МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Программные средства аутентификации в браузере для операционной системы QP OS

Студент 18.06.21г Ширикова Вероника Михайловна
(подпись, дата) (ФИО полностью)

Руководитель 18.06.21г Князев В. Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 22.06.21 Такташкин Д. В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой Макарычев П. П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н. А.
(подпись) (фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«21» января 2021 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Ширикова Вероника Михайловна гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства аутентификации в браузере для операционной системы QP OS

Тема утверждена приказом ПГУ №_____от "_28__"__декабря_____2020

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н. _____

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: разработать программные средства аутентификации пользователя в браузере для операционной системы QP OS
Среда выполнения: операционная система QP OS
Среда разработки: операционная система Windows
Языковые средства разработки: C#, UML
Инструментальные средства разработки: Microsoft Visual Studio 2019, ArgoUML, Erwin Data Modeler, Microsoft Project
Технологии разработки:RUP, ООП.
Основные функции:
а) идентификация пользователя по введенному логину
б) выбор схемы аутентификации
в) аутентификации с помощью протокола Kerberos, NTLM, Digest или Basic
г) доступ к ресурсу после проверки подлинности данных пользователя

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Тестирование

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	*Анализ предметной области постановка задачи*	15%	14.04.2021–21.04.2021	
2	*Проектирование*	25%	22.04.2021–05.05.2021	
3	*Реализация*	40%	06.05.2021–30.05.2021	
4	*Тестирование*	5%	31.05.2021–02.06.2021	
5	*Оформление пояснительной записки*	15%	03.06.2021–14.06.2021	

Дата выдачи задания "28" декабря 2020

Руководитель бакалаврской работы _____ 28.12.2020г.
(подпись, дата)

Задание к исполнению принял студент _____ 28.12.2020г.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Л.Р. Фомичева
(подпись, дата) 24.06.2021

Реферат

Пояснительная записка содержит 110 листов, 22 рисунка, 9 таблиц, 24 использованных источников, 5 приложений.

C#, UML, QP OS, ООП, СЕРВЕР, ДВУСТОРОННЯЯ АУТЕНТИФИКАЦИЯ, БРАУЗЕР, РОССИЙСКИЕ КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ, KERBEROS

Объектом разработки являются программные средства аутентификации, внедренные в браузер специального назначения ООО НТП «Криптософт».

Цель работы – разработка средств двусторонней аутентификации для операционной системы QP OS.

Результатом разработки является исполняемый файл «Titan.exe», предназначенный для работы под управлением операционной системы QP OS.

Проектирование проводилось с помощью языка моделирования UML с использованием среды разработки диаграмм «ArgoUML».

Разработка проводилась на языке программирования C#. В качестве среды разработки использовалась IDE Visual Studio 2019. Разработка проводилась с использованием операционной системs Windows 10. Среда исполнения – операционная система QP OS.

Был получен акт о внедрении данного продукта в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

Содержание

Введение 6
1 Анализ предметной области и требований 8
1.1 Анализ предметной области 8
1.2 Анализ программного обеспечения аналогичного назначения 12
1.2.1 Анализ аутентификации Google Chrome 12
1.2.2 Анализ аутентификации Microsoft Edge 13
1.2.3 Результат анализа программ аналогов 14
1.3 Анализ требований 15
1.3.1 Анализ функциональных требований 15
1.3.2 Анализ нефункциональных требований 20
1.4 Планирование разработки и оценки бюджета 20
2 Проектирование программных средств аутентификации в браузере для операционной системы QP OS 32
2.1 Проектирование архитектуры 32
2.2 Проектирование интерфейса 34
3 Реализация программных средств аутентификации в браузере для операционной системы QP OS 36
4 Тестирование программных средств 38
4.1 Функциональное тестирование 38
4.2 Модульное тестирование 38
4.3 Метрики кода 41
4.3.1 Цикломатическая метрика Мак-Кейба 41
4.3.2 Метрика Холстеда 44
4.3.3 Метрика Чепина 47
Заключение 49
Список использованных источников 50
Приложение А. Глоссарий 53
Приложение Б. Исходный код программных средств 55
Приложение В. Результаты тестирования 86

Приложение Г. Исходный код функций тестирования 89
Приложение Д. Акт о внедрении 109

Введение

В настоящее время растет потребность к различным информационным сервисам. Это связано с тем, что информационные технологии занимают все большее место в жизни человека. Цифровизация помогает перевести общественные отношения на качественно другой уровень, однако вместе с тем поднимает вопросы информационной безопасности.

Значимость и приоритетность развития информационных технологий в Российской Федерации подтверждается принятием в 2017 году Государственной программы под названием «Цифровая экономика Российской Федерации».

Для разработки и реализации различных мероприятий программа определяет такие основополагающие принципы информационной безопасности, как: «использование российских технологий обеспечения целостности, аутентификации и доступности передаваемой информации», «преимущественное использование отечественного программного обеспечения и оборудования» и «применение технологий защиты информации с использованием российских криптографических стандартов» [1].

Помимо Государственной программы следует упомянуть Указ Президента Российской Федерации от 9 мая 2017 г. № 203 «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы», в котором сказано, что «Для устойчивого функционирования информационной инфраструктуры Российской Федерации необходимо … обеспечить использование российских криптоалгоритмов и средств шифрования при электронном взаимодействии федеральных органов» [2].

Безопасность является одним из главных принципов любой информационной системы. Данный принцип считается сложным, так как он должен распространяться на всю систему. Любой дефект или недочет в безопасности может сделать всю систему небезопасной.

Использование всех механизмов безопасности нецелесообразно. Поэтому необходимо сначала дать описание требований безопасности.

Политика безопасности описывает, какие действия разрешено выполнять объектам в системе, а какие нет [3]. Важными механизмами безопасности, с помощью которых можно применять политику, являются [3]:

– шифрование;
– аутентификация;
– авторизация;
– аудит.

В данной работе будут рассмотрены вопросы аутентификации. Аутентификация является процессом проверки и подтверждения заявленной личности пользователя, клиента или другого объекта [4].

В информационных системах использование криптографических методов в механизмах безопасности является основой. Существуют различные криптографические системы, в зависимости от того, используются одинаковые ключи шифрования и дешифрования или нет [3].

В симметричных криптосистемах используется общий секретный ключ, который должен храниться в секрете для обеспечения работы защиты. В асимметричных криптосистемах (система с открытым ключом) один ключ является открытым, а другой хранится в секрете.

Безопасность играет важную роль в информационных системах. Эффективность всей системы контроля доступа зависит от того, насколько хорошо и надежно аутентифицирован пользователь. Аутентификация является основой для создания доверительных отношений при удалённом взаимодействии и должна быть одним из самых важных доверенных сервисов при формировании единого пространства доверия [4].

Актуальность темы обусловлена ростом ценности информации, постоянным появлением новых угроз информационной безопасности и важностью процесса аутентификации при построении защиты информационной системы. Следует отметить, что тема выпускной квалификационной работы по разработке программных средств аутентификации является актуальной и перспективной.

1 Анализ предметной области и требований

1.1 Анализ предметной области

В любой информационной системе процесс регистрации пользователей заключается в прохождении трех взаимоствязанных этапов: идентификация, аутентификация и авторизация [5]. Дадим определения этим терминам.

Идентификация – процедура распознавания субъекта по его идентификатору.

Аутентификация – процедура проверки подлинности субъекта, которая позволяет достоверно убедиться в том, что субъект, предъявивший свой идентификатор, на самом деле является именно тем субъектом, идентификатор которого он использует.

Авторизация – процедура предоставления субъекту определенных прав доступа к ресурсам системы после прохождения им процедуры аутентификации.

В данной работе будет описан механизм аутентификации пользователей.

Учитывая степень доверия и политику безопасности систем, проводимая проверка подлинности может быть односторонней или взаимной (двусторонней). Обычно она проводится с помощью криптографических способов.

Для повышенной безопасности применяют двустороннюю аутентификацию.

Основными программно-техническими способами реализации аутентификации являются [6]:

– пароли;
– с использованием хеш-функции;
– на основе шифрования с открытым ключом;
– сервер аутентификации Kerberos;
– биометрия;
– идентификационные карты и электронные ключи.

Парольная аутентификация включает в себя проверку введенного

пользователем пароля с тем, что хранится в БД. Если они совпадают, то пользователю предоставляется доступ к системе и ресурсам.

Современная криптография разделяет криптографические методы на четыре крупных класса [7]:

– симметричные криптосистемы;
– криптосистемы с открытым ключом;
– системы электронной цифровой подписи (ЭЦП);
– системы управления ключами.

В симметричных криптосистемах для шифрования и дешифрования используется один и тот же ключ. То есть, чтобы начать использовать систему, наобходимо получить общий секретный ключ, но необходимо исключить к нему доступ злоумышленника.

В ассиметричных системах используются открытый и закрытый (секретный) ключи, которые математически связаны друг с другом. Но установить один ключ, зная другой, с вычислительной точки зрения сложно.

В зависимости от политики безопасности систем и уровня доверия различают два вида аутентификации:

– односторонняя, когда только пользователь доказывает свою подлинность;
– двусторонняя (взаимная), когда проверяется подлинность как пользователя, так и сервера.

Двусторонняя аутентификация используется криптографические способы.

Аутентификация является одним из важнейших условий для организации защищенного канала связи между клиентом и сервером, контроля целостности и шифрования данных информационного обмена. На сегодняшний день существует ряд решений обеспечения на основе практического использования криптографических средств аутентификации.

