содержащегося в билете, помогающая серверу обнаружить воспроизведение (дублирование) информации и позволяющая выбрать "настоящий сеансовый ключ" для обслуживания конкретного сеанса связи между клиентом и сервером. Аутентификатор шифруется сеансовым ключом, копия которого находится в билете (листинг 3).

Комментарии к листингу:

- authenticator-vno — номер версии формата аутентификатора (5).
- crealm и cname — то же, что и для билета.
- cksum — контрольная сумма прикладных данных, передаваемых серверу.
- cusec — микросекундная часть временного штампа клиента.
- ctime — основная часть временного штампа клиента.
- subkey — выбранный клиентом ключ для защиты конкретного сеанса связи. Если это поле не задано, будет использоваться сеансовый ключ.
- seq-number — начальный номер последовательности сообщений. Механизм номеров используется для контроля целостности последовательности и, в частности, для защиты от дублирования. Начальный номер рекомендует-

ся выбирать случайным образом, а последующие номера не использовать дважды даже в разных сеансах.

- authorization-data — то же, что и для билета (позволяет уточнить информацию, переданную в билете).

6. Типы пересылаемых сообщений

6.1. Обмен с сервером начальной аутентификации

Общение с сервером начальной аутентификации обычно инициируется клиентом, желающим получить удостоверение к определенному серверу, но не имеющим пока других удостоверений. Для шифровки и расшифровки используется секретный ключ клиента. Как правило, данный обмен происходит при входе в систему, для получения удостоверения к серверу выдачи билетов. Кроме того, общение с сервером начальной аутентификации используется для получения удостоверения к серверам, требующим знания именно секретного ключа (пример — сервер смены пароля).

В своем запросе (KRB_AS_REQ) клиент посылает открытым текстом свое имя и имя сервера, к которому он хочет получить удостоверение. Ответ, KRB_AS_REP, содержит билет, который клиент должен будет представить серверу, и сеансовый ключ для совместного использования клиентом и сервером. Билет шифруется секрет-

ным ключом сервера, сеансовый ключ и дополнительная информация — секретным ключом клиента. Сообщение KRB_AS_REP содержит данные, позволяющие связать его с предыдущим запросом (KRB_AS_REQ) и обнаружить дублирование сообщений. В случае какой-либо ошибки возвращается сообщение KRB_ERROR, которое не шифруется.

Сервер аутентификации не делает попыток убедиться в подлинности обратившегося к нему субъекта. Просто он возвращает информацию, воспользоваться которой может лишь тот, кто знает секретный ключ субъекта.

6.2. Обмен с сервером выдачи билетов

Обмен между клиентом и сервером выдачи билетов инициируется клиентом, когда тот хочет получить удостоверение к определенному серверу (возможно, зарегистрированному в удаленной области управления), обновить или зарегистрировать существующий билет или получить билет с доверенностью. В первом случае клиент должен располагать предварительно полученным от сервера начальной аутентификации билетом к TGS. Формат сообщений при обмене с сервером выдачи билетов почти тот же, что и для сервера начальной аутентификации. Основное отличие состоит в том, что для шифровки и расшифровки используется сеансовый ключ.

```
Authenticator ::= [APPLICATION 2] SEQUENCE {
    authenticator-vno [0]      INTEGER,
    crealm [1]                 Realm,
    cname [2]                  PrincipalName,
    cksum [3]                   Checksum OPTIONAL,
    cusec [4]                   INTEGER,
    ctime [5]                   KerberosTime,
    subkey [6]                  EncryptionKey OPTIONAL,
    seq-number [7]              INTEGER OPTIONAL,
    authorization-data [8]      AuthorizationData OPTIONAL
}
```

Листинг 3.

Запрос (KRB_TGS_REQ) состоит из информации, подтверждающей подлинность клиента, и заявки на удостоверение. Ответ (KRB_TGS_REP) содержит запрашиваемое удостоверение, зашифрованное сеансовым ключом, и данные, позволяющие обнаружить дублирование сообщений и связать ответ с запросом.

6.3. Аутентификационный обмен клиент/сервер

Данный обмен используется сетевыми приложениями для взаимной проверки подлинности. Клиент должен располагать предварительно полученным удостоверением к серверу. Клиент передает серверу билет, аутентификатор (зашифрованный сеансовым ключом) и некоторую дополнительную учетную информацию. Сервер в ответ (если этот ответ требуется) возвращает только аутентификатор, зашифрованный тем же сеансовым ключом.

6.4. Защищенные сообщения

Сообщения типа KRB_SAFE могут использоваться партнерами по общению, желающими контролировать целостность передаваемой информации. Контроль целостности достигается за счет включения в сообщение криптографической контрольной суммы. Используется последний субсеансовый или сеансовый ключ.

6.5. Конфиденциальные сообщения

Сообщения типа KRB_PRIV могут использоваться партнерами по общению, желающими защитить данные от нелегального прочтения и контролировать их целостность. Цель достигается криптографическими методами.

6.6. Пересылка удостоверений

Сообщения типа KRB_CRED используются для

Направление сообщения	Тип сообщения
Клиент → Kerberos	KRB_AS_REQ
Kerberos → клиент	KRB_AS_REP или KRB_ERROR

Табл. 1. Обмен с сервером начальной аутентификации.

Направление сообщения	Тип сообщения
Клиент → Kerberos	KRB_TGS_REQ
Kerberos → клиент	KRB_TGS_REP или KRB_ERROR

Табл. 2. Обмен с сервером выдачи билетов.

Направление сообщения	Тип сообщения
Клиент → сервер приложений	KRB_AP_REQ
Сервер приложений → клиент (может отсутствовать)	KRB_AP_REP или KRB_ERROR

Табл. 3. Аутентификационный обмен клиент/сервер.

Поле	Значение
name	Идентификатор субъекта
key	Секретный ключ субъекта
p_kvno	Номер версии ключа субъекта
max_life	Максимальный срок годности билетов
max_renewable_life	Максимальный суммарный срок годности обновляемых билетов

Табл. 4. Обязательные поля базы данных Kerberos.

пересылки удостоверений с одного хоста на другой. В сообщении входят билет и зашифрованные данные, содержащие сеансовый ключ и другую ассоциированную с билетом информацию.

7. База данных Kerberos

Сервер Kerberos должен иметь доступ к базе данных, содержащей идентификаторы и

секретные ключи субъектов. Записи в этой базе должны содержать по крайней мере информацию, показанную в табл. 4.

Секретные ключи субъектов могут шифроваться "мастер-ключом" Kerberos.

Когда секретный ключ сервера изменяется нормальным образом (то есть не в результате компрометации старого ключа),

Поле	Значение
K_kvno	Номер версии ключа Kerberos
expiration	Время окончания срока годности записи
attributes	Битовая шкала атрибутов
mod_date	Время последней модификации
mod_name	Идентификатор субъекта, выполнившего модификацию

Табл. 5. Дополнительные поля базы данных Kerberos.

старый ключ следует сохранять до тех пор, пока остаются годными билеты, выданные с его помощью. Таким образом, возможна ситуация, когда один субъект имеет несколько активных ключей. Информация, шифруемая секретным ключом, всегда помечается версией примененного ключа, иначе правильная расшифровка может стать невозможной.

Субъекту с несколькими активными ключами соответствует несколько записей в базе данных Kerberos. При выдаче билетов и в процессе начальной аутентификации всегда используется самый свежий ключ (с максимальным номером версии).

В реализации Kerberos в рамках проекта Афина записи базы данных содержали дополнительные поля, показанные в табл. 5.

Битовая шкала атрибутов может задавать характеристики билетов, допустимые для данного субъекта, или влиять на процесс преобразования пароля в секретный ключ.

Если политика безопасности, проводимая организацией, требует более детальной отчетности, в записи базы данных Kerberos могут быть включены поля, помогающие отследить все операции с билетами. Правда, при этом следует учитывать необходимость частых изменений базы, что может быть нежелательным и по соображениям эффективности, и по соображе-

ниям безопасности. Также возможно, но нежелательно использование внешней по отношению к Kerberos базы данных (например, службы имен сетевой операционной системы).

8. Предлагаемые направления развития системы Kerberos

Популярность системы Kerberos довольно быстро растет, а вместе с ней растет и число предложений, направленных на развитие Kerberos. Эти предложения можно разбить на две основные группы:

- стандартизация программного интерфейса (как часть процесса выработки более общего прикладного программного интерфейса безопасности, затрагивающего не только аутентификацию)
- стандартизация использования в процедуре начальной аутентификации одноразовых паролей и/или асимметричных методов шифрования

Можно надеяться, что доступность системы Kerberos и гласность процедуры стандартизации сделают аутентификационный сервис еще более надежным и гибким.

Приложение 1. Терминологический словарь

Авторизация — процесс определения правомочности использования клиентом определенной услуги, перечня до-

ступных объектов, а также разрешенных видов доступа.

Аутентификатор — запись, содержащая информацию, про которую можно доказать, что она была недавно сгенерирована с использованием сеансового ключа, известного только клиенту и серверу.

Аутентификационный заголовок — запись, содержащая билет и удостоверение. Эта запись будет представлена серверу в процессе проверки подлинности.

Аутентификационный маршрут — последовательность промежуточных областей управления, пересекаемых в процессе проверки подлинности субъектов из разных областей.

Аутентификация (проверка подлинности) — проверка того, что субъект является тем, за кого он себя выдает.

Билет — запись, помогающая клиенту доказать серверу свою подлинность. Билет содержит сведения о клиенте, сеансовый ключ, временной штамп и другую информацию. На билете ставится печать с помощью секретного ключа сервера. Билет действителен только вместе со свежим аутентификатором.

Билет на билеты (Ticket-Granting Ticket, TGT) — билет к серверу выдачи билетов (Ticket-Granting Server, TGS), позволяющий получать билеты к другим серверам.



Доверенность — документ, дающий предъявителю право на доступ к определенным объектам или услугам. В системе Kerberos это может быть билет, использование которого ограничено содержимым поля авторизации. К билету должен прилагаться сеансовый ключ.

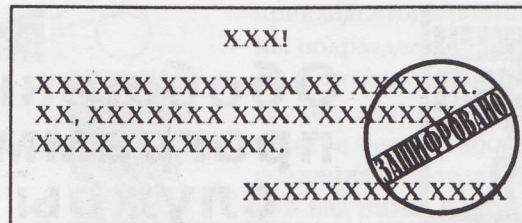
Идентификатор субъекта — имя, используемое для уникальной идентификации субъекта.

Клиент — процесс, использующий сетевые сервисы от имени пользователя. Отметим, что в некоторых случаях сервер сам может являться клиентом некоторого другого сервера (например, сервер печати может пользоваться услугами файлового сервера).

Обычный текст — исходные данные для функции шифрования или результат функции расшифровки. Расшифровка преобразует зашифрованный текст в обычный.

Печать — средство, позволяющее так зашифровать запись, содержащую несколько полей, что замена отдельных полей невозможна без знания ключа шифрования (иначе подмена будет обнаружена).

Сеансовый ключ — временный ключ шифрования, используемый двумя субъектами. Время жизни такого ключа ограничено одним сеансом работы в системе.



Секретный ключ — ключ шифрования, разделяемый субъектом и KDC и используемый длительное время. В случае, когда субъектом является пользователь, секретный ключ вычисляется по паролю.