Рассмотрим аутентификацию на основе общего секретного ключа. Протокол использует общий подход, называемый «вызов-ответ» [3]. То есть

одна из сторон запрашивает у другой ответ, который может быть правильным, только если другая сторона знает общий секретный ключ.

В данной системе одним и тем же секретным ключом осуществляется и шифрование, и дешифрование [8]:

$$E_k(P) = C$$

$$D_k(C) = P,$$

где E – функция шифрования;

k – ключ;

P – открытый текст;

C – шифротекст;

D – функция дешифрования.

При этом справедливо следующее равенство:

$$D_k(E_k(P)) = P.$$

Существует несколько схем аутентификации, которые отличаются по уровню безопасности. Рассмотрим их последовательно.

Одной из широко используемых систем на основе общих секретных ключей является Kerberos [3].

Kerberos представляет собой средство проверки личности участников (например, пользователь рабочей станции или сервер) на незащищенной сети [9]. Kerberos выполняет аутентификацию как доверенная сторонняя служба аутентификации, используя обычную криптографию (с общим секретным ключом).

Базовый процесс аутентификации Kerberos происходит следующим образом [9]. Сначала клиент отправляет запрос на сервер аутентификации (AS) на получение «учетных данных» для данного сервера. AS отвечает этими учетными данными, зашифрованными в клиентском ключе.

Учетные данные состоят из «билета» для сервера и временного ключа шифрования (часто называется «сеансовым ключом»). Клиент передает билет (который содержит личность клиента и копию сеансового ключа, все зашифрованы в ключе сервера) на сервер.

Ключ сеанса (сейчас разделяется клиентом и сервером) используется для

аутентификации клиента и может дополнительно использоваться для аутентификации сервера. Это также может использоваться для шифрования дальнейшего обмена данными между двумя сторонами или для обмена отдельным ключом подсессии, который будет использоваться для дальнейшего шифрования коммуникация.

Kerberos основана на протоколе аутентификации Нидхэма-Шредера. Протокол Нидхэма-Шрёдера предназначен для решения проблемы аутентификации. Алгоритм предназначен для организации аутентифицированного канала между разными ЭВМ в сети по схеме точка-точка. Задача решается с помощью одного или двух серверов аутентификации с использованием общедоступных или общих секретных ключей. Данный протокол предоставляет децентрализованную услугу аутентификации [10].

NTLM это протокол проверки подлинности запроса и ответа, который использует три сообщения для аутентификации клиента в среде, ориентированнаой на соединение [11]. Данный протокол основан на подходе, при котором пароль не передается в чистом виде.

Negotiate это схема из семейства Windows authentication, с помощью которой клиент может выбрать между двумя протоколами, NTLM и Kerberos [12]. Так как Kerberos считается более безопасным протоколом, предпочтение отдается ему. Однако он может функционировать, только если клиент и сервер являются частью домена Windows.

Протоколы NTLM и Kerberos являются двусторонними видами аутентификации.

Одними из самых простых и небезопасных схем аутентификации являются Basic и Digest.

При использовании схемы Basic логин и пароль пользователя передаются на сервер в незашифрованном виде [12]. Они передаются в заголовке Authorization в виде base64 кода. Относительно безопасной данная схема может считаться только при использовании HTTPS протокола.

HTTPS является протоколом безопасной передачи данных.

Аутентификация Digest является более безопасной версией схемы Basic.

Основное отличие в том, что пароль не передается в открытом виде, а шифруется с помощью алгоритма хеширования MD5 [12]. Однако основным недостатком является неприменимость современных хэш-функций для хранения паролей на сервере.

Интерфейс поставщика поддержки безопасности (SSPI) позволяет приложению использовать различные модели безопасности, доступные на компьютере или в сети, не изменяя интерфейс системы безопасности [13].

Поставщик поддержки безопасности (SSP) содержится в библиотеке динамической компоновки DLL, которая реализует SSPI, делая один или несколько пакетов безопасности доступными для приложений.

Актуальность выбранной темы подчеркивается тем, что любая информационная система нуждается в надежной защите данных. Основной целью выпускной квалификационной работы стала разработка собственноых программных средств аутентификации в браузере для опрационной системы QP OS, которые будут удовлетворять принципам безопасности.

1.2 Анализ программного обеспечения аналогичного назначения

Данное разрабатываемое средство является достаточно популярным средством для аутентификации пользователей на различных ресурсах. Существует множество аналогов двусторонней аутентификации.

В настоящее время практически все браузеры поддерживают двустороннюю аутентификацию, основанную на использовании протоколов Kerberos и NTLM.

Самые популярные примеры двусторонней аутентификации – Google Chrome и Microsoft Edge. Рассмотрим их ниже.

1.2.1 Анализ аутентификации Google Chrome

На устройствах Chrome для доступа к внутренним ресурсам используется система единого входа Kerberos (SSO). Идея состоит в том, что вы входите один раз под своей учетной записью, далее все сервисы, на которых настроена SSO, доступны без ввода логина и пароля [14].

Можно выделить некоторые достоинства данного типа аутентификации:

– удобство для пользователей;

– более высокая безопасность;

– простота доступа к другим сервисам.

Но несмотря на все удобство, существует ряд недостатков:

– если злоумышленник получит данные учетной записи пользователя, то может быть получен доступ ко всем ресурсам, подключенным к SSO;

– если KDC (центр распространения ключей) недоступен, никто не сможет получить доступ к ресурсам;

– Kerberos требует, чтобы системные часы на всех клиентах и серверах были синхронизированы, в случае с большим количеством ресурсов, могут быть сбои.

1.2.2 Анализ аутентификации Microsoft Edge

Браузер Microsoft Edge поддерживает несколько аутентифицированных функций. Рассмотрим некоторые из них.

Для настройки автоматического входа на веб-сайты для пользователей можно воспользоваться опцией единого входа.

Microsoft Edge имеет встроенную поддержку единого входа на основе PRT. Первичный токен обновления (PRT) – это ключ Azure AD, который используется для аутентификации на устройствах Windows 10, iOS и Android. Он активирует функцию единого входа (SSO) в приложениях, используемых на этих устройствах [15].

Azure Active Directory (Azure AD) – это облачная служба управления удостоверениями и доступом от корпорации Майкрософт [22].

Браузер Microsoft Edge имеет встроенную поддержку бесшовного SSO без необходимости расширений [15]. Единый вход автоматически регистрирует пользователей, когда они находятся на устройствах, подключенных к общей сети.

Microsoft Edge также поддерживает встроенную проверку подлинности

Windows (WIA) для запросов проверки подлинности во внутренней сети организации для любого приложения, которое использует браузер для своей проверки подлинности.

Когда веб-сайт пытается аутентифицировать пользователей, используя механизм Negotiate, и SSO недоступен, пользователем предлагается возможность выполнить аутентификацию с помощью Windows Hello Cred UI.

К достоинствам проверки подлинности Microsoft Edge можно отнести:

– наличие нескольких способов аутентификации;

– простота доступа к другим ресурсам.

Но механизмы безопасности браузера Microsoft Edge имеет ряд недостатков:

– может быть получен доступ ко всем ресурсам, подключенным к SSO;

– для использования некоторых способов аутентификации необходимо выполнить дополнительную настройку браузера.

1.2.3 Результат анализа программ аналогов

По результатам анализа можно следать вывод, что приведенные выше аналоги имеют ряд недостатков перед разрабатываемым средством.

Одним из главных недостатков аутентификации Google Chrome является то, что используется только один протокол безопасности Kerberos и при его сбое пользователь не сможет получить доступ к ресурсу.

Также разрабатываемые программные средства предназначены для наилучшей защиты всех ресурсов от атак. Если злоумышленник получит учетный данные, он сможет получить доступ до всех ресурсов, что противоречит политике безопасности.

Средство аутентификации Microsoft Edge удовлетворяет части требований. Первый недостаток Google Chrome также относится к данному средству.

Также следует учесть, что данные средства аутентификации не могут быть адаптированы для использования под управлением операционной

системы QP OS, а также не имеют поддержки российских криптографических алгоритмов.

Таким образом, было принято решение в ходе данной работы реализовать собственные программные средства аутентификаци пользователей в браузере для операционной системы QP OS.

1.3 Анализ требований

Анализ требований – это процесс сбора требований к программному обеспечению (ПО), их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения [15].

Выделяют функциональные и нефункциональные требования.

Функциональные требования – это перечень сервисов, которые должна выполнять система, причём должно быть указано, как система реагирует на те или иные входные данные, как она ведёт себя в определённых ситуациях и т.д. В некоторых случаях указывается, что система не должна делать.

Нефункциональные требования описывают характеристики системы и её окружения, а не поведение системы. Также может быть приведён перечень ограничений, накладываемых на действия и функции, выполняемые системой. Они включают временные ограничения, ограничения на процесс разработки системы, стандарты и т.д. [16].

1.3.1 Анализ функциональных требований

Основной задачей разрабатываемой двусторонней аутентификации является аутентификация пользователей браузером в QP OS.

Были выделены функции, которые должны быть реализованы в браузере для данного метода аутентификации пользователей:

- выдача окна для ввода логина и пароля пользователя;
- проверка корректности введенных данных;
- выдача сообщения об отказе в доступе (пользователь не найден);
- выбор схемы аутентификации;

– возможность аутентификации с помощью протокола Kerberos;

– возможность аутентификации с помощью протокола NTLM;

– возможность аутентификации с помощью протокола Digest;

– возможность аутентификации с помощью протокола Basic;

– выдача страницы запрашиваемого ресурса (при разрешении доступа).

На рисунке 1 представлена диаграмма вариантов использования, разработанная на основе выделенных требований, описывающая функциональное назначение системы.

Разработка всех диаграмм UML будет проводиться с помощью средства моделирования «ArgoUML».

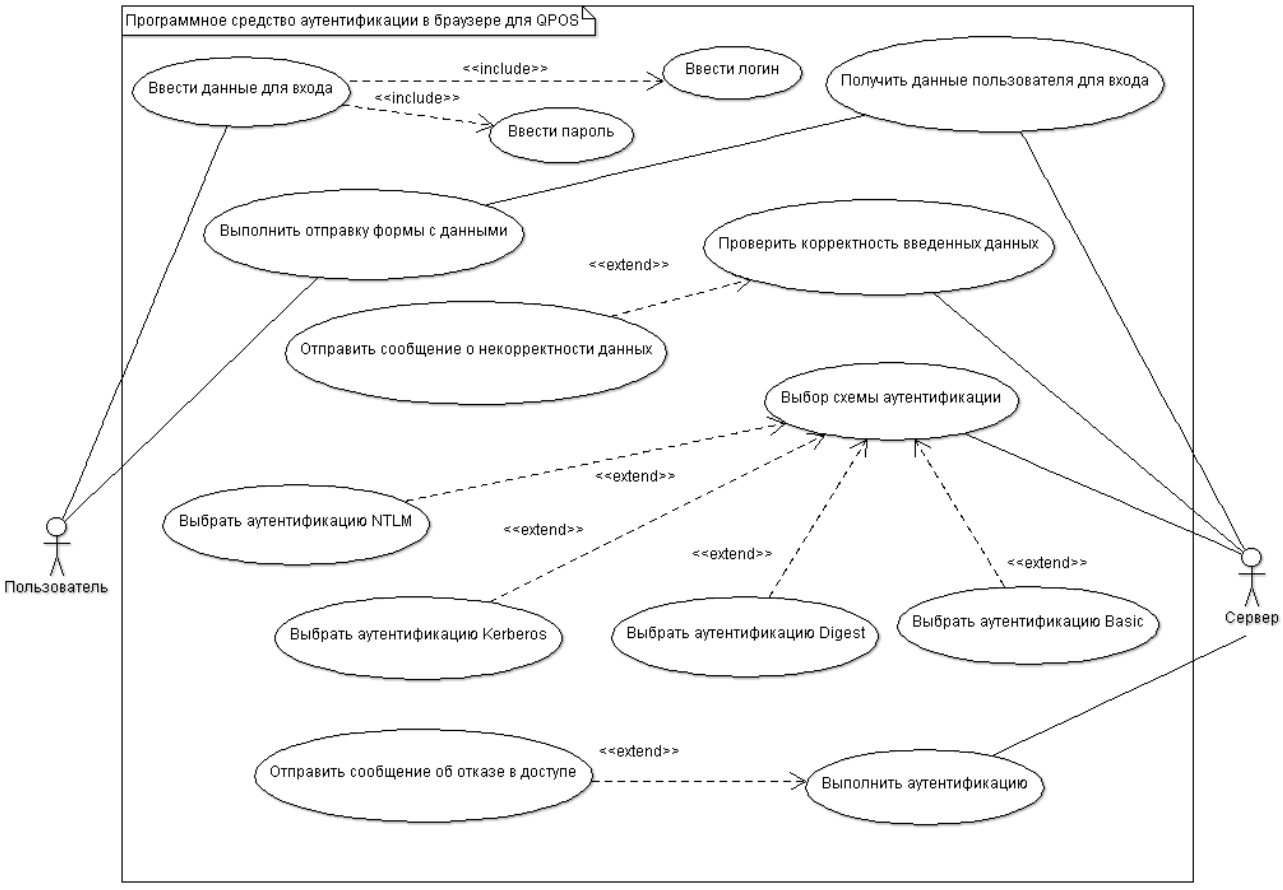

Рисунок 1 – Диаграмма вариантов использования программного средства аутентификации

В ходе проектирования системы было выделено 2 сущности (актера):

– пользователь вводит данные для входа в специальную форму и

выполняет отправку;

– сервер выполняет обработку запросов (проверка корректности введеных данных и т.д.).

Для того, чтобы более наглядно представить функционирование прецентдентов были описаны спецификации прецендентов «Проверить корректность введенных данных» и «Выбор схемы аутентификации».

Спецификакия прецендента «Проверить корректность введенных данных» представлена на рисунке 2.

Прецедент: Проверить корректность введенных данных
ID:
Краткое описание: Сервер проверяет корректность введенных данных пользователя для входа
Главные актёры: Сервер
Второстепенные актёры: Пользователь
Предусловия: 1. Пользователь ввел данные для входа. 2. Пользователь выполнил отправку формы с данными для входа. 3. Сервер получил отправленные пользователем данные.
Основной поток: 1. Сервер выполняет проверку логина на пустоту 2. Если логин не пустой 2.1 Сервер выполняет проверку на наличие недопустимых символов 2.2 Если таких символов нет 2.2.1 Сервер выполняет проверку пароля на пустоту 2.2.2 Если пароль не пустой 2.2.2.1 Сервер выполняет проверку на наличие недопустимых символов 2.2.2.2 Если таких символов нет 2.2.2.2.1Сервер отправляет данные для проверки их наличия в БД 3. Иначе сервер отправляет сообщение о некорректности введенных данных
Постусловия: 1. Проверка данных на корректность произведена
Альтернативные потоки: Нет.

Рисунок 2 – Спецификация прецедента «Проверить корректность введенных данных»

Спецификакия прецендента «Выбор схемы аутентификации» представлена на рисунке 3.

Прецедент: Выбор схемы аутентификации
ID:
Краткое описание: Сервер выбирает подходящую схему аутентификации пользователя
Главные актёры: Сервер
Второстепенные актёры: Пользователь
Предусловия: 1. Сервер получил учетные данные пользователя.
Основной поток: 1. Сервер пытается выполнить аутентификацию с помощью протокола Kerberos 2. Если невозможно использовать Kerberos 2.1 Сервер пытается выполнить аутентификацию с помощью протокола NTLM 2.2 Если невозможно использовать NTLM 2.2.1 Сервер пытается выполнить аутентификацию с помощью протокола Digest 2.2.2 Если невозможно использовать Digest 2.2.2.1 Сервер пытается выполнить аутентификацию с помощью протокола Basic 3. Сервер выполняет аутентификацию с помощью выбранной схемы
Постусловия: 1. Выполнена аутентификация пользователя с помощью выбранной схемы
Альтернативные потоки: Нет.

Рисунок 3 – Спецификация прецедента «Выбор схемы аутентификации»

Более подробное описание функционирования прецедентов можно представить в виде диаграммы деятельности. Данный вид UML-диаграммы показывает последовательность действий, которая необходима для достижения конечной цели рассматриваемого прецедента. На рисунке 4 представлена диаграмма деятельности для варианта использования «Выбор схемы аутентификации».

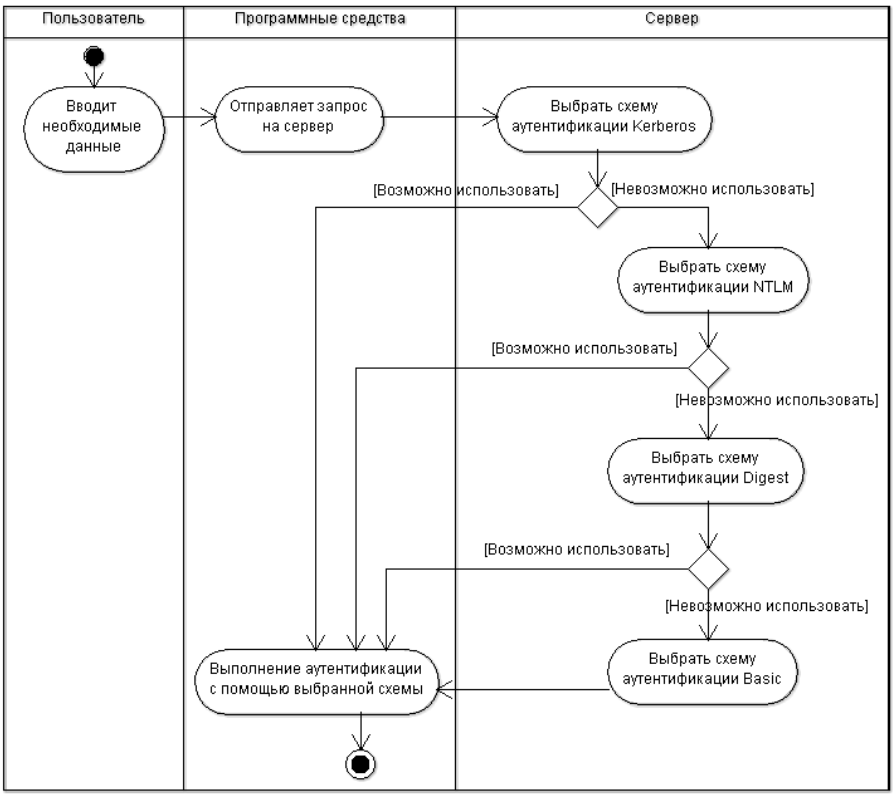

Рисунок 4 – Диаграмма деятельности для прецедента «Выбор схемы аутентификации»

На рисунке 5 представлена диаграмма последовательности для прецедента «Проверить корректность введенных данных».