Сервер — субъект, предоставляющий ресурсы сетевым клиентам.

Сервис (услуга) — ресурс, предоставляемый сетевым клиентам. Зачастую может поддерживаться несколькими серверами (например, сервис удаленных файлов).

Субсеансовый ключ — временный ключ шифрования, используемый двумя субъектами. Обмен субсеансовыми ключами защищается посредством сеансовых ключей. Время жизни субсеансового ключа ограничено одним сеансом общения между субъектами.

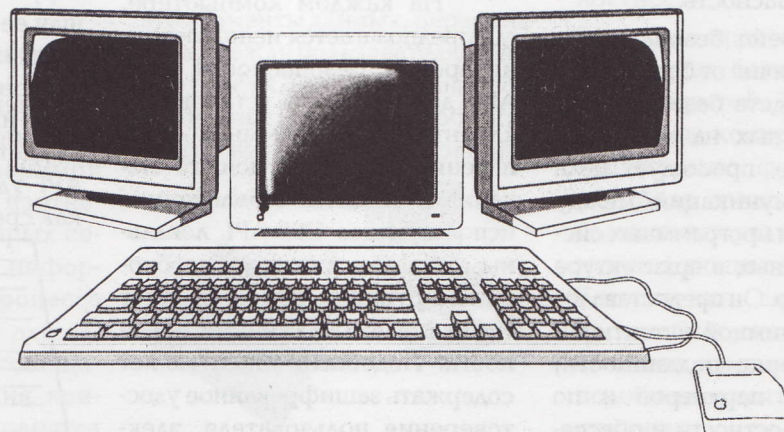
Субъект — пользователь, клиент или сервер, имеющий уникальное имя и участвующий в сетевом общении.

Удостоверение — совокупность билета и секретного сеансового ключа, необходимого для успешного использования билета в процессе проверки подлинности.

Центр распределения билетов (Key Distribution Center, KDC) — сетевой сервис, предоставляющий билеты и временные сеансовые ключи, или конкретный сервер, оказывающий данную услугу, или хост, на котором функционирует этот сервер. KDC обслуживает запросы и на начальный билет (соответствующий компонент иногда называют сервером начальной аутентификации — Authentication Server, AS), и на билеты к конкретным серверам (этот компонент именуют сервером выдачи билетов — Ticket-Granting Server, TGS).

Цербер (Kerberos) — не только мифологический трехголовый пес, охранявший подземное царство, но и название сервиса проверки подлинности для проекта Афина (Массачусетский технологический институт), протокол, используемый этим сервисом, а также программная реализация сервиса аутентификации.

Зашифрованный текст — результат функции шифрования. Шифрование преобразует обычный текст в зашифрованный.





Обобщенный прикладной программный интерфейс службы безопасности

Содержание

1. Введение	18
2. Основные понятия	18
3. Классификация функций безопасности	20
4. Логика работы пользователей интерфейса безопасности	23
5. Службы безопасности, которые могут предоставить интерфейс GSS-API	23
6. Детальное описание некоторых функций безопасности	25
7. Ограничения интерфейса безопасности	30
8. Представление некоторых объектов интерфейса безопасности в среде языка C	31
9. Заключение	32
Литература	32

1. Введение

Обобщенный прикладной программный интерфейс службы безопасности (Generic Security Service Application Program Interface – GSS-API) – это набор спецификаций, одобренных сообществом Internet и распространяющих идеологию открытых систем на такую традиционно закрытую область, как информационная безопасность.

Интерфейс безопасности GSS-API, в отличие от большинства других средств безопасности, ориентированных на локальное использование, преследует цель защиты коммуникаций между компонентами программных систем, построенных в архитектуре клиент/сервер. Он предоставляет услуги по взаимной аутентификации (проверке подлинности) общающихся партнеров и по контролю целостности и обеспе-

чению конфиденциальности пересылаемых сообщений. Пользователями интерфейса безопасности GSS-API являются коммуникационные протоколы (обычно прикладного уровня) или другие программные системы, самостоятельно выполняющие пересылку данных.

Обобщенный интерфейс безопасности GSS-API не зависит от конкретной языковой среды и от механизма безопасности, обеспечивающего реальную защиту. Последнее обстоятельство позволяет создавать приложения, которые на уровне исходного текста мобильны по отношению к смене механизма безопасности. Тем самым реализуется открытость прикладных систем и соответствующих средств защиты. Следует отметить, что средства защиты могут быть существенно разными – от систем Kerberos, в основе которых лежат симметричные методы шифрования, и до продуктов, реализующих спецификации X.509, где явно оговаривается использование открытых ключей. Попутно мы видим, что обобщенный интерфейс безопасности GSS-API имеет под собой вполне конкретную почву.

На каждом компьютере, где предполагается использовать интерфейс безопасности GSS-API, должно быть установлено клиентское программное обеспечение соответствующего механизма защиты. Приложение, использующее GSS-API, локальным образом вызывает необходимые функции, получая в ответ так называемые токены безопасности. Подобный токен может содержать зашифрованное удостоверение пользователя, элек-

тронную подпись под сообщением или целое зашифрованное сообщение. Приложения обмениваются токенами безопасности, достигая тем самым аутентификации, целостности и конфиденциальности общения. Поскольку коммуникационные аспекты вынесены за пределы обобщенного интерфейса безопасности GSS-API, он автоматически оказывается независимым от сетевых протоколов. Сетевая мобильность приложений должна обеспечиваться иными средствами.

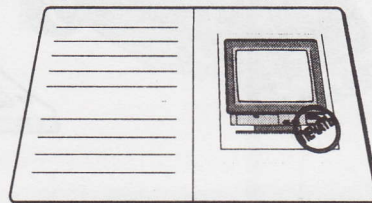
Данная работа основана на второй версии интерфейса GSS-API, описанной в документе [1]. Первая версия зафиксирована в RFC 1508 [2]. Отображение обобщенного интерфейса в языковую среду C – тема спецификации RFC 1509 [3]. Система Kerberos (версия 5) описана в RFC 1510 [4]. С рекомендациями X.509 можно ознакомиться по публикации [5].

2. Основные понятия

В этом разделе будут рассмотрены базовые элементы обобщенного интерфейса безопасности GSS-API.

2.1. Удостоверение

Удостоверение – это структура данных, позволяющая ее владельцу доказать партнеру по общению или третьей стороне, что он (владелец) является именно тем, за кого себя выдает. Таким образом, в GSS-API удостоверения выступают как средство аутентификации.



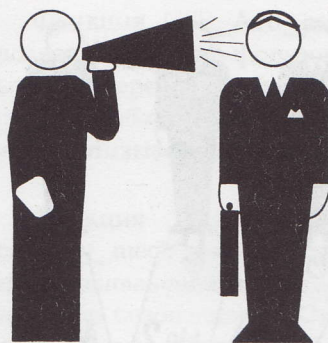
Естественно, что служба безопасности (например, Kerberos) совместно с операционной системой должны обеспечить защиту удостоверений от несанкционированного использования и/или изменения. Процессам, действующим от имени пользователей, не предоставляется прямого доступа к удостоверениям. Вместо этого, при необходимости процессы снабжаются дескрипторами удостоверений. Дескрипторы не содержат секретной информации и не нуждаются в защите. Нет смысла воровать дескрипторы, поскольку их интерпретация для разных процессов-предъявителей будет различной.

Вообще говоря, одному пользователю могут понадобиться удостоверения нескольких видов. Во-первых, структура удостоверений, несомненно, зависит от механизма безопасности, стоящего за обобщенным интерфейсом. Во-вторых, можно представить себе ситуацию, когда для общения с разными партнерами нужны различные удостоверения. В-третьих, особым образом должны быть устроены так называемые делегируемые удостоверения, служащие для передачи прав на выполнение определенных действий от имени некоторого пользователя. Есть и другие факторы, влияющие на структуру удостоверений.

В процессе входа пользователя в систему могут формироваться удостоверения стандартного вида, выдаваемые по умолчанию.

2.2. Контекст безопасности

Контекст безопасности — это пара структур данных (по одной локально хранимой структуре для каждого партнера по общению), в которых содержится разделяемая информация о состоянии процесса общения, необходимая для защиты пересылаемых сообщений. Как и удостоверения, контексты безопасности хранятся



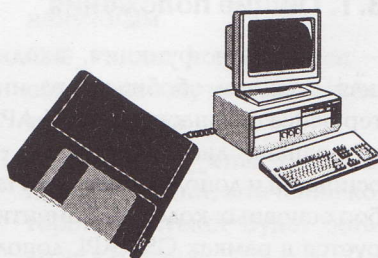
внутренним для службы безопасности образом — прикладные процессы снабжаются лишь дескрипторами контекстов. Контексты формируются на основании локально выданных или делегированных удостоверений.

Партнеры по общению могут по очереди или одновременно использовать несколько контекстов, если необходимо поддерживать информационные потоки разной степени защищенности.

Интерфейс GSS-API не зависит от используемого сетевого протокола. По этой причине формирование контекста безопасности никак не связано с установлением соединения в сетевом смысле. Более того, для GSS-API безразлично, используется ли протокол с установлением соединения или без такового. Организация потока сообщений, а также выделение из входного потока данных, генерируемых в рамках GSS-API, — обязанность прикладных систем.

2.3. Токены безопасности

Токены безопасности — это элементы данных, пересылаемые между пользователями интерфейса GSS-API с целью поддержания работоспособности этого интерфейса и защиты



прикладной информации. Токены подразделяются на два класса. Контекстный класс предназначен для установления контекстов безопасности и для выполнения управляющих действий над ними. В рамках уже установленного контекста для защиты сообщений используются токены сообщений.

Когда приложение хочет начать общение с удаленным партнером, оно обращается к интерфейсу GSS-API с просьбой инициализировать контекст. В ответ возвращается контекстный токен безопасности, который приложение обязано переслать партнеру. Тот, получив токен, передает его локальному экземпляру GSS-API для проверки подлинности инициатора и продолжения (обычно — завершения) формирования контекста.

Если приложению необходимо защитить сообщение, обеспечив возможность контроля целостности и подлинности источника данных, оно передает сообщение интерфейсу GSS-API, получая в ответ токен, содержащий криптографическую контрольную сумму (имитовставку, хэш, дайджест), заверенную электронной подписью отправителя. После этого приложение должно переслать как само сообщение, так и защищающий его токен. Партнер, получив обе порции данных, передает их своему локальному экземпляру GSS-API для проверки авторства и целостности.

Для приложения структура токенов безопасности является закрытой. Токены генерируются и контролируются исключительно функциями GSS-API. Дело приложения — переслать их и передать соответствующим функциям для обработки. Естественно, служба безопасности обнаружит попытки приложения изменить токен, если подобные попытки будут приняты.

2.4. Типы механизмов безопасности, имена и привязка к каналам

Вполне возможно, что на удаленных системах функционирует несколько различных служб безопасности. В этих условиях для успешного формирования контекста безопасности партнеры по общению должны тем или иным способом договориться о том, какая именно служба будет использоваться.