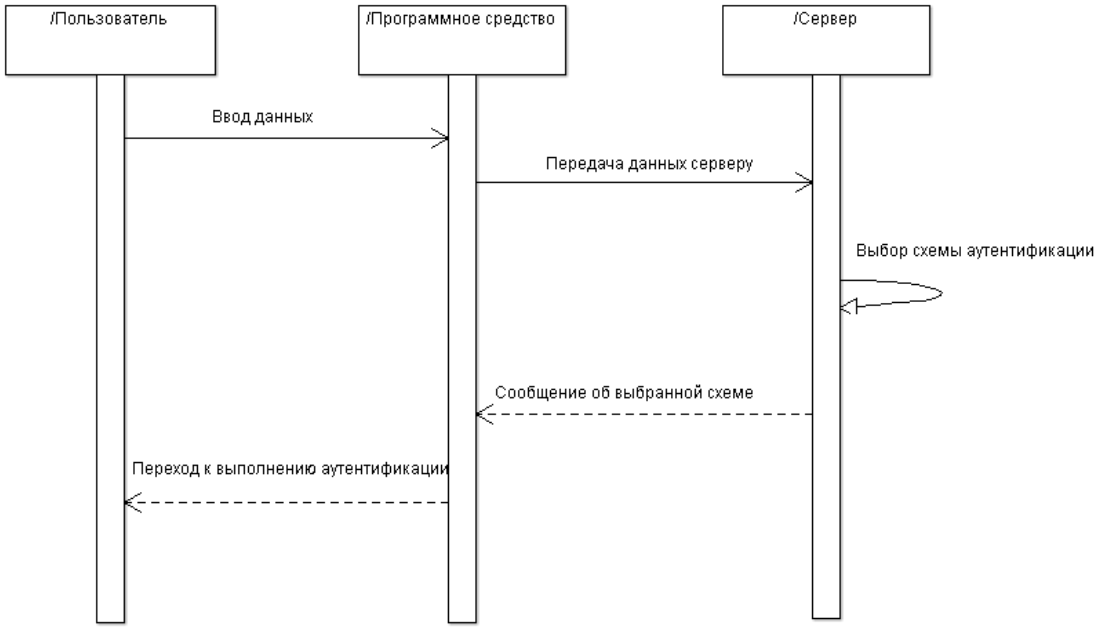

Рисунок 5 – Диаграмма последовательности для прецедента «Выбор схемы аутентификации»

Данная диаграмма отображает более подробное взаимодействие объектов во времени, а также показывает обмен сообщениями между объектами в рамках рассматриваемого сценария.

1.3.2 Анализ нефункциональных требований

К разрабатываемому программному средству был предъявлен ряд нефункциональных требований.

Данные программные средства аутентификации должны работать в браузере специального назначения под управлением операционной системы QP OS. Это основное требование к разработке.

Программные средства должны быть разработаны на языке программирования C# в среде разработки Microsoft Visual Studio 2019, который позволяет создавать безопасные и надежные приложения. Разработка проводилась под операционной системой Windows. Также он содержит множество компонентов, необходимых для использования при разработке данных программных средств.

Также выделим некоторые требования к клиентской части:

– интерфейс должен быть удобным и понятным для пользователя;

– исключить возникновение ситуаций, приводящих к аварийному завершению работы.

Необходимо обеспечить надежность функционирования программного средства.

1.4 Планирование разработки и оценки бюджета

Для оценки стоимости и трудоемкости проекта необходимо провести правильное планирование. Оценка бюджета – это распределение статей доходов и расходов [17]. Для этого были определены задачи и ресурсы, которые необходимы для создания программного средства.

Ниже представлены задачи, которые представляют этапы создания проекта (таблица 1).

Таблица 1 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	13 дней	-
3	Анализ аналогичных решений	7 дней	1
4	Анализ функциональных требований	4 дня	3
5	Принятие окончательного решения	2 дня	4
6	Анализ требований завершен	-	5
7	Составление ТЗ	11 дней	-
8	Формулирование целей и задач	4 дня	6
9	Оформление ТЗ	5 дней	8
10	Согласование ТЗ с руководством	2 дня	9
11	Составление ТЗ закончено	-	10
12	Реализация проекта	43 дня	-
13	Разработка пользовательского интерфейса	10 дней	11
14	Разработка структуры данных	2 дня	11
15	Разработка модулей обработки данных	1 день	14; 13
16	Разработка первого типа аутентификации	15 дней	15
17	Разработка второго типа аутентификации	15 дней	16
18	Реализация проекта завершена	-	17
19	Отладка	18 дней	-
20	Тестирование пользовательского интерфейса	7 дней	18
21	Тестирование первого типа аутентификации	7 дней	18
22	Тестирование второго типа аутентификации	8 дней	18
23	Исправление ошибок	10 дней	20; 21; 22
24	Отладка завершена	-	23
25	Обучение пользователей	7 дней	-
26	Принципы работы программы	3 дня	24
27	Изучение интерфейса	4 дня	26
28	Обучение завершено	-	27
29	Передача в эксплуатацию	5 дней	-
30	Акт ввода в эксплуатацию	2 дня	28
31	Передача в эксплуатацию	3 дня	30
32	Передача в эксплуатацию завершена	-	31
33	Конец проекта	-	32

Из таблицы 1 видно, что для реализации проекта необходимо выполнить 33 задачи.

В таблице 2 представлено описание затрачиваемых ресурсов.

Таблица 2 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	30000 Р/месяц	300 Р/ч
Программист2	Трудовой	50000 Р/месяц	350 Р/ч
Начальник отдела	Трудовой	60000 Р/месяц	400 Р/ч
Компьютер	Материальный	500 Р	

Распределение ресурсов по задачам представлено в таблице 3.

Таблица 3 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Анализ аналогичных решений	Программист 2; Компьютер	100% 1 шт.
4	Анализ функциональных требований	Программист 2; Компьютер	100% 1 шт.
5	Принятие окончательного решения	Начальник отдела; Программист 1; Программист 2;	50% 25% 25%
6	Анализ требований завершен		
7	Составление ТЗ		
8	Формулирование целей и задач	Начальник отдела; Программист 2;	50% 50%
9	Оформление ТЗ	Начальник отдела; Программист 1; Программист 2; Компьютер	50% 25% 25% 3 шт.
10	Согласование ТЗ с руководством	Начальник отдела;	100%
11	Составление ТЗ закончено		
12	Реализация проекта		
13	Разработка пользовательского интерфейса	Программист 1; Компьютер	100% 1 шт.
14	Разработка структуры данных	Программист 2; Компьютер	100% 1 шт.
15	Разработка модулей обработки данных	Программист 2; Компьютер	100% 1 шт.
16	Разработка первого типа аутентификации	Программист 1; Программист 2; Компьютер	50% 50% 2 шт.

Продолжение таблицы 3

№	Название задачи	Ресурс	Единицы
17	Разработка второго типа аутентификации	Программист 1; Программист 2; Компьютер	50% 50% 2 шт.
18	Реализация проекта завершена		
19	Отладка		
20	Тестирование пользовательского интерфейса	Программист 1; Компьютер	100% 1 шт.
21	Тестирование первого типа аутентификации	Программист 1; Программист 2; Компьютер	50% 50% 2 шт.
22	Тестирование второго типа аутентификации	Программист 1; Программист 2; Компьютер	50% 50% 2 шт.
23	Исправление ошибок	Программист 1; Программист 2; Компьютер	50% 50% 2 шт.
24	Отладка завершена		
25	Обучение пользователей		
26	Принципы работы программного средства	Начальник отдела; Программист 1; Компьютер	50% 50% 1 шт.
27	Изучение интерфейса	Начальник отдела; Программист 1; Компьютер	50% 50% 1 шт.
28	Обучение завершено		
29	Передача в эксплуатацию		
30	Акт ввода в эксплуатацию	Начальник отдела; Компьютер	100% 1 шт.
31	Передача в эксплуатацию	Начальник отдела; Программист 1; Программист 2; Компьютер	50% 25% 25% 1 шт.
32	Передача в эксплуатацию завершена		
33	Конец проекта		

Из представленных таблиц видно, что реализацией проекта будут заниматься 2 программиста и начальник отдела, которым для работы нужны компьютеры.

Для вычисления длительности реализации проекта и его критического проекта построим сетевой график. Он представлен на рисунке 6.

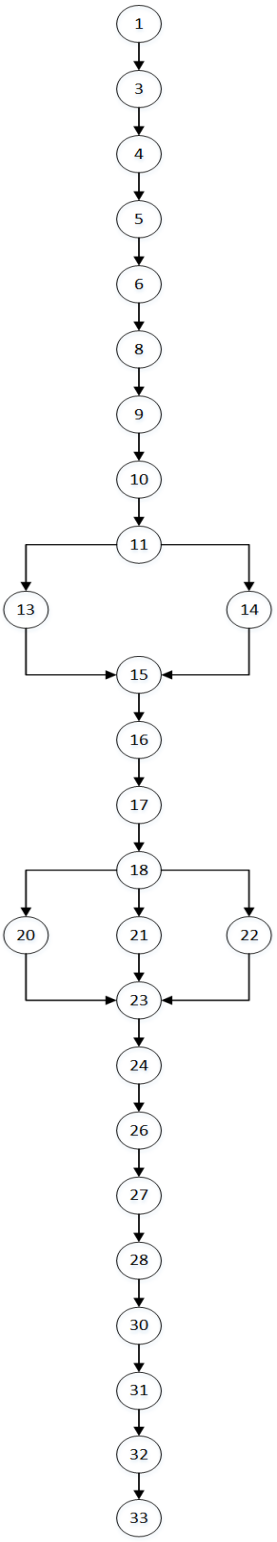

Рисунок 6 – Сетевой график проекта

Критические работы имеют одинаковые значения раннего и позднего времени. Формула имеет вид:

$$T_p(i) = \max\left(T_p(j) + t_j\right)\ j \in G,$$

где G – множество работ, непосредственно предшествующих работе i;

t_i – время выполнения работы i;

$T_p(i)$ – раннее время начала работы i;

$T_{\text{п}}(i)$ – позднее время начала работы i.

Раннее время начальной работы проекта принимается равным нулю.

Так как последняя работа – это веха с нулевой длительностью, то раннее время ее начала совпадает с длительностью всего проекта:

$$T_{\text{п}}(i) = \min\left(T_{\text{п}}(j) - t_j\right)\ j \in H,$$

где H – это множество работ, непосредственно следующих за работой i.

Вычисление раннего времени начала работ приведено в таблице 4.

Таблица 4 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	7	0+7	7
5	4	7	4	7+4	11
6	5	11	2	11+2	13
8	6	13	0	13+0	13
9	8	13	4	13+4	17
10	9	17	5	17+5	22
11	10	22	2	22+2	24
13	11	24	0	24+0	24
14	11	24	0	24+0	24
15	13 14	24 24	10 2	Max(24+10;24+2)	34
16	15	34	1	34+1	35
17	16	35	15	35+15	50
18	17	50	15	50+15	65
20	18	65	0	65+0	65
21	18	65	0	65+0	65
22	18	65	0	65+0	65
23	20 21 22	65 65 65	7 7 8	Max(65+7;65+7;65+8)	73
24	23	73	10	73+10	83
26	24	83	0	83+0	83
27	26	83	3	83+3	86
28	27	86	4	86+4	90
30	28	90	0	90+0	90
31	30	90	2	90+2	92

Продолжение таблицы 4

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
32	31	92	3	92+3	95
33	32	95	0	95+0	95

Из результатов представленной выше таблицы видно, что выполнение проекта должно занять 95 рабочих дней. Вычисление позднего времени начала работ приведено в таблице 5.

Таблица 5 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Раннее время начала
	№	Позднее время начала	Длительность		
33	-	-	-	-	95
32	33	95	0	95-0	95
31	32	95	3	95-3	92
30	31	92	2	92-2	90
28	30	90	0	90-0	90
27	28	90	4	90-4	86
26	27	86	3	86-3	83
24	26	83	0	83-0	83
23	24	83	10	83-10	73
22	23	73	8	73-8	65
21	23	73	7	73-7	66
20	23	73	7	73-7	66
18	22 21 20	65 66 66	0 0 0	Min(65-0;66-0;66-0)	65
17	18	65	15	65-15	50
16	17	50	15	50-15	35
15	16	35	1	35-1	34
14	15	34	2	34-2	32
13	15	34	10	34-10	24
11	14 13	32 24	0 0	Min(32-0;24-0)	24
10	11	24	2	24-2	22
9	10	22	5	22-5	17
8	9	17	4	17-4	13
6	8	13	0	13-0	13
5	6	13	2	13-2	11
4	5	11	4	11-4	7
3	4	7	7	7-7	0
1	3	0	0	0-0	0

Исходя из найденных значений раннего и позднего начала работ можно найти критический путь. Все вычисления представлены в таблице 6.

Таблица 6 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	7	7	0
5	11	11	0
6	13	13	0
8	13	13	0
9	17	17	0
10	22	22	0
11	24	24	0
13	24	24	0
14	24	32	8
15	34	34	0
16	35	35	0
17	50	50	0
18	65	65	3
20	65	66	1
21	65	66	1
22	65	65	0
23	73	73	0
24	83	83	0
26	83	83	0
27	86	86	0
28	90	90	0
30	90	90	0
31	92	92	0
32	95	95	0
33	95	95	0

Microsoft Project – программа управления проектами, который предназначен для долгосрочной стратегии [18].

На рисунке 7 представлен граф проекта с указанием штриховыми линиями критического пути.

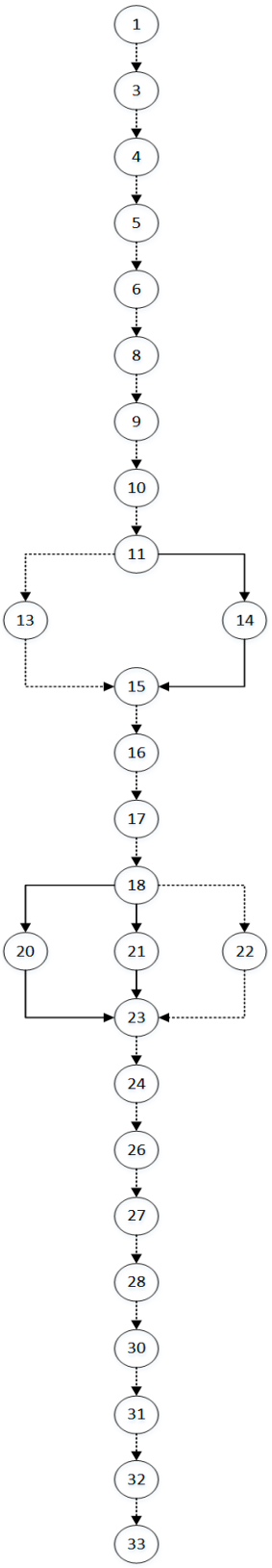

Рисунок 7 – Критический путь проекта

На рисунке 8 изображено введенное в систему управления проектами Microsoft Project содержание таблицы 1 с вычисленными датами начала и завершения каждой задачи. Началом проекта была принята дата 11.01.2021.

	Название задачи	Длительность	Начало	Окончание	Предшественники
1	Начало проекта	0 дней	Пн 11.01.21	Пн 11.01.21	
2	**Анализ требований**	**13 дней**	**Пн 11.01.21**	**Ср 27.01.21**	
3	Анализ аналогичных решений	7 дней	Пн 11.01.21	Вт 19.01.21	1
4	Анализ функциональных требований	4 дней	Ср 20.01.21	Пн 25.01.21	3
5	Приняте окончательного решения	2 дней	Вт 26.01.21	Ср 27.01.21	4
6	Анализ требований завершен	0 дней	Ср 27.01.21	Ср 27.01.21	5
7	**Составление ТЗ**	**11 дней**	**Чт 28.01.21**	**Чт 11.02.21**	
8	Формулирование целей и задач	4 дней	Чт 28.01.21	Вт 02.02.21	6
9	Оформление ТЗ	5 дней	Ср 03.02.21	Вт 09.02.21	8
10	Согласование ТЗ с руководством	2 дней	Ср 10.02.21	Чт 11.02.21	9
11	Составление ТЗ закончено	0 дней	Чт 11.02.21	Чт 11.02.21	10
12	**Реализация проекта**	**40,9 дней**	**Пт 12.02.21**	**Пт 09.04.21**	
13	Разработка пользовательского интерфейса	10 дней	Пт 12.02.21	Чт 25.02.21	11
14	Разработка структуры данных	2 дней	Пт 12.02.21	Пн 15.02.21	11
15	Разработка модулей обработки данных	0,9 дней	Пт 26.02.21	Пт 26.02.21	14;13
16	Разработка первого типа аутентификации	15 дней	Пт 26.02.21	Пт 19.03.21	15
17	Разработка второго типа аутентификации	15 дней	Пт 19.03.21	Пт 09.04.21	16
18	Реализация проекта завершена	0 дней	Пт 09.04.21	Пт 09.04.21	17
19	**Отладка**	**18 дней**	**Пт 09.04.21**	**Ср 05.05.21**	
20	Тестирование пользовательского интерфейса	7 дней	Пт 09.04.21	Вт 20.04.21	18
21	Тестирование первого типа аутентификации	7 дней	Пт 09.04.21	Вт 20.04.21	18
22	Тестирование второго типа аутентификации	8 дней	Пт 09.04.21	Ср 21.04.21	18
23	Исправление ошибок	10 дней	Ср 21.04.21	Ср 05.05.21	20;21;22
24	Отладка завершена	0 дней	Ср 05.05.21	Ср 05.05.21	23
25	**Обучение пользователей**	**7 дней**	**Ср 05.05.21**	**Пт 14.05.21**	
26	Принципы работы программного средства	3 дней	Ср 05.05.21	Пн 10.05.21	24
27	Изучение интерфейса	4 дней	Пн 10.05.21	Пт 14.05.21	26
28	Обучение завершено	0 дней	Пт 14.05.21	Пт 14.05.21	27
29	**Передача в эксплуатацию**	**5 дней**	**Пт 14.05.21**	**Пт 21.05.21**	
30	Акт ввода в эксплуатацию	2 дней	Пт 14.05.21	Вт 18.05.21	28
31	Передача в эксплуатацию	3 дней	Вт 18.05.21	Пт 21.05.21	30
32	Передача в эксплуатацию завершена	0 дней	Пт 21.05.21	Пт 21.05.21	31
33	Конец проекта	0 дней	Пт 21.05.21	Пт 21.05.21	32