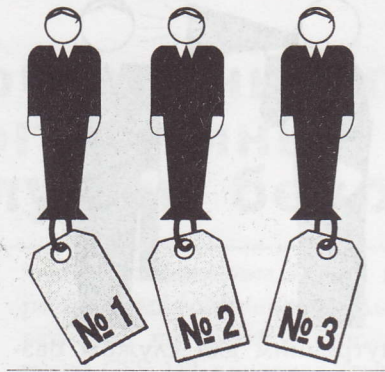
Конкретная служба характеризуется типом реализуемого механизма безопасности. В свою очередь, тип обозначается посредством структуры данных, называемой идентификатором объекта. На уровне обобщенного программного интерфейса структура идентификаторов объектов не уточняется.

Каждая система обязана предлагать некоторый механизм безопасности как подразумеваемый, которым приложению и рекомендуется пользоваться из соображений мобильности.

Если токены являются структурой, закрытой для приложений, то структура имен, употребляемых при формировании контекста безопасности (имеются в виду имена партнеров по общению), закрыта для функций GSS-API. Имена рассматриваются этими функциями просто как последовательности байт, интерпретируемые, вероятно, коммуникационными компонентами приложений.

В GSS-API предусматривается наличие трех типов имен — внутренних, печатных и объектных. Как правило, аргументами функций GSS-API служат внутренние имена. Интерфейс GSS-API предоставляет функции для преобразования имен и выполнения некоторых других действий над ними.

Отметим, что интерпретация имен — дело довольно сложное, поскольку они могут, вооб-



ще говоря, принадлежать разным пространствам имен. Чтобы избежать неоднозначности, в состав имени включается идентификатор его типа.

Чтобы усилить защиту информации, в интерфейсе GSS-API предлагается возможность связывания контекста безопасности с определенными каналами передачи данных. Более точно, инициатор формирования контекста может указать набор каналов, которые допустимо использовать в рамках открываемого сеанса общения. Партнер должен подтвердить свое согласие на связывание с этим набором каналов. Канал характеризуется целевым адресом и некоторыми другими параметрами, такими как формат пересылаемой по нему информации, степень ее защищенности и т.п. Если так случится, что токен, отправленный для установления контекста, будет перехвачен, использование соответствующего контекста ограничится рамками связанных с ним каналов. Это, как можно надеяться, затруднит действия злоумышленника.

3. Классификация функций безопасности

3.1. Общие положения

Каждая функция, входящая в состав обобщенного интерфейса безопасности GSS-API, возвращает два кода ответа — основной и дополнительный. Набор основных кодов регламентируется в рамках GSS-API, допол-

нительные коды могут быть специфичны для различных служб безопасности.

Основные коды подразделяются на информирующие и сигнализирующие об ошибке. Перечислим некоторые информирующие коды:

- GSS_S_COMPLETE — нормальное завершение
- GSS_S_CONTINUE_NEEDED — требуется дополнительный вызов данной функции
- GSS_S_DUPLICATE_TOKEN — обнаружено дублирование токена защиты сообщений

Следующие значения входят в число кодов, сигнализирующих об ошибке:

- GSS_S_BAD_NAME — задано некорректное имя
- GSS_S_CONTEXT_EXPIRED — истек срок действия контекста
- GSS_S_CREDENTIALS_EXPIRED — истек срок действия удостоверения
- GSS_S_DEFECTIVE_TOKEN — обнаружено повреждение токена безопасности
- GSS_S_NO_CONTEXT — не задан контекст
- GSS_S_NO_CRED — не задано удостоверение

Проанализировав основной код, приложение может сделать вывод о нормальном или ненормальном завершении функции. В последнем случае оно может принять во внимание дополнительный код, что даст пользователю больше информации, но, вообще говоря, уменьшит мобильность приложения.

3.2. Работа с удостоверениями

Работа приложения, опирающегося на интерфейс GSS-API, должна начинаться с получения удостоверения, которое засвидетельствует, что приложение имеет право выступать от имени определенного субъекта.

Интерфейс GSS-API предоставляет следующие функции для работы с удостоверениями:

- `GSS_Acquire_cred` — запрос удостоверения для последующего использования
- `GSS_Release_cred` — отказ от удостоверения, ставшего ненужным
- `GSS_Inquire_cred` — получение информации об удостоверении
- `GSS_Add_cred` — постепенное формирование удостоверения
- `GSS_Inquire_cred_by_mech` — получение информации об удостоверении, связанной с конкретным механизмом безопасности

Как уже указывалось, приложение получает в свое распоряжение не само удостоверение, а его дескриптор. Этот дескриптор используется затем в качестве аргументов функций `GSS_Release_cred`, `GSS_Inquire_cred`, `GSS_Add_cred`, а также функций формирования контекста безопасности.

Функция `GSS_Acquire_cred` нужна в первую очередь серверным процессам, желающим явно указать, от чьего имени они выступают. Интерактивные пользователи могут неявным образом получать подразумеваемые удостоверения при входе в систему. Аналогично, при выходе такого пользователя из системы может автоматически выполняться вызов функции `GSS_Release_cred`.

Функция `GSS_Inquire_cred` позволяет получить информацию об удостоверении — имя ассоциированного субъекта, срок годности, применимость для операций с контекстом (инициация и/или принятие), поддерживаемые механизмы безопасности. Данная функция особенно полезна применительно к подразумеваемому удостоверению, характеристики которого могут меняться от системы к системе.

Функция `GSS_Add_cred` позволяет постепенно формировать удостоверения, добавляя в них поля, необходимые различным механизмам безопасности.

Функция `GSS_Inquire_cred_by_mech` полезна в среде, поддерживающей несколько механизмов безопасности. Она, в частности, позволяет убедиться в том, что данное удостоверение допускает использование совместно с конкретным механизмом безопасности.

3.3. Создание и уничтожение контекста безопасности

Создание контекста безопасности предшествует всем операциям по обмену сообщениями между потенциальными партнерами. В процессе формирования контекста партнеры могут убедиться в подлинности друг друга, а также договориться о степени защищенности коммуникаций. Впрочем, обычно, в силу асимметричности отношения клиент/сервер, аутентификация носит односторонний характер — сервер убеждается в подлинности клиента.

Для работы с контекстами обобщенный интерфейс безопасности GSS-API предоставляет следующие функции:

- `GSS_Init_sec_context` — формирование "исходящего" контекста, то есть части контекста, относящейся к инициатору общения
- `GSS_Accept_sec_context` — формирование "входящего" контекста, то есть части контекста, относящейся к вызываемому партнеру
- `GSS_Delete_sec_context` — отказ от контекста, ставшего ненужным
- `GSS_Process_context_token` — обработка полученного контекстного токена
- `GSS_Context_time` — выяснение времени, в течение которого контекст будет оставаться годным

- `GSS_Inquire_context` — получение информации о контексте
- `GSS_Wrap_size_limit` — выяснение максимального размера сообщения, которое можно зашифровать в рамках заданного контекста безопасности
- `GSS_Export_sec_context` — передача контекста другому процессу
- `GSS_Import_sec_context` — прием контекста, переданного другим процессом

Инициатор общения вызывает функцию `GSS_Init_sec_context`, передавая ей свое удостоверение — явно полученное или подразумеваемое. Кроме того, инициатор специфицирует запрашиваемую процедуру взаимной аутентификации и уровень защиты сообщений. В ответ ему возвращается токен, который следует переслать партнеру. Партнер, получив токен, передает его функции `GSS_Accept_sec_context` (вместе со своим удостоверением). Как правило, на этом формирование контекста заканчивается. Партнеры должны проанализировать флаги, ассоциированные с контекстом, чтобы узнать, каков реально предоставленный уровень защиты.

Если инициатор общения (обычно это клиент) желает убедиться в подлинности партнера, он передает функции `GSS_Init_sec_context` флаг `mutual_req_flag` (требуется взаимная аутентификация). В таком случае вызов `GSS_Init_sec_context` возвращает в качестве основного кода значение `GSS_S_CONTINUE_NEEDED` (а не `GSS_S_COMPLETE`). Соответствующим образом меняется и генерируемый контекстный токен. Партнер, приняв и обработав (с помощью функции `GSS_Accept_sec_context`) присланную информацию, получит на выходе установленный флаг `mutual_state` и новый контекстный токен, который следует вернуть инициатору. Последний должен

повторно обратиться к функции `GSS_Init_sec_context` с полученным токеном. При отсутствии ошибок `GSS_Init_sec_context` вернет, наконец, основной код `GSS_S_COMPLETE`, и формирование контекста на этом завершится.

В принципе служба безопасности может требовать при формировании контекста обмена несколькими токенами (даже без взаимной аутентификации). В этом случае инициатору придется несколько раз вызывать функцию `GSS_Init_sec_context`, а его партнеру — функцию `GSS_Accept_sec_context`. Все вызовы, кроме последнего, вернут основной код `GSS_S_CONTINUE_NEEDED`. Последний вызов, завершающий формирование контекста на соответствующей стороне, в нормальном случае выдаст основной код `GSS_S_COMPLETE`.

Общающиеся партнеры должны сами определять (способами, внешними по отношению к GSS-API), что из пересылаемой информации представляет собой контекстный токен и какой функции его следует передать.

При отказе от контекста, ставшего ненужным, можно получить от функции `GSS_Delete_sec_context` управляющий токен, подлежащий пересылке партнеру. Партнер, приняв токен, должен вызвать функцию `GSS_Process_context_token`, которая проанализирует управляющую информацию и удалит вторую половину сбрасываемого контекста.

Как правило, в процессе формирования контекста партнер получает информацию о его инициаторе. Инициатор может, посредством флага `anon_req_flag`, запросить формирование "анонимного" контекста. Это имеет смысл при пользовании общедоступными услугами (получение свободно распространяемой

информации и т.п.). Партнер вправе принять или отвергнуть анонимный контекст.

Функция `GSS_Inquire_context` позволяет получить информацию о контексте — имена инициатора общения и его партнера, срок годности, тип задействованного механизма безопасности, а также ассоциированные флаги (`replay_det_state` — обеспечивается отслеживание продублированных сообщений, `conf_avail` — предоставляется возможность шифровать сообщения и т.п.).

Функция `GSS_Export_sec_context` позволяет получить токен, пригодный для передачи контекста безопасности другому процессу в пределах одной вычислительной системы. Передавать можно только полностью сформированный контекст. Вообще говоря, экспортировав контекст, процесс теряет право на его использование.

Для приема (импорта) контекста безопасности служит функция `GSS_Import_sec_context`.

3.4. Защита сообщений

В процессе формирования контекста партнеры по общению имеют возможность убедиться в подлинности друг друга. Все остальные средства службы безопасности направлены на защиту сообщений.

Интерфейс GSS-API предоставляет следующие функции для работы с сообщениями:

- `GSS_GetMIC` — формирование токена, позволяющего контролировать целостность сообщения и подлинность его источника
- `GSS_VerifyMIC` — проверка целостности сообщения и подлинности источника с помощью ассоциированного токена
- `GSS_Wrap` — формирование инкапсулированного, возможно, зашифрованного, сообщения, содержащего ин-

формацию для контроля целостности и проверки подлинности источника

- `GSS_Unwrap` — разбор инкапсулированного сообщения

При формировании контекста инициатор специфицирует требуемый уровень защиты сообщений. Ответные флаги показывают, обеспечивается ли этот уровень на самом деле. Флаг `integ_avail` информирует о возможности контроля целостности и подлинности источника сообщения, флаг `conf_avail` — о доступности средств шифрования. Прежде чем обращаться к функциям `GSS_GetMIC/GSS_Wrap`, приложение должно проверить состояние перечисленных флагов.