Рисунок 8 – Задачи проекта в среде Microsoft Project

На рисунке 9 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

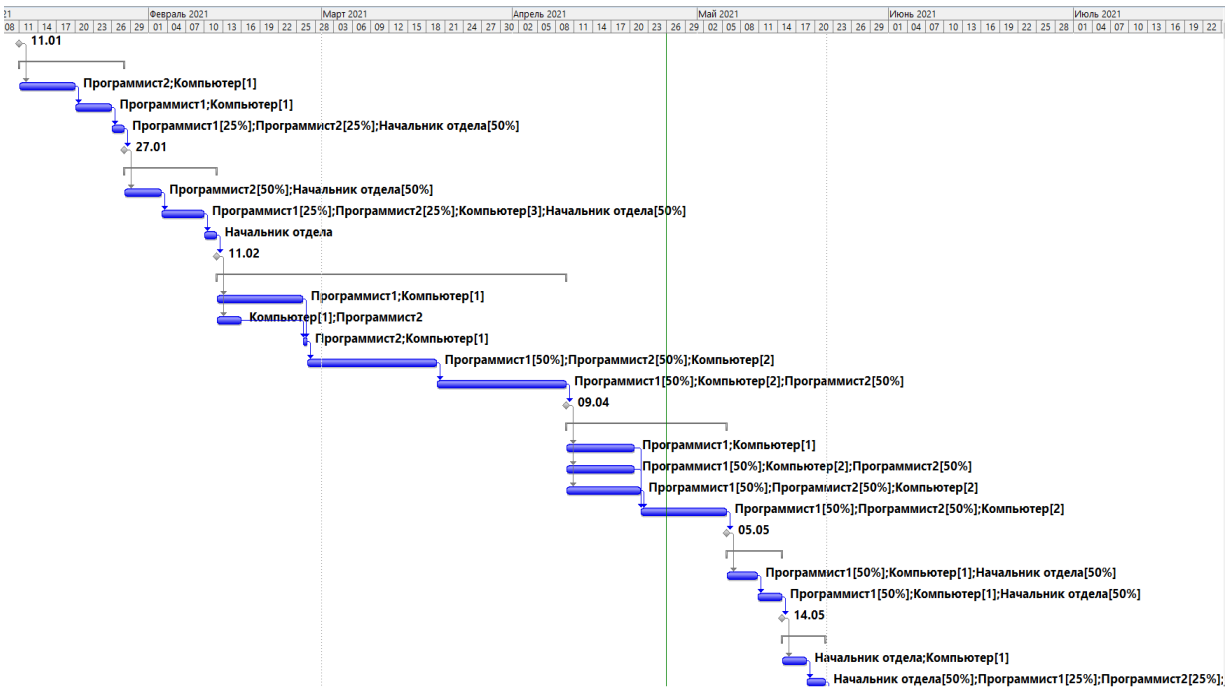

Рисунок 9 – Диаграмма Ганта

Из диаграммы видно, что вычисленный программой критический путь проходит по задачам 1, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 26, 27, 28, 30, 31, 32, 33 и составляет 95 дней.

На рисунке 10 представлен график трудового ресурса «Программист1».

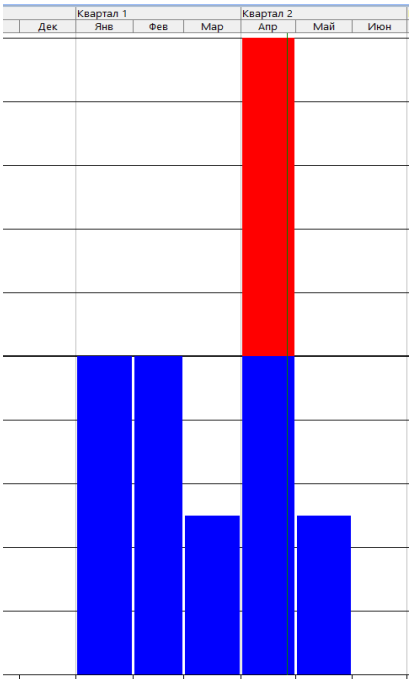

Рисунок 10 – График ресурсов для ресурса «Программист1»

На рисунке видно, что у трудового ресурса во время проведения проекта нагрузка превышает нормальную. Данная перегрузка наблюдается в апреле.

На рисунке 11 изображен отчет проекта, на котором можно увидеть подсчитанную Microsoft Project длительность реализации проекта – 95 дней и предполагаемые затраты – 216970 руб.

НИР2007
SPecialiST RePack

на дату Пн 26.04.21

Даты			
Начало:	Пн 11.01.21	Окончание:	Пт 21.05.21
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность			
Текущий план:	94,9 дней	Оставшаяся:	94,9 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	94,9 дней	Процент завершени	0%

Трудозатраты			
Текущий план:	775,52 ч	Оставшиеся:	775,52 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	775,52 ч	Процент завершени	0%

Затраты			
Текущий план:	216 970,00 ₽	Оставшиеся:	216 970,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	216 970,00 ₽		

Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	33	Трудовые ресурсы:	2
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	1
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	33	Итого ресурсов:	4

Рисунок 11 – Обзорный отчет проекта

Полученные результаты свидетельствуют об экономической целесообразности данного проекта.

2 Проектирование программных средств аутентификации в браузере для операционной системы QP OS

2.1 Проектирование архитектуры

Выделяют несколько технологий разработки программного продукта – каскадная модель, спиральная модель, технологии XP, RUP, RAD.

Данные программные средства разрабатывались при помощи методологии разработки ПО RUP, которая является одной из самых популярных методологий разработки ПО в настоящее время. Большим плюсом разработки программного обеспечения в RUP является разделение проекта на несколько мелких проектов, которые выполняются последовательно, и каждая итерация разработки четко определена набором целей, которые должны быть достигнуты в конце итерации. Таким образом, уже на стадии планирования целей и задач проекта, можно разделить команду разработчиков на конкретные мелкие проекты [19].

При разработке программных средств была выбрана двухуровневая архитектура – клиент-сервер (рисунок 12).

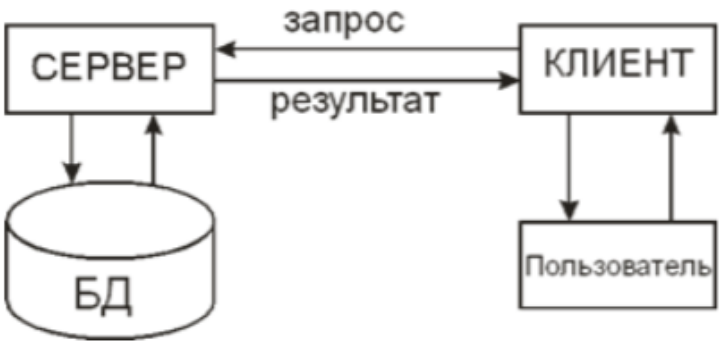

Рисунок 12 – Клиент-серверная архитектура

При таком подходе на сервере находится база данных вместе с СУБД и прикладными программами, клиентская часть содержит только уровень представления данных [20].

Согласно поставленным требованиям средства аутентификации пользователей в браузере для операционной системы QP OS представляет собой службу, имеющую пользовательской интерфейс. Его проектирование

будет описано далее. Клиенты службы взаимодействую с ней по протоколу HTTP.

С помощью интерфейса поставщика поддержки безопасности (SSPI) приложение может использовать различные модели безопасности. Все подпрограммы используют сообщения «запрос-ответ».

Во время сеанса связи между клиентом и сервером действует набор атрибутов и правил безопасности, называемый контекст безопасности.

После ввода учетных данных клиент отправляет на сервер введенные данные для входа.

Сервер инициирует исходящий контекст безопасности на стороне клиента из дескриптора учетных данных. После возвращается токен, который клиент должен передать серверу.

Сервер принимает полученный от клиента токен (ключ доступа) и устанавливает контекст безопасности. Для завершения установления контекста безопасности сервер может запросить более одного маркера ответа от клинета.

Используя полученный токен, сервер выбирает, какой поставщик службы поддержки безопасности (SSP) он может использовать. Первым уровнем он обращается к Microsoft Negotiate. Он анализирует запрос и выбирает лучший поставщик служб для выполнения запроса на основе политики безопасности. Negotiate выбирает между Kerberos и NTLM. В первую очередь будет выбран Kerberos, так как он считается более безопасным. Если приложение не предоставляет достаточных сведений для его использования, Negotiate всегда выбирает поставщика безопасности NTLM.

Если сервер не поддерживает Negotiate, выполняется протокол дайджест-доступа. Microsoft Digest это упрощенный механизм реагирования на запрос для проверки подлинности клиентов. Предназначения для служб, использующих протокол HTTP или связь на основе SASL.

При невозможности использования протокола дайджест-доступа, применяется самый простой и незащищенный протокол Basic.

При успешном завершении аутентификации, клиент получает доступ к ресурсу. Иначе сервер отказывает в доступе.

2.2 Проектирование интерфейса

На основе выявленных функциональных и нефункциональных требований был спроектирован пользовательский интерфейс. Основной страницей программного средства является страница аутентификации.

Рассмотрим элементы, расположенные на данной странице (рисунок 13).