Функция `GSS_GetMIC` по сообщению формирует отдельный токен безопасности. Функция `GSS_Wrap` "упаковывает" контрольную информацию вместе с сообщением (быть может, зашифрованным). Приложения должны уметь различать токены безопасности и сообщения (инкапсулированные или нет) и обрабатывать их соответствующим образом.

Служба безопасности может предоставлять дополнительные услуги в виде отслеживания продублированных сообщений и (более сильного) контроля последовательности сообщений. При формировании контекста эти услуги могут быть заказаны (флаги `replay_det_req_flag` и `sequence_req_flag`). Ответные флаги показывают, действительно ли обеспечивается запрошенный уровень защиты. Если это так, то в ассоциированные токены безопасности или инкапсулированные сообщения прозрачным для приложения образом могут вставляться порядковые номера, временные штампы и т.п. Соответственно, приложение должно быть готово получить от функций `GSS_VerifyMIC/GSS_Unwrap` основные коды завершения

GSS_S_DUPLICATE_TOKEN (обнаружено дублирование сообщений), GSS_S_OLD_TOKEN (старое сообщение), GSS_S_UNSEQ_TOKEN (опоздавшее сообщение), GSS_S_GAP_TOKEN (сообщение пришло слишком рано — некоторые предшествующие сообщения еще не получены). Подозрительные сообщения, несмотря на ненормальный код завершения, передаются приложению, которое трактует ситуацию в соответствии с избранной политикой безопасности. В частности, ничто не мешает обработать сообщение обычным образом.

Некоторые службы безопасности могут по выбору предоставлять различное качество защиты (Quality Of Protection — QOP). Выбор подходящего качества важен для приложения с точки зрения разумного расходования ресурсов, поскольку сильная защита может требовать значительных накладных расходов. В спецификациях интерфейса GSS-API определяется формат элемента данных, описывающего качество защиты. Это 32-битное целое, состоящее из двух 16-битных частей, одна из которых относится к контролю целостности, а другая — к обеспечению конфиденциальности. В обоих случаях задается степень контроля, идентификаторы используемых алгоритмов и информация, специфичная для выбранного алгоритма.

3.5. Вспомогательные средства

В число вспомогательных входят следующие функции:

- GSS_Display_status — получение текста сообщения, ассоциированного с кодом завершения
- GSS_Indicate_mechs — получение списка типов механизмов безопасности, поддерживаемых локальной системой
- GSS_Compare_name —

сравнение имен на равенство

- GSS_Display_name — преобразование имени из внутреннего представления в печатное
- GSS_Import_name — преобразование имени из печатного представления во внутреннее
- GSS_Release_name — отказ от ставшего ненужным внутреннего имени
- GSS_Release_buffer — отказ от ставшего ненужным печатного имени
- GSS_Release_oid_set — отказ от ставшего ненужным набора идентификаторов объектов
- GSS_Import_name_object — преобразование именуемого объекта во внутреннее имя
- GSS_Export_name_object — преобразование внутреннего имени в именуемый объект

и некоторые другие. Мы не будем подробно описывать вспомогательные функции.

4. Логика работы пользователей интерфейса безопасности

На рис. 1 приведен примерный сценарий взаимодействия между клиентом и сервером. Предполагается, что для взаимной аутентификации достаточно обмена одной парой токенов безопасности и что партнеры запрашивают и контроль целостности, и обеспечение конфиденциальности сообщений, а служба безопасности предоставляет им эти средства.

Мы видим, что общая схема взаимодействия удаленных партнеров под защитой интерфейса безопасности GSS-API довольно проста, однако практическая реализация всех необходимых проверок (кодов завершения, установленных флагов) требует известной аккуратности.

5. Службы безопасности, которые могут предоставить интерфейс GSS-API

Обобщенный прикладной программный интерфейс службы безопасности GSS-API — это "всего лишь" спецификации, приобретающие реальную силу, только если за ними стоят конкретные системы. Мы рассмотрим две разновидности подобных систем — Kerberos и службы безопасности, основанные на рекомендациях X.509.

Данный раздел предназначен для тех, кто знаком с соответствующими службами.

5.1. Система Kerberos

Понятия системы Kerberos можно отображать на понятия обобщенного интерфейса безопасности GSS-API разными способами. Мы рассмотрим простейший из них.

Роль удостоверения выполняет билет на билеты (в терминологии Kerberos — Ticket Granting Ticket, TGT). Функция GSS_Acquire_cred выдаст дескриптор удостоверения.

Функция GSS_Init_sec_context пройдет маршрут от локальной области управления до удаленной, в которой располагается сервер, и получит билет к серверу. В контекст безопасности, помимо прочих данных, войдет сгенерированный сеансовый ключ. Функция GSS_Init_sec_context вернет в качестве токена безопасности сообщение, имеющее Kerberos-формат KRB_AP_REQ и использующееся при аутентификационном обмене клиент/сервер.

После получения сервером сообщения KRB_AP_REQ и его обработки функцией GSS_Accept_sec_context, обе стороны будут располагать общим сеансовым ключом, который они смогут использовать для защиты пересылаемых сообщений. Если требуется взаимная аутентификация,

КЛИЕНТ

СЕРВЕР

```
{Получение удостоверения клиентом}
{Неявное получение подразумеваемого
удостоверения при входе в систему}

{Начало формирования контекста клиентом}
GSS_Init_sec_context (NULL {подразумеваемое
удостоверение}, 0 {дескриптор контекста
пока не сформирован}, targ_name {имя
удаленного сервера}, ..., mutual_req_flag
{истина, если нужна взаимная аутентификация}, ...,
NULL {пока нет токена, полученного от партнера})
→
GSS_S_CONTINUE_NEEDED, ...,
client_context_handle {дескриптор контекста
для клиента}, ..., con_token_1 {контекстный токен,
который нужно переслать серверу}, ... {флаги}, ...
```

Пересылка токена con_token_1 серверу

```
{Завершение формирования контекста клиентом}
GSS_Init_sec_context (NULL, client_context_handle,
targ_name, ..., con_token_2)
→
GSS_S_COMPLETE, ..., client_context_handle, ...,
NULL {нет токена, который нужно переслать
серверу}, ...

{Формирование клиентом сообщения с контролем
целостности и доказательством подлинности
источника данных}
GSS_GetMIC (client_context_handle, 0 {подразумеваемое
качество защиты}, message_1 {защищаемое
сообщение})
→
GSS_S_COMPLETE, ...,
mes_token_1 {токен, защищающий сообщение}
```

Пересылка сообщения message_1 и токена mes_token_1 серверу

```
{Прием клиентом зашифрованного сообщения}
GSS_Unwrap (client_context_handle,
encaps_message_2)
→
conf_state {истина, если служба безопасности
обеспечивает конфиденциальность}, qop_state
{обеспечиваемое качество защиты},
GSS_S_COMPLETE {если все в порядке}, ...,
message_2 {расшифрованное сообщение}
```

```
{Обработка полученного управляющего токена}
GSS_Process_context_token (client_context_handle,
con_token_3)
→
GSS_S_COMPLETE, ...

{Выход клиента из системы}
```

```
{Получение удостоверения сервером}
GSS_Acquire_cred(server_name
{имя, под которым регистрируется сервер}, ...)
→
GSS_S_COMPLETE, ...,
serv_cred_handle {дескриптор удосто-
верения сервера},
...

{Обслуживание других клиентов}
...
```

```
{Принятие контекста сервером}
GSS_Accept_sec_context(serv_cred_handle,
0 {дескриптор контекста пока не сформи-
рован}, ..., con_token_1)
→
GSS_S_COMPLETE, ...,
client_name {имя инициатора формирова-
ния контекста}, ..., serv_context_handle {дескрип-
тор контекста для сервера}, ... {флаги}, ...,
con_token_2 {контекстный токен, который
нужно вернуть клиенту}
```

Пересылка токена con_token_2 клиенту

```
{Проверка сервером целостности и аутентичности
полученного сообщения}
GSS_VerifyMIC (serv_context_handle, message_1,
mes_token_1)
→
qop_state {обеспечиваемое качество защиты},
GSS_S_COMPLETE {если все в порядке},
...
```

```
{Формирование сервером зашифрованного сообщения}
GSS_Wrap (serv_context_handle, conf_req_flag {истина,
если требуется шифрование}, 0 {подразумеваемое
качество защиты}, message_2)
→
GSS_S_COMPLETE, ...,
conf_state {истина, если служба безопасности обеспе-
чивает конфиденциальность}, encaps_message_2
{инкапсулированное сообщение message_2, зашиф-
рованное, с добавленной информацией для контроля
целостности и подлинности источника}
```

Пересылка зашифрованного сообщения encaps_message_2 клиенту

```
{Сброс сервером контекста безопасности,
ставшего ненужным}
GSS_Delete_sec_context(serv_context_handle)
→
GSS_S_COMPLETE, ..., con_token_3 {контекстный токен,
который можно переслать клиенту, чтобы тот также
удалил свою часть контекста безопасности}
```

Пересылка токена con_token_3 клиенту

{Обслуживание других клиентов}

{Обмен другими сообщениями}

Рис. 1. Сценарий взаимодействия между клиентом и сервером.

Детали, связанные с преобразованием имен, опущены. Текст в фигурных скобках является комментарием. Стрелка отделяет входные параметры от выходных.

сервер должен переслать клиенту Kerberos-сообщение KRP_AP_REP.

Естественным образом устанавливается соответствие между сроками годности удостоверений и контекстов с одной стороны, и сроками годности соответствующих билетов (обычных или предназначенных для получения других билетов) с другой стороны.

Выше было отмечено, что интерфейс GSS-API предусматривает возможность делегирования полномочий. Аналогичное средство имеется и в системе Kerberos (флаги PROXIABLE, PROXY, FORWARDABLE, FORWARDED).

5.2. Системы, основанные на рекомендациях X.509

Как известно, рекомендации X.509 предусматривают использование шифрования с открытыми ключами.

Функция GSS_Acquire_cred формирует удостоверение, в которое входит секретный ключ пользователя. Тем самым открывается доступ к этому ключу от имени пользователя.

Функция GSS_Init_sec_context получает от службы директорий заверенный сертификат сервера. Тем самым для клиента становится доступным открытый ключ сервера. GSS_Init_sec_context генерирует сеансовый ключ (как часть контекста безопасности) и шифрует его открытым ключом сервера (для вставки в выходной токен). Сеансовый ключ будет использоваться для защиты сообщений, пересылаемых между клиентом и сервером. В токен, выдаваемый функцией GSS_Init_sec_context, помимо зашифрованного сеансового ключа включаются сведения о клиенте, зашифрованные его (клиента) секретным ключом. Таким образом гарантируется конфиденциальность сеансового ключа (только сервер сможет его расшифровать) и достоверность информации о клиенте (посколь-

ку она заверена его электронной подписью).

Функция GSS_Accept_sec_context, вызываемая сервером, получает от службы директорий сертификат клиента и, в частности, его открытый ключ. После этого становится возможной проверка подлинности присланного токена безопасности и аутентификация клиента. Сервер расшифровывает своим секретным ключом присланный в токене сеансовый ключ. После этого может начаться эффективный обмен защищенными сообщениями между клиентом и сервером.