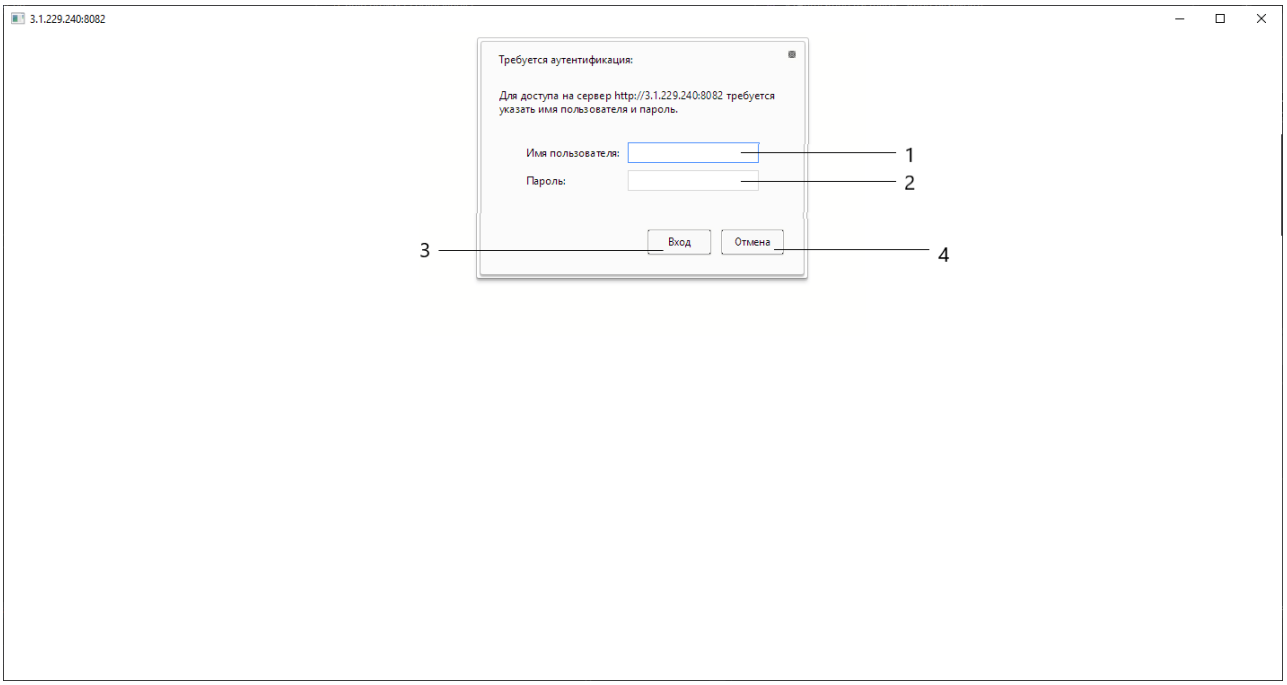

Рисунок 13 – Страница идентификация

Цифрами на рисунке 13 отмечены следующие компоненты страницы:

1 – текстовое поле для ввода логина;

2 – текстовое поле для ввода пароля;

3 – кнопка входа в качестве пользователя;

4 – кнопка отмены.

При успешном выполнении идентификация и аутентификации, пользователь получает доступ к системе (рисунок 14).

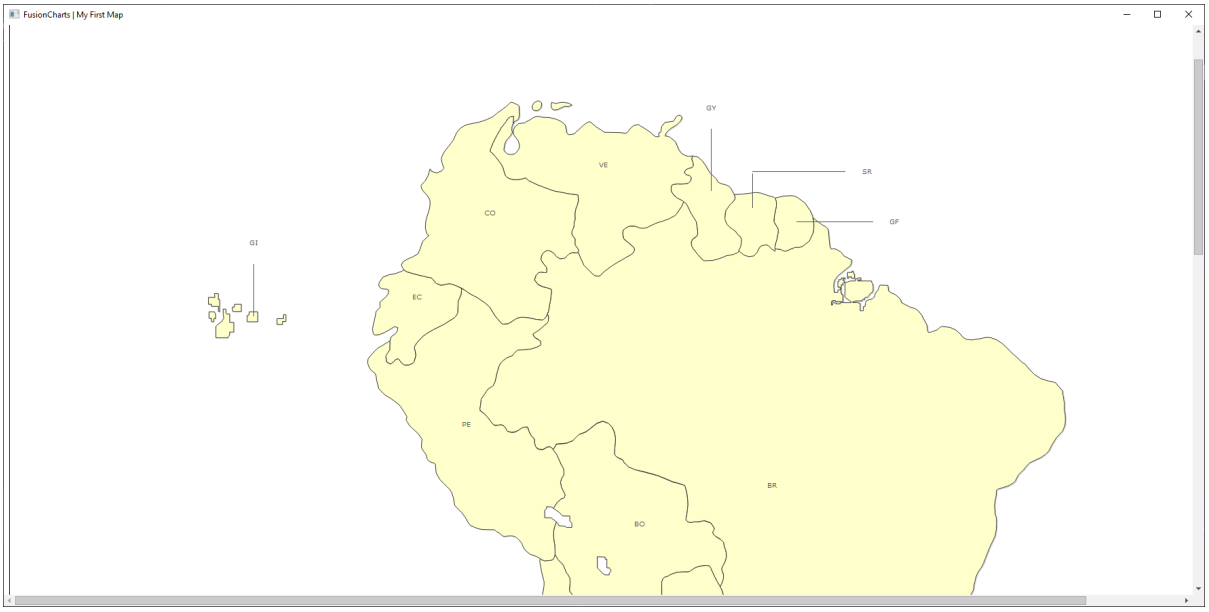

Рисунок 14 – Страница ресурса

При неуспешном выполнении идентификации и аутентификации, нажатии кнопки отмены или ввода недействительных данных, пользователь переходит на страницу отказа в доступе (рисунок 15).

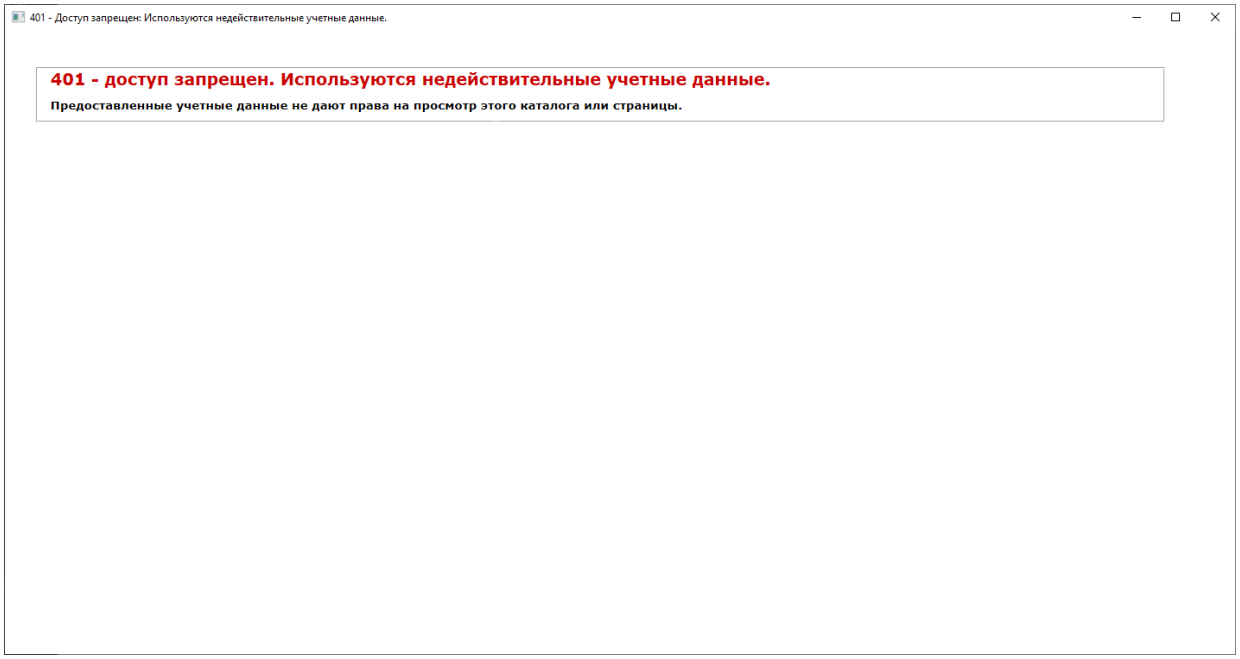

Рисунок 15 – Страница отказа в доступе

Таким образом, было проведено проектирование интерфейса программного средства. Интерфейс является простым и интуитивно понятным.

3 Реализация программных средств аутентификации в браузере для операционной системы QP OS

Результатом разработки является исполняемый файл «Titan.exe», предназначенный для работы под управлением операционной системы QP OS. Разработанные программные средства являются встроенным функционалом в браузер Titan.

Разработка программных средств аутентификации велась на языке программирования C# в среде программирования Microsoft Visual Studio 2019. Разработка проводилась в операционной системе Windows 10.

Исходный код наиболее значимых классов программных средств приведен в приложении Б.

При запуске браузера открывается окно аутентификации. При вводе логина и пароля происходит идентификация пользователя.

Учетные данные пользователя отправляются на сервер с помощью HTTP, протокола прикладного уровня передачи данных.

При успешной идентификации, начинают выполняться программные средства аутентификации.

В случае успеха пользователь получает доступ к запрашиваемому ресурсу, в ином случае пользателю отказывают в доступе.

На диаграмме классов, представленной ниже, показаны классы, реализующие работу схем аутентификации (рисунок 16).