6. Детальное описание некоторых функций безопасности

В этом разделе детально описываются входные и выходные параметры важнейших функций безопасности, а также выполняемые этими функциями действия. Подробное рассмотрение позволяет лучше почувствовать дух спецификаций GSS-API. Кроме того, появляется возможность остановиться на некоторых сравнительно тонких моментах, которые выше были лишь упомянуты.

В описаниях интерфейсов после имени параметра следует имя его типа. Поскольку интерфейс GSS-API – обобщенный, в нем не уточняется отображение на конкретную языковую среду. Впрочем, во многих случаях трактовка типов очевидна.

В фигурных скобках после имени параметра и его типа следует разъяснение назначения параметра.

6.1. GSS_Acquire_cred – получение дескриптора удостоверения

Функция GSS_Acquire_cred предназначена для получения дескриптора удостоверения. Как правило, к ней обращаются только серверные компо-

ненты приложений (клиентские удовлетворяются подразумеваемыми удостоверениями).

Входные параметры:

- `desired_name`: INTERNAL_NAME {имя, которое должно быть вписано в удостоверение; NULL означает подразумеваемое значение, определяемое по локальным правилам}
- `lifetime_req`: INTEGER {запрашиваемый срок годности удостоверения, измеряемый в секундах; 0 означает подразумеваемый срок}
- `desired_mechs`: SET_OF_OBJECT_IDENTIFIER {набор механизмов безопасности, которые должны поддерживать данное удостоверение; пустое множество означает заявку на подразумеваемый механизм. Вероятно, каждый механизм разместит в удостоверении свою, нужную только ему информацию}
- `cred_usage`: INTEGER {возможные способы использования удостоверения: 0 – для инициализации и принятия контекстов, 1 – только для инициализации, 2 – только для принятия}

Выходные параметры:

- `major_status`: INTEGER {основной код завершения}
- `minor_status`: INTEGER {дополнительный код завершения}
- `output_cred_handle`: CREDENTIAL_HANDLE {дескриптор выданного удостоверения}
- `actual_mechs`: SET_OF_OBJECT_IDENTIFIER {набор предоставленных механизмов безопасности}
- `lifetime_rec`: INTEGER {срок годности удостоверения; для обозначения неограниченного срока имеется выделенное значение}

Возможные значения основного кода завершения:

- GSS_S_COMPLETE – запрашиваемое удостоверение ус-

пешно получено, со сроком годности (от текущего момента) `lifetime_rec` секунд, предназначенное для использования в соответствии со значением `cred_usage`, поддерживаемое набором механизмов безопасности `actual_mechs`. Для последующего доступа к удостоверению следует использовать дескриптор `output_cred_handle`.

- `GSS_S_BAD_MECH` – запрашивается неподдерживаемый механизм безопасности. Удостоверение не выдается (в качестве значения `output_cred_handle` возвращается `NULL`).
- `GSS_S_BAD_NAME` – запрашиваемое имя не удастся проинтерпретировать. Удостоверение не выдается.
- `GSS_S_BAD_NAME_TYPE` – запрашиваемое имя не удастся проинтерпретировать. Удостоверение не выдается.
- `GSS_S_BAD_NAME` – некорректное имя. Удостоверение не выдается.
- `GSS_S_FAILURE` – удостоверение не удалось выдать по причинам, не специфицируемым на уровне GSS-API (например, из-за отсутствия прав на использование запрашиваемого имени).

Из соображений мобильности лучше всего заказывать подразумеваемые значения там, где это возможно. Соответствующие выходные параметры позволяют узнать, что же реально предоставлено (например, какие механизмы безопасности поддерживаются).

6.2. GSS_Add_cred – постепенное формирование удостоверений

Функция `GSS_Add_cred` позволяет постепенно формировать удостоверения, пополняя их поддержкой новых механизмов безопасности, изменяя срок годности или предполагаемые способы использования.

Входные параметры:

- `cred_handle`: `CREDENTIAL_HANDLE` {дескриптор

удостоверения, полученный в результате предыдущих вызовов функций `GSS_Acquire_cred` или `GSS_add_cred`; `NULL` означает добавление к подразумеваемому удостоверению}

- `desired_name`: `INTERNAL_NAME` {имя, которое должно быть вписано в удостоверение; `NULL` обозначает подразумеваемое имя}
- `initiator_time_req`: `INTEGER` {запрашиваемый срок годности удостоверения для инициации контекстов; 0 – подразумеваемый срок}
- `acceptor_time_req`: `INTEGER` {запрашиваемый срок годности удостоверения для приема контекстов; 0 – подразумеваемый срок}
- `desired_mech`: `OBJECT_IDENTIFIER` {добавляемый механизм безопасности}
- `cred_usage`: `INTEGER` {способ использования удостоверения}

Выходные параметры:

- `major_status`: `INTEGER`
- `minor_status`: `INTEGER`
- `output_cred_handle`: `CREDENTIAL_HANDLE`
- `actual_mechs`: `SET_OF_OBJECT_IDENTIFIER`
- `initiator_time_rec`: `INTEGER` {время в секундах; специальное значение резервируется для неограниченного срока годности}
- `acceptor_time_rec`: `INTEGER`
- `real_cred_usage`: `INTEGER` {результатирующий способ использования удостоверения}

Смысл выходных параметров тот же, что и для функции `GSS_Acquire_cred`, лишь новое значение `real_cred_usage` показывает, для чего можно использовать скорректированное удостоверение. Прежними остались и основные коды завершения. Добавилось только одно новое значение – `GSS_S_NO_CRED`, означающее, что дескриптор `cred_handle` не соответ-

ствует корректное удостоверение.

Во всех вызовах функций `GSS_Acquire_cred` и `GSS_Add_cred`, имеющих дело с одним удостоверением, значения `desired_name` должны совпадать.

6.3. GSS_Init_sec_context – инициация контекста безопасности

Функция `GSS_Init_sec_context` предназначена для инициации контекста безопасности, формируемого общающимися партнерами, и для генерации токена, пересылка которого позволит удаленному партнеру выполнить свою часть работы по формированию контекста.

Входные параметры:

- `claimant_cred_handle`: `CREDENTIAL_HANDLE` {дескриптор удостоверения инициатора взаимодействия; `NULL` означает подразумеваемое удостоверение}
- `input_context_handle`: `CONTEXT_HANDLE` {дескриптор формируемого контекста; 0 означает, что формирование только начинается}
- `targ_name`: `INTERNAL_NAME` {имя партнера по общению}
- `mech_type`: `OBJECT_IDENTIFIER` {механизм безопасности, который должен поддерживать формируемый контекст; `NULL` означает подразумеваемый механизм}
- `deleg_rec_flag`: `BOOLEAN` {запрашивается ли делегирование полномочий}
- `mutual_req_flag`: `BOOLEAN` {запрашивается ли взаимная аутентификация}
- `replay_det_req_flag`: `BOOLEAN` {запрашивается ли выявление продублированных токенов}
- `sequence_req_flag`: `BOOLEAN` {запрашивается ли контроль целостности последовательности сообщений}

- anon_req_flag: BOOLEAN {запрашивается ли анонимный контекст}
- lifetime_req: INTEGER {запрашиваемый срок годности контекста в секундах; 0 означает подразумеваемый срок}
- chan_bindings: OCTET_STRING {набор каналов, с которым связывается контекст}
- input_token: OCTET_STRING {токен, полученный от партнера в результате предшествующих действий по формированию контекста; NULL означает, что формирование только начинается}
- trans_state: BOOLEAN {возможен ли экспорт контекста}
- prot_ready_state: BOOLEAN {обеспечивается ли защита сообщений, когда формирование контекста еще не завершено}
- conf_avail: BOOLEAN {обеспечивается ли шифрование сообщений}
- integ_avail: BOOLEAN {обеспечивается ли контроль целостности сообщений}
- lifetime_rec: INTEGER {срок годности контекста в секундах}

Возможные значения основного кода завершения:

Выходные параметры:

- major_status: INTEGER
- minor_status: INTEGER
- output_context_handle: CONTEXT_HANDLE {дескриптор формируемого контекста}
- actual_mech_type: OBJECT_IDENTIFIER {реальный механизм, поддерживающий формируемый контекст — значение, заведомо отличное от NULL}
- output_token: OCTET_STRING {выходной токен, который нужно переслать партнеру, чтобы тот мог продолжить формирование контекста; NULL означает, что больше ничего пересылать не нужно}
- deleg_state: BOOLEAN {обеспечивается ли передача полномочий}
- mutual_state: BOOLEAN {обеспечивается ли взаимная аутентификация}
- replay_det_state: BOOLEAN {обеспечивается ли выявление продублированных токенов}
- sequence_state: BOOLEAN {обеспечивается ли контроль целостности последовательности сообщений}
- anon_state: BOOLEAN {обеспечивается ли анонимность контекста}
- GSS_S_COMPLETE — контекст успешно инициализирован, а выходной токен содержит достаточно информации для завершения формирования контекста партнером и для начала защищенного обмена сообщениями.
- GSS_S_CONTINUE_NEEDED — сгенерированный выходной токен нужно переслать партнеру. Тот должен прислать ответ, который послужит аргументом input_token для последующего ассоциированного вызова функции GSS_Init_sec_context. Пока не будет выдан код завершения GSS_S_COMPLETE, контекст безопасности нельзя считать сформированным.
- GSS_S_DEFECTIVE_TOKEN — обнаружено нарушение целостности входного токена input_token. Формирование контекста не может быть продолжено.
- GSS_S_DEFECTIVE_CREDENTIAL — обнаружено нарушение целостности удостоверения, заданного аргументом claimant_cred_handle. Формирование контекста не может быть продолжено.
- GSS_S_BAD_SIG — входной токен input_token содержит некорректную подпись. Формирование контекста не может быть продолжено.
- GSS_S_NO_CRED — контекст не может быть сформирован либо по причине некорректности значения claimant_cred_handle, либо из-за того, что удостоверение не предназначено для инициации контекста, либо из-за отсутствия прав на использование данного удостоверения.
- GSS_S_CREDENTIALS_EXPIRED — удостоверение просрочено. Формирование контекста не может быть продолжено.
- GSS_S_BAD_BINDINGS — обнаружено несоответствие между информацией о наборе каналов, заданной аргументом chan_bindings, и аналогичной информацией, извлеченной из входного токена input_token. Несоответствие означает, что партнеры не могут договориться о привязке контекста к каналам. Формирование контекста не может быть продолжено.
- GSS_S_NO_CONTEXT — значение параметра input_context_handle не является корректным дескриптором контекста, в то время как оно должно быть таковым (выполняется не первый из ассоциированных вызовов GSS_Init_sec_context). Формирование контекста не может быть продолжено.
- GSS_S_BAD_NAME_TYPE — запрашиваемое имя не удается проинтерпретировать. Формирование контекста не может быть продолжено.
- GSS_S_BAD_NAME — некорректное имя. Формирование контекста не может быть продолжено.
- GSS_S_BAD_MECH — запрашивается неподдерживаемый механизм безопасности. Формирование контекста не может быть продолжено.

- GSS_S_FAILURE — формирование контекста не может быть продолжено по причинам, не специфицируемым на уровне GSS-API.

Вызов функции GSS_Init_sec_context может привести к временной блокировке обратившегося к ней процесса, если для генерации контекста и получения выходного токена служба безопасности должна воспользоваться услугами сервера аутентификации, службы директорий или иного удаленного сервера.

Обычно для инициации контекста достаточно одного обращения к функции GSS_Init_sec_context. Если это не так (например, из-за требования взаимной аутентификации), то для всех последующих обращений значение параметра claimant_cred_handle должно оставаться неизменным. Это значение позволяет связать в цепочку ассоциированные вызовы GSS_Init_sec_context.

При первом обращении к GSS_Init_sec_context значение параметра input_context_handle должно равняться 0. При последующих вызовах его следует устанавливать равным output_context_handle (ассоциированные вызовы Goutput_context_handle, начиная со второго, не меняют дескриптор контекста безопасности).

Инициатор, посредством входных флагов, может запросить у службы безопасности предоставление дополнительных услуг: возможности делегирования прав доступа (флаг deleg_rec_flag), организации взаимной аутентификации (mutual_req_flag), контроля целостности последовательности сообщений (флаги replay_det_req_flag и sequence_req_flag), сохранения инкогнито инициатора контекста (anon_req_flag). Соответствующие выходные флаги показывают, в состоянии ли служба безопасности предос-

тавить запрашиваемые услуги. Кроме того, еще два выходных флага, integ_avail и conf_avail, показывают, поддерживается ли службой безопасности контроль целостности сообщений и их конфиденциальность (в принципе можно представить себе механизм, обеспечивающий только аутентификацию партнеров).

Отказ в поддержке некоторых запрашиваемых услуг не обязательно должен вести к отказу от формирования контекста безопасности. Мобильное приложение может принять те или иные меры (например, проинформировать пользователя о достижимом уровне защиты, взять на себя шифрование и т.п.), позволяющие продолжить работу.

6.4. GSS_Accept_sec_context — принятие контекста безопасности

Функция GSS_Accept_sec_context предназначена для продолжения (как правило — завершения) формирования контекста безопасности. Мы будем называть это принятием контекста.

Входные параметры:

- acceptor_cred_handle: CREDENTIAL_HANDLE {дескриптор удостоверения партнера, принимающего контекст; NULL означает подразумеваемое удостоверение}
- input_context_handle: CONTEXT_HANDLE {дескриптор формируемого контекста; 0 означает, что принятие только начинается}
- chan_bindings: OCTET_STRING {набор каналов, с которым связывается контекст}
- input_token: OCTET_STRING {токен, полученный от партнера в результате предшествующих действий по формированию контекста}

Выходные параметры:

- major_status: INTEGER
- minor_status: INTEGER
- src_name: INTERNAL_NAME {имя инициатора формирования контекста}
- actual_mech_type: OBJECT_IDENTIFIER {реальный механизм, поддерживающий формируемый контекст — значение, заведомо отличное от NULL}
- output_context_handle: CONTEXT_HANDLE {дескриптор формируемого контекста}
- deleg_state: BOOLEAN {обеспечивается ли передача полномочий}
- mutual_state: BOOLEAN {обеспечивается ли взаимная аутентификация}
- replay_det_state: BOOLEAN {обеспечивается ли выявление продублированных токенов}
- sequence_state: BOOLEAN {обеспечивается ли контроль целостности последовательности сообщений}
- anon_state: BOOLEAN {обеспечивается ли анонимность контекста}
- trans_state: BOOLEAN {возможен ли экспорт контекста}
- prot_ready_state: BOOLEAN {обеспечивается ли защита сообщений, когда формирование контекста еще не завершено}
- conf_avail: BOOLEAN {обеспечивается ли шифрование сообщений}
- integ_avail: BOOLEAN {обеспечивается ли контроль целостности сообщений}
- lifetime_rec: INTEGER {срок годности контекста в секундах}
- delegated_cred_handle: CREDENTIAL_HANDLE {дескриптор удостоверения, позволяющего инициировать новые контексты безопасности от имени партнера по

общению и, тем самым, пользоваться его правами доступа}

- output_token: OCTET_STRING {выходной токен, который нужно переслать партнеру, чтобы тот мог продолжить формирование контекста; NULL означает, что больше ничего пересылать не нужно}

Функция GSS_Accept_sec_context во многом аналогична функции инициации контекста GSS_Init_sec_context. Это относится и к кодам завершения, и к возможности повторных вызовов (если для формирования контекста требуется обмен несколькими токенами), и к блокировке обратившегося процесса, и к трактовке большинства входных и выходных параметров.

Отдельного рассмотрения заслуживают выходные параметры src_name и delegated_cred_handle. Первый из них содержит имя инициатора формирования контекста. В принципе анализ этого имени может повлиять на поведение принимающего контекст. Если обеспечивается анонимность инициатора (установлен флаг anon_state), параметр src_name получает значение NULL (имя инициатора не было включено в токен безопасности).

Если установлен флаг deleg_state, то параметр delegated_cred_handle содержит дескриптор удостоверения, выданного на имя инициатора формирования контекста. Тем самым принимающей стороне делегируются полномочия формировать новые контексты безопасности не только от своего имени, но и от имени инициатора. Делегирование полномочий полезно, когда серверу требуется выполнить некоторые действия от имени клиента (например, сервер печати должен прочитать выдаваемые файлы).

6.5. GSS_GetMIC – обеспечение целостности сообщения

Функция GSS_GetMIC служит для обеспечения контроля целостности и подлинности источника сообщения. Сообщение и генерируемый токен безопасности являются корректными входными данными для последующего обращения к функции GSS_VerifyMIC.

Входные параметры:

- context_handle: CONTEXT_HANDLE {дескриптор контекста, в рамках которого происходит обмен сообщениями}
- qop_req: INTEGER {запрашиваемое качество защиты; 0 обозначает подразумеваемое качество}
- message: OCTET_STRING {защищаемое сообщение}

Выходные параметры:

- major_status: INTEGER
- minor_status: INTEGER
- per_msg_token: OCTET_STRING {генерируемый токен безопасности}

Возможные значения основного кода завершения:

- GSS_S_COMPLETE – сообщение и сгенерированный токен безопасности готовы для передачи партнеру.
- GSS_S_CONTEXT_EXPIRED – истек срок годности контекста.
- GSS_S_CREDENTIALS_EXPIRED – контекст распознан как корректный, однако срок годности ассоциированных с ним удостоверений истек.
- GSS_S_NO_CONTEXT – не задан контекст.
- GSS_S_BAD_QOP – некорректное значение параметра qop_req.
- GSS_S_FAILURE – сообщение не может быть защищено по причинам, не специфицируемым на уровне GSS-API.

С помощью информации, содержащейся в контексте без-

опасности, для сообщения определяется порядковый номер и/или временной штамп, порождается электронная подпись отправителя и т.п. Сгенерированные данные включаются в токен безопасности, который следует передать вместе с исходным сообщением (последнее функцией GSS_GetMIC не изменяется).

Естественно, успешное завершение функции GSS_GetMIC не гарантирует, что после передачи сообщения удаленному партнеру проверка, выполняемая функцией GSS_VerifyMIC, даст положительный результат.

6.6. GSS_VerifyMIC – проверка целостности сообщения

Функция GSS_VerifyMIC позволяет проверить целостность и подлинность источника сообщения, защищенного функцией GSS_GetMIC.

Входные параметры:

- context_handle: CONTEXT_HANDLE {дескриптор контекста, в рамках которого происходит обмен сообщениями}
- message: OCTET_STRING {контролируемое сообщение}
- per_msg_token: OCTET_STRING {токен безопасности, ассоциированный с сообщением}

Выходные параметры:

- qop_state: INTEGER {обеспеченное качество защиты}
- major_status: INTEGER
- minor_status: INTEGER

Возможные значения основного кода завершения:

- GSS_S_COMPLETE – проверка дала положительный результат.
- GSS_S_DEFECTIVE_TOKEN – обнаружено искажение токена безопасности.
- GSS_S_BAD_SIG – токен безопасности содержит не-

корректную электронную подпись.

- GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN, GSS_S_UNSEQ_TOKEN, GSS_S_GAP_TOKEN – обнаружено дублирование или нарушение последовательности сообщений.
- GSS_S_CONTEXT_EXPIRED, GSS_S_CREDENTIALS_EXPIRED, GSS_S_NO_CONTEXT – то же, что и в случае GSS_GetMIC.
- GSS_S_FAILURE – сообщение не может быть проверено по причинам, не специфицируемым на уровне GSS-API.

С помощью информации, содержащейся в контексте безопасности, проверяется целостность и подлинность источника сообщения и ассоциированного токена безопасности (корректность значения криптографической контрольной суммы, подлинность электронной подписи, допустимость порядкового номера и временного штампа и т.п.).

Выходной параметр `qop_state` показывает, защита какого качества была задействована соответствующим вызовом функции GSS_GetMIC.

6.7. GSS_Wrap – шифрование сообщения

Функция GSS_Wrap, помимо услуг, предоставляемых вызовом GSS_GetMIC, может обеспечить шифрование сообщения. В отличие от GSS_GetMIC, результатом работы GSS_Wrap является инкапсулированное сообщение, содержащее необходимую контрольную информацию, а не исходное сообщение с дополнительным токеном безопасности.

Входные параметры:

- context_handle: CONTEXT_HANDLE {дескриптор контекста, в рамках которого происходит обмен сообщениями}

- conf_req_flag: BOOLEAN {нужно ли шифровать защищаемое сообщение}
- qop_req: INTEGER {запрашиваемое качество защиты; 0 обозначает подразумеваемое качество}
- input_message: OCTET_STRING {защищаемое сообщение}

Выходные параметры:

- major_status: INTEGER
- minor_status: INTEGER
- conf_state: BOOLEAN {было ли сообщение на самом деле зашифровано}
- output_message: OCTET_STRING {инкапсулированное сообщение}

Возможные значения основного кода завершения для функции GSS_Wrap те же, что и для GSS_GetMIC.

С помощью информации, содержащейся в контексте безопасности, сообщение защищается от искажений и, быть может, шифруется. Порождаемое выходное сообщение `output_message` подлежит передаче партнеру по общению.

6.8. GSS_Unwrap – расшифровка сообщения

Функция GSS_Unwrap позволяет расшифровать, проверить целостность и подлинность источника применительно к сообщению, защищенному функцией GSS_Wrap.

Связь между функциями GSS_Wrap и GSS_Unwrap аналогична той, что существует между функциями GSS_GetMIC и GSS_VerifyMIC.

Входные параметры:

- context_handle: CONTEXT_HANDLE {дескриптор контекста, в рамках которого происходит обмен сообщениями}
- input_message: OCTET_STRING {контролируемое сообщение}

Выходные параметры:

- conf_state: BOOLEAN {было ли сообщение на самом деле зашифровано}
- qop_state: INTEGER {обеспеченное качество защиты}
- major_status: INTEGER
- minor_status: INTEGER
- output_message: OCTET_STRING {расшифрованное сообщение}

Возможные значения основного кода завершения для функции GSS_Unwrap те же, что и для GSS_VerifyMIC.

С помощью информации, содержащейся в контексте безопасности, функция GSS_Unwrap расшифровывает сообщение, проверяется его целостность и подлинность источника. Расшифрованное сообщение выдается в качестве значения выходного параметра `output_message`.

7. Ограничения интерфейса безопасности

Обобщенный прикладной программный интерфейс службы безопасности GSS-API не содержит средств, направленных на контроль прав доступа или хотя бы на передачу авторизационной информации. Для этого, однако, есть веские причины.

Интерфейс безопасности должен обслуживать разнообразные приложения. Содержательная трактовка прав доступа, как и определение того, что (или кто) является объектом или субъектом контроля, может быть выполнена только самими приложениями. Даже для устоявшихся и сравнительно однородных сервисов, таких как файловый, авторизация в разных операционных средах может носить существенно разный характер. Более того, характер авторизации может меняться и для одной операционной Среды при переходе от изолированной системы к рас-

пределенной конфигурации. В качестве примера достаточно привести SunOS, где правила доступа к файлам, находящимся на смонтированных NFS-томах, гораздо сложнее, чем в локальном случае.

Другую причину можно назвать политической. Спецификациям GSS-API в их нынешнем виде удовлетворяет такая популярная система, как Kerberos, не содержащая средств контроля доступа. Специфицировать подобные средства в GSS-API — значит лишиться поддержки со стороны Kerberos. В результате такой акции интерфейс безопасности "повиснет в воздухе", из-под него будет выбита почва.

В то же время проблемы авторизации, делегирования полномочий с ограничениями, несомненно, требуют решения (на прикладном уровне). Как передать серверу печати право на доступ только к определенным файлам и только на чтение? Как защитить строки своих таблиц в базе данных от удаления сервером приложений, оставив за ним возможность добавления информации? В общем виде решение очевидно — нужно перестроить систему авторизации на основе архитектуры клиент/сервер. В нынешней ситуации, когда каждое приложение (операционная система, СУБД, почтовая служба и т.п.) использует специфические методы контроля доступа, универсальное, стандартное решение в духе открытых систем получить невозможно.

Впрочем, первый и весьма важный шаг уже сделан. Имеется в виду реализация серверов аутентификации, которые позволили системно-независимым образом идентифицировать субъекты доступа. Трудно сказать, насколько быстро процесс пойдет дальше. По-видимому, рассчитывать на быстрый прогресс не прихо-

```
typedef struct gss_buffer_desc_struct {
    size_t length;
    void *value;
} gss_buffer_desc, *gss_buffer_t;
```

Листинг 1.

```
typedef struct gss_OID_desc_struct {
    OM_uint32 length;
    void *elements;
} gss_OID_desc, *gss_OID;
```

Листинг 2.

```
typedef struct gss_OID_set_desc_struct {
    int count;
    gss_OID elements;
} gss_OID_set_desc, *gss_OID_set;
```

Листинг 3.

```
OM_uint32 GSS_Init_sec_context (
    OM_uint32 *minor_status,
    gss_cred_id_t claimant_cred_handle,
    gss_ctx_id_t *context_handle,
    gss_name_t targ_name,
    gss_OID mech_type,
    int req_flags,
    OM_uint32 time_req,
    gss_channel_bindings_t chan_bindings,
    gss_buffer_t input_token,
    gss_OID *actual_mech_type,
    gss_buffer_t output_token,
    int *ret_flags,
    OM_uint32 *time_rec
);
```

Листинг 4.

```
#define GSS_C_DELEG_FLAG 1
#define GSS_C_MUTUAL_FLAG 2
. . .
#define GSS_C_CONF_FLAG 16
. . .
#define GSS_C_QOP_DEFAULT 0
#define GSS_C_INDEFINITE 0xfffffffful
#define GSS_S_COMPLETE 0
. . .
```

Листинг 5.

дится — слишком велика масса программного обеспечения, нуждающегося в пересмотре.

Еще одна проблема технического плана, вытекающая из архитектуры интерфейса GSS-API, состоит в том, что для коротких сообщений накладные расходы, связанные с обеспечением целостности и конфиденциальности данных, могут оказаться весьма значительными. Это может резко ухудшить характеристики таких протоколов, как telnet. Выбор минимально допустимого

качества защиты и буферизация данных — вот методы, которые, вероятно, способны улучшить ситуацию.

8. Представление некоторых объектов интерфейса безопасности в среде языка C

Представление средства языка C объектов, фигурирующих в обобщенном интерфейсе безопасности GSS-API, по большей части довольно очевидно (или не может быть стандартизовано, так как зависит от

специфики механизма безопасности).

Прежде всего, вводится тип `OM_uint32`, соответствующий 32-битным беззнаковым целым значениям. Большинство структурных значений представляется с помощью указателя на дескриптор (листинг 1).

Тип `gss_buffer_t` используется при задании составных аргументов — имен, дескрипторов, токенов, сообщений и т.п.

Идентификаторы объектов представляются следующим образом (листинг 2).

Указатели `elements` ссылаются на начало представления идентификаторов, то есть на последовательности байт, устроенных в соответствии с базовыми правилами ASN.1.

Наборы объектных идентификаторов представлены на листинге 3.

Вводятся и некоторые другие типы, уточняющие представление структурированных значений.

Описание функции `GSS_Init_sec_context` на языке C показано на листинге 4.

Отметим, что параметр `context_handle` является здесь одновременно входным и выходным, а основной код завершения возвращается как результат функции.

На листинге 5 приведено определение еще нескольких величин.

Разумеется, есть еще много аспектов, оговоренных в документе [3], например, кто и когда отводит память под объекты и под дескрипторы, и каким образом эту память можно освобождать. Мы, однако, не будем на этом останавливаться.

9. Заключение

Обобщенный прикладной программный интерфейс службы безопасности GSS-API — попытка распространения идеологии открытых систем на такую важную и сложную область, как защита коммуникаций в среде клиент/сервер. Попытка эта представляется успешной. Сформулирован компактный интерфейс, допускающий естественное отображение на разные языковые среды и различные механизмы безопасности. В то же время выра-

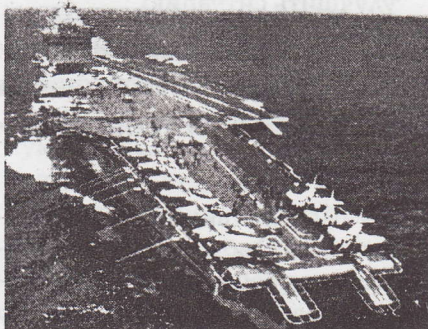
зительных возможностей интерфейса достаточно для решения задач аутентификации партнеров по общению и обеспечения целостности и конфиденциальности пересылаемой информации.

Литература

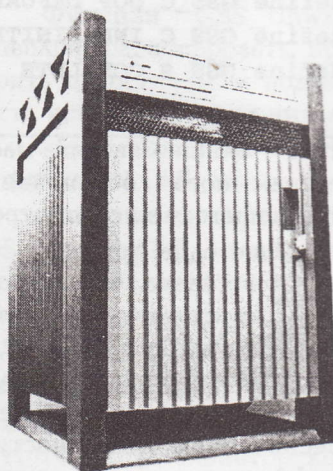
1. J. Linn. Generic Security Service Application Program Interface, Version 2. — Internet-Draft, February 1996.
2. J. Linn. Generic Security Service Application Program Interface. — Request for Comments: 1508, September 1993.
3. J. Wray. Generic Security Service API: C-bindings. — Request for Comments: 1509, September 1993.
4. J. Kohl. The Kerberos Network Authentication Service (V5). — Request for Comments: 1510, September 1993.
5. Recommendation X.509. The Directory — Authentication Framework. — Melbourne, 1988.



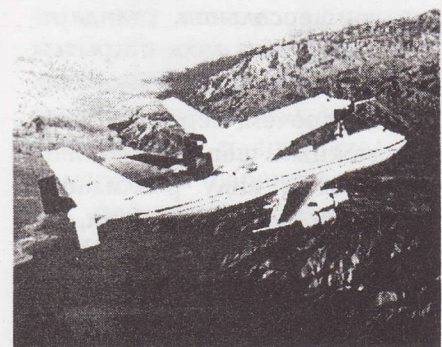
ULTRA™ ENTERPRISE™



Enterprise — действительно мощный сервер



Новый сервер компании Sun Microsystems — Ultra Enterprise 3000



Enterprise — это, конечно, корабль многообразного использования



Управление Автоматизации как банковский сервер

Последние годы в России характеризуются разного рода потрясениями. Одни из них имеют почти вселенский масштаб, другие носят статус локальных событий. Первые больше на слуху, но вторые ближе к телу. Некоторые из вторых повторяются настолько часто и становятся настолько характерны, что о них уже пора кричать в полный голос, что я и собираюсь сделать.

Еще несколько лет назад мы все жили в совке и почти все имели совковое мышление в отношении всего чего угодно. Этим мы были похожи. В других отношениях мы были разными. Среди прочих групп и слоев граждан была одна, которая с известной гордостью называла себя технической интеллигенцией. Споры нет, технари обладали порой совершенно уникальными знаниями. Этим можно (и нужно!) было гордиться, но профессиональных навыков работы "в фирме" никто не имел. (Как ни прискорбно, не имел их и я.) Пришла "бархатная революция". Появились первые ростки новых рыночных отношений, в том числе многочисленные коммерческие банки. Кто же взялся за их автоматизацию? Конечно, технари. И тут началось!

Люди, которые вчера сидели за перегородками в мрачных комнатных помещениях, сегодня попали в светлые помещения с хорошими столами. Раньше они мирно общались со своими сослуживцами или сами с собой, теперь им пришлось общаться с КЛИЕНТАМИ. Изменился темп деятельности: если раньше начальство не вспоминало про них неделями, то теперь они оказывались нужны сразу нескольким руководителям по нескольку раз в день. Для многих это настоящее потрясение, которое затме-

вает политические игры любого масштаба.

Маленькие стрессы имеют свойство объединяться вместе. И через некоторое время это становится проблемой не одного человека, а всего коллектива. Если данную болезнь запустить, то скоро она становится хронической, и тогда хирург автоматически превращается в патологоанатома. "Случай запущенный -- вы лечению не подлежит".

К чему такой пафос? А к тому, что, как ни странно, данная проблема является главной применительно к Управлениям Автоматизации (УА) банков (именно так я буду называть ИТ-департаменты российских коммерческих банков). Более того, первое, что должен сделать конструктивно настроенный начальник УА, это четко и ясно позиционировать себя и своих сотрудников по отношению к другим сотрудникам банка. Смысл такого позиционирования выражается одной фразой: "СОТРУДНИКИ БАНКА ЯВЛЯЮТСЯ КЛИЕНТАМИ (!!!) СОТРУДНИКОВ УПРАВЛЕНИЯ АВТОМАТИЗАЦИИ".

Это означает, что ко всем другим сотрудникам банка следует относиться именно как к клиентам, со всеми присущими такому отношению атрибутами. Должен существовать стандарт на обслуживание клиентов, который включает в себя как минимум:

- перечень услуг;
- ритуал оказания этих услуг;
- регламент в случае возникновения нештатных ситуаций.

Одним словом, Вы должны заключить с клиентом соглашение об услугах. Такое соглашение не может быть сформулировано устно. Наиболее правильный

путь — составить документ под названием "ПОЛОЖЕНИЕ ОБ УСЛУГАХ" (далее называемое СОГЛАШЕНИЕ) и зафиксировать его на уровне руководства банка. Разумеется, по мере накопления изменений содержимое такого документа будет меняться. При этом разногласия в точках зрения на качество обслуживания должны быть урегулированы еще на этапе обсуждения изменений к СОГЛАШЕНИЮ.

Что происходит, если этим принципам не следовать? Возможны две крайности, примерно в равной пропорции тиражируемые в большинстве коммерческих банков.

Первый вариант. Начальник УА имеет мягкий, податливый характер. Тогда в банке пыльным цветом расцветают горизонтальные свойские отношения. Сотрудники банка (сослуживцы) живут как бы одной большой коммуной, где все друзья, товарищи и братья.

В такой атмосфере появляются любимчики и негодники. В отдельно взятых подразделениях вдруг немотивированно расширяется дисковое пространство, ПК оснащаются CD-ROM-устройствами и звуковыми колонками. В то же время, в других подразделениях сотрудники трудятся на допотопных ПК производства времен царя Гороха. Общий уровень сервиса неуклонно снижается, что, однако, не удается померить ввиду отсутствия механизма.

Второй вариант. Начальник УА имеет жесткий, неуступчивый характер. Тогда все обращения к нему встречают жесткий отпор. Любой запрос, даже выраженный в форме прошения, воспринимается как объяв-

ление войны. Некоторым подчиненным начальника УА это нравится. Они копируют его поведение, и Управление Автоматизации превращается в неприступную крепость. Только одно оружие в состоянии пробить броню такой крепости — слово верховного главнокомандующего. Конечно, долго так продолжаться не может. Рано или поздно одно из таких слов будет последним для начальника УА на этом посту.

Что же надо делать на практике, чтобы избежать такого печального конца?

Допустим, Вы осознали необходимость относиться к сотрудникам банка как к клиентам. Это самое главное.

Теперь надо перечислить все аппаратные и программные продукты, которыми пользуются Ваши КЛИЕНТЫ. В программной части элемент подобного списка будет выглядеть, например, так:

- У25 — Автоматизированная банковская система
- У26 — Редактор текстов Лексикон
- У27 — Редактор текстов Word
- У28 — Электронная таблица Excel
- У29 — Информационно-правовая система "Консультант-Плюс"
- У30 — Электронная почта cc:Mail

Теперь на каждую позицию нужно сделать краткую аннотацию, в которой отмечается, для каких прикладных целей используется каждый программный продукт и как он сопровождается. Например, о редакторе Word Вы напишите:

Услуги отложенного действия (по заявке):

- У27-О1 — обучение навыкам работы со стандартным интерфейсом;
- У27-О2 — обучение использованию разделов меню: Help, Window;

- У27-О3 — обучение использованию разделов меню: File, Edit, View;
- У27-О4 — обучение использованию разделов меню: Format, Table.

Услуги немедленного действия (по запросу):

- У27-Н1 — ответы по стандартным функциям редактора (см. выше);
- У27-Н2 — настройка каталогов шаблонов;
- У27-Н3 — консультационная справка о наборе шрифтов;
- У27-Н4 — нештатные ситуации.

Не оказываются услуги:

- обучение программированию макросов Visual Basic;
- раздел меню Tools."

Услуги отложенного действия выполняются только на основании письменной заявки по заранее составленному расписанию, например, один раз в две недели в вечернее время или выходные дни. В зависимости от количества поданных заявок, частоту проведения семинаров по обучению можно увеличивать, но не уменьшать. В противном случае будет увеличиваться время ввода в эксплуатацию нового рабочего места.

Услуги немедленного действия оказываются в тот момент, когда служба сервиса получает запрос по телефону. Услуги этого разряда являются основными. Оперативное оказание этих услуг должно стать главной задачей Вашей службы сервиса. Должна быть разработана система учета и анализа запросов, инструкции персоналу, система эскалации информации о нештатных ситуациях и т.д.

Нештатным ситуациям уделяется особое внимание. Если для остальных позиций предусматривается характер-

ное время разрешения проблемы порядка 15 минут, то для нештатных ситуаций отводится время 3-4 часа. При этом надо понимать, что пользователь в общем случае не понимает разницы между обычным вопросом о настройке редактора и действительно нештатной ситуацией. И то и другое не дает ему возможности полноценно работать.

В разделе *Не оказываются услуги* перечисляются те услуги, которые не оказываются ни при каких условиях. Только тогда, когда персонал Вашей службы сервиса четко понимает, что он должен и что не должен делать, Вы можете быть уверены в надежном предоставлении качественного обслуживания.

Если Вы согласны с тезисом **"СОТРУДНИК БАНКА — МОЙ КЛИЕНТ"** и собираетесь продвигать его в жизнь, остановитесь на мгновение. Учтите, что на этом пути у Вас будет мало сторонников, но очень много противников. На Вас будет оказываться давление сразу с двух сторон. Во-первых, менеджеры разного уровня будут требовать особенного к себе отношения, что не согласуется с понятием стандартизованного клиента. Вам надо будет проявить известную изворотливость. Во-вторых, Ваши собственные сотрудники по разным причинам начнут противодействовать: будет и открытое неприятие, и тихий саботаж. Поэтому в начале пути не надо резких движений, действуйте тихой сапой — так оно вернее.

И никому не показывайте эту статью!

Об авторе. Рассказов Алексей в 1993-1996 г.г. занимал должность начальника отдела автоматизации Русского Продовольственного банка.



AVD Systems

Партнер Jet Infosystems, компания AVD Systems, представляет RTworks — экспертную систему реального времени.

Экспертная система реального времени RTworks — это комплекс инструментальных средств для разработки программного обеспечения сложных распределенных интеллектуальных систем управления реального времени: АСУ технологических процессов; менеджмент больших сетей; системы управления транспортным движением; наземные телеметрические комплексы; системы контроля окружающей среды; управление трубопроводами и т.д.

RTworks относится к категории "framework software", т.е. каркаса общего назначения, на котором базируется проблемно-ориентированное ПО РВ. Этот каркас предназначен для сокращения сроков разработки больших программных систем реального времени с открытой архитектурой клиент/сервер и интерфейсами, стандартными для программной отрасли.

Модульность и масштабируемость

Система RTworks состоит из 6-ти компонент, называемых 'процессами', которые позволяют про-

извести декомпозицию сложной системы управления на подсистемы в соответствии с физической структурой объекта управления :

- **RTdaq**, Data Acquisition Process — Интерфейс со внешними источниками данных
- **RTie**, Inference Engine Process — Оболочка экспертной системы и машина логического вывода
- **RThci**, Human Computer Interface Process — Визуализатор динамических данных для разработки интерфейса оператора сложных систем РВ
- **RTarchive**, Data Archive Process — Процесс накопления информационного архива
- **RTplayback**, Data Playback Process — Процесс воспроизведения информационного архива
- **RTserver**, Interprocess Communication Server Process — Сервер межпроцессной коммуникации.

Обмен информацией между процессами-клиентами в RTworks происходит через RTserver с использованием унифицированного протокола обмена сообщениями. В конкретной системе может работать несколько экземпляров какого-либо RTworks-процесса, а может не быть и ни одного.

Распределенность

Процессы RTworks могут быть распределены для исполне-

ния на различные узлы сети, которая может быть и неоднородной. Перераспределение процессов (например, для повышения производительности системы) не требует никаких изменений в прикладном программном обеспечении.

Открытость

Все процессы RTworks имеют открытый программный интерфейс для расширения функциональных возможностей, предоставляемых данным процессом. Пользователь может также подключать к RTserver'у собственные клиентские процессы.

Отказоустойчивость (24x7)

RTworks предназначена для построения систем, работающих 24 часа в сутки 7 дней в неделю. Каждый процесс в системе может резервироваться дублирующим процессом, получающим те же сообщения и данные, что и основной. Как правило, программное резервирование применяется вместе с аппаратным, т.е. дублирующий процесс выполняется на другом компьютере.

Поддерживаемые платформы

Sun SPARC (SunOS и Solaris), HP 9000/700 (HP-UX), DEC Alpha (OpenVMS и OSF/1), IBM RS6000 (AIX) и SGI (IRIX).

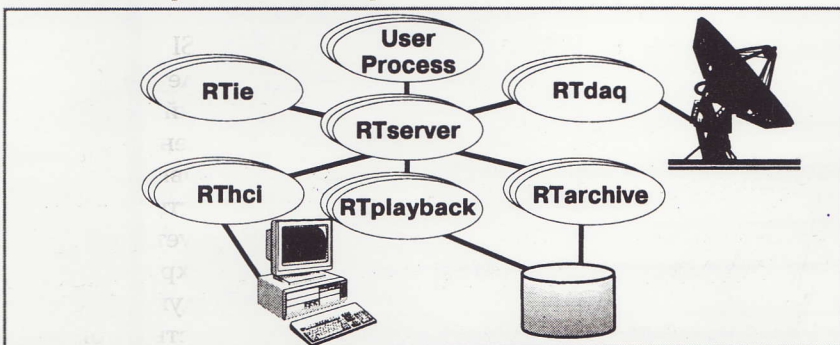
Наиболее популярной платформой для системы RTworks являются рабочие станции компании Sun Microsystems. Для них в первую очередь выпускаются новые версии. Большинство устанавливаемых систем RTworks в России произведено на этих рабочих станциях.

AVD Systems

121170 Москва а/я 38

tel/fax: (095) 145-1169

e-mail: avd@avdsys.msk.su



INFO

Информационный бюллетень Jet Info

Индекс по каталогу РОСПЕЧАТИ - 32555

Главный редактор: В.А.Галатенко
Технический редактор: С.И.Демочкин

Полное или частичное воспроизведение материалов, содержащихся в настоящем издании, допускается только с разрешения Jet Infosystems

Jet Infosystems

Россия, 103006, Москва,
ул.Краснопролетарская, 6
тел. (095) 972 11 82
(095) 972 13 32
факс (095) 972 07 91
e-mail: JetInfo@jet.msk.su

Перед Вами один из номеров бесплатного периодического информационно-технического бюллетеня "Jet Info", издаваемого компанией "Инфосистемы Джет" - ведущим российским системным интегратором в области UNIX-систем.

Если Вы еще не являетесь подписчиком "Jet Info", но хотели бы получать его регулярно, просим Вас отправить в наш адрес заполненный купон этого объявления. Отправьте этот купон и в том случае, если Вы сменили свой почтовый адрес или решили изменить форму доставки Вам бюллетеня.

Информация для подписчиков: не забудьте в течение каждого декабря и июля обязательно отправлять нам этот отрывной купон для перерегистрации.



1. Организация _____

2. Фамилия, имя, отчество и должность лица, в чей адрес будет осуществляться рассылка _____

3. Индекс и почтовый адрес _____

4. Телефон и/или факс _____

5. Адрес электронной почты _____

6. В какой форме Вы хотели бы получать бюллетень

печатный вариант по почте

электронную версию по электронной почте

7. Прежняя форма и адрес подписки (для подписчиков, меняющих адрес и форму подписки, либо тех, кто проходит перерегистрацию) _____

8. Общее число компьютеров в организации _____

9. Ваша организация использует

Вычислительную технику

Apple

DEC Alpha

DEC VAX

IBM PC

HP 9000

Silicon Graphics

Sun Microsystems

Другое _____

Операционные системы

MS-DOS (Windows)

Novell NetWare

SCO Unix

Solaris / SunOS

UnixWare

VAX VMS

Другое _____

10. Материалы на какую тему заинтересовали бы Вас в первую очередь _____

Мы будем рады, если кто-то из Вас захочет принять участие в подготовке выпусков "Jet Info" в качестве автора. Будем также признательны за любые конструктивные замечания и предложения по всем аспектам подготовки, выпуска и распространения нашего бюллетеня.

Редакция "Jet Info"