Кратко опишем назначения представленных ниже классов.

Класс HttpAuthHandler представляет собой абстрактный класс, имеющий общие методы для классов наследников, например, инициализация вызова схемы аутентификации или завершении аутентификации. Методы, имеющие ключевое слово abstract или virtual, переопределяются в классах наследниках.

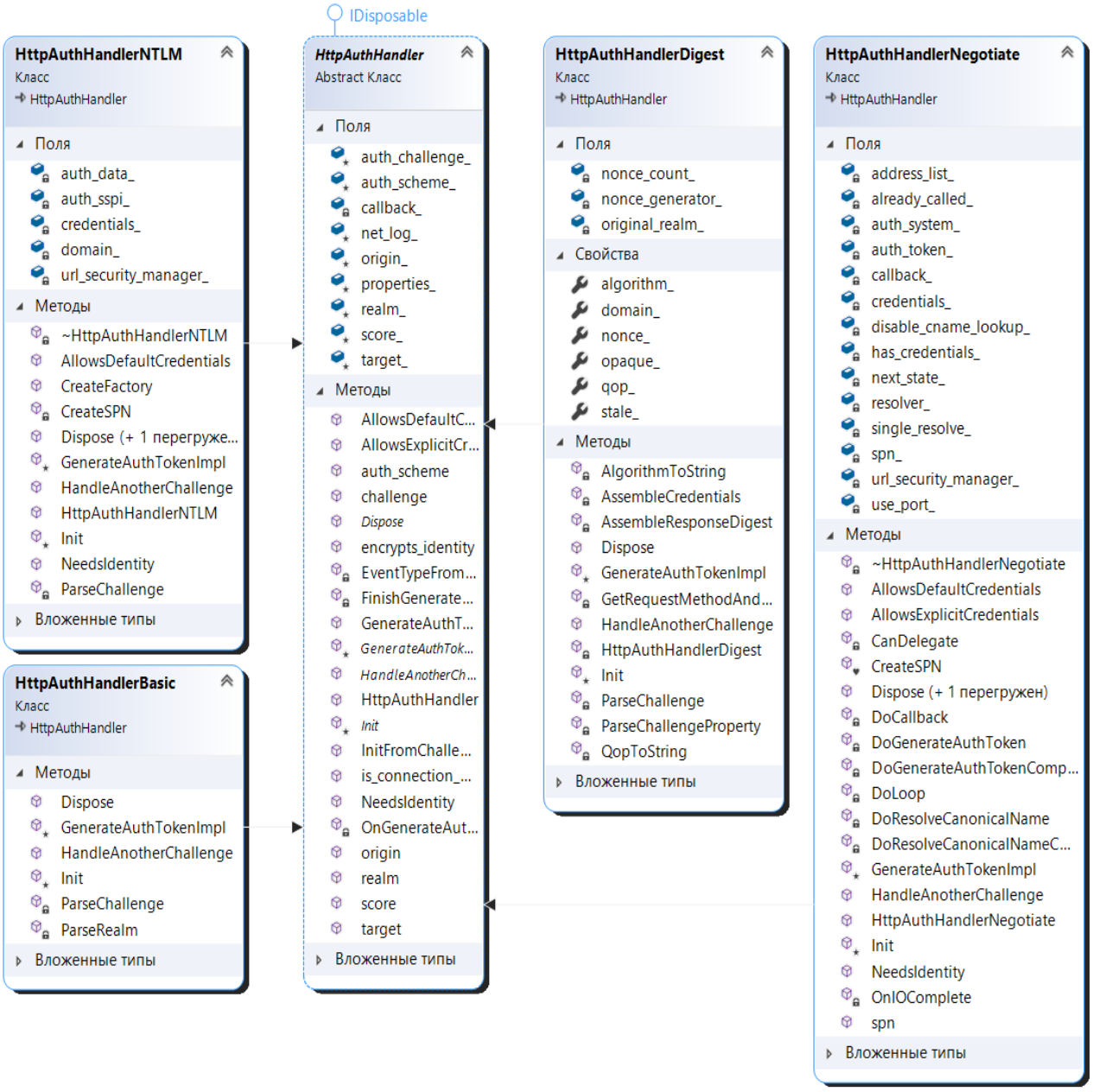

Рисунок 16 – Диаграмма классов программного средства

Класс HttpAuthHandlerNegotiate производит аутентификацию с помощью протокола Kerberos. При невозможности его использования, выполняется реализация класса HttpAuthHandlerNTLM.

Классы HttpAuthHandlerDigest и HttpAuthHandlerBasic реализуют методы для механизмов аутентификации Digest и Basic соответственно.

4 Тестирование программных средств аутентификации

4.1 Функциональное тестирование

Функциональное тестирование относится к категории тестирования «черного ящика». Каждая функция программы тестируется ввода её исходных данных и анализа выходных. При этом внутренняя структура программы учитывается очень редко [21].

Функциональное тестирование предполагает составление плана тестирования, который строится на основе вариантов использования и включает в себя описание тестов и их результаты.

План функционального тестирования программных средств приведен в таблице 7, результаты приведены в приложении В.

Таблица 7 – План тестирования

№	Вариант использования	Тест	Результат
1	Ввести данные для входа	Выполняется вход с корректными данными	Тест пройден. Сервер предоставил доступ к ресурсу (рисунок В.1).
2	Ввести данные для входа	Выполняется вход с некорректными данными	Тест пройден. Выдана страница отказа в доступе (рисунок В.2).
3	Ввести данные для входа	Выполняется вход с пустыми входными данными	Тест пройден. Повторно выдана страница для аутентификации (рисунок В.3).

На основе анализа результатов тестирования можно сделать вывод, что основные функции разработанного программного средства аутентификации работают корректно.

4.2 Модульное тестирование

Модульное тестирование – это процесс тестирования отдельных блоков, подпрограмм, классов и процедур, образующих крупную программу.

Модульное тестирование в основном ориентировано на использование метода «белого ящика». Объясняется это прежде всего тем, что при последующем переходе к тестированию более крупных программных единиц, например, программ в целом, применимость метода «белого ящика» снижается [23].

Для создания модульных тестов воспользуемся платформой модульного тестирования для C#, встроенной в Visual Studio. Для тестирования выберем класс HttpAuthHandler.

Реализация теста приведена в листинге 1.

Исходный код функций тестирования, используемых в модульном тестировании, приведен в приложении Г.

Листинг 1 – Исходный код модульного теста

```
using Blink.External;
using Blink.External.Net;
using Blink.URL;
using BlinkUnitTest.Blink.External.Net;
using BlinkUnitTest.Blink.External.Net.Http;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;
using System.Text;

namespace Cryptosoft.Titan.External.Net.Http
{
    using CapturedEntryList = List<CapturedNetLogEntry>;

    /// <summary>
    /// Summary description for HttpAuthHandlerUnitTest
    /// </summary>
    [TestClass]
    public class HttpAuthHandlerUnitTest
    {
        [TestMethod]
        public void HttpAuthHandlerTest_NetLog()
        {
              var Origin = new GURL("http://www.example.com");
            String challenge = "Mock asdf";
            AuthCredentials credentials = new
                   AuthCredentials("user", "pass");
            Ref<String> auth_token = null;
            HttpRequestInfo request = new HttpRequestInfo();

            for (int i = 0; i < 2; ++i)
            {
                bool async = (i == 0);
```

```
                for (int j = 0; j < 2; ++j)
                {
                    int rv = (j == 0) ?
                   (Int32)NetErrors.Error.OK :
                   (Int32)NetErrors.Error.ERR_UNEXPECTED;
                    for (int k = 0; k < 2; ++k)
                    {
                        TestCompletionCallback test_callback =
new TestCompletionCallback();
                        HttpAuth.Target target =
                            (k == 0) ?
HttpAuth.Target.AUTH_PROXY : HttpAuth.Target.AUTH_SERVER;
                        NetLog.EventType event_type =
                            (k == 0) ?
NetLog.EventType.TYPE_AUTH_PROXY :
NetLog.EventType.TYPE_AUTH_SERVER;
                        HttpAuthChallengeTokenizer tokenizer =
new HttpAuthChallengeTokenizer(
                            challenge,
                            0,
                            challenge.Length);
                        HttpAuthHandlerMock mock_handler = new
HttpAuthHandlerMock();
                        CapturingNetLog capturing_net_log =
null;
                        BoundNetLog bound_net_log =
BoundNetLog.Make(
                            capturing_net_log,
                            NetLog.SourceType.SOURCE_NONE);


mock_handler.InitFromChallenge(tokenizer, target,
                                                        Origin,
bound_net_log);

mock_handler.SetGenerateExpectation(async, rv);

mock_handler.GenerateAuthToken(credentials, request,

test_callback.callback(), auth_token);
                        if (async)
                            test_callback.WaitForResult();

                        CapturedEntryList entries;
                        capturing_net_log.GetEntries(out
entries);

                        Assert.AreEqual(2u, entries.Count);

Assert.IsTrue(NetLogTest.LogContainsBeginEvent(entries, 0,
event_type));

Assert.IsTrue(NetLogTest.LogContainsEndEvent(entries, 1,
event_type));
```

```
                    }
                }
            }
        }
    }
}
```

В Visual Studio доступны графические пользовательские интерфейсы для тестирования функций. С помощью них можно удобно запускать модульные тесты и смотреть результаты. Скриншот, подтверждающий проведение модульного тестирования, приведен на рисунке 17.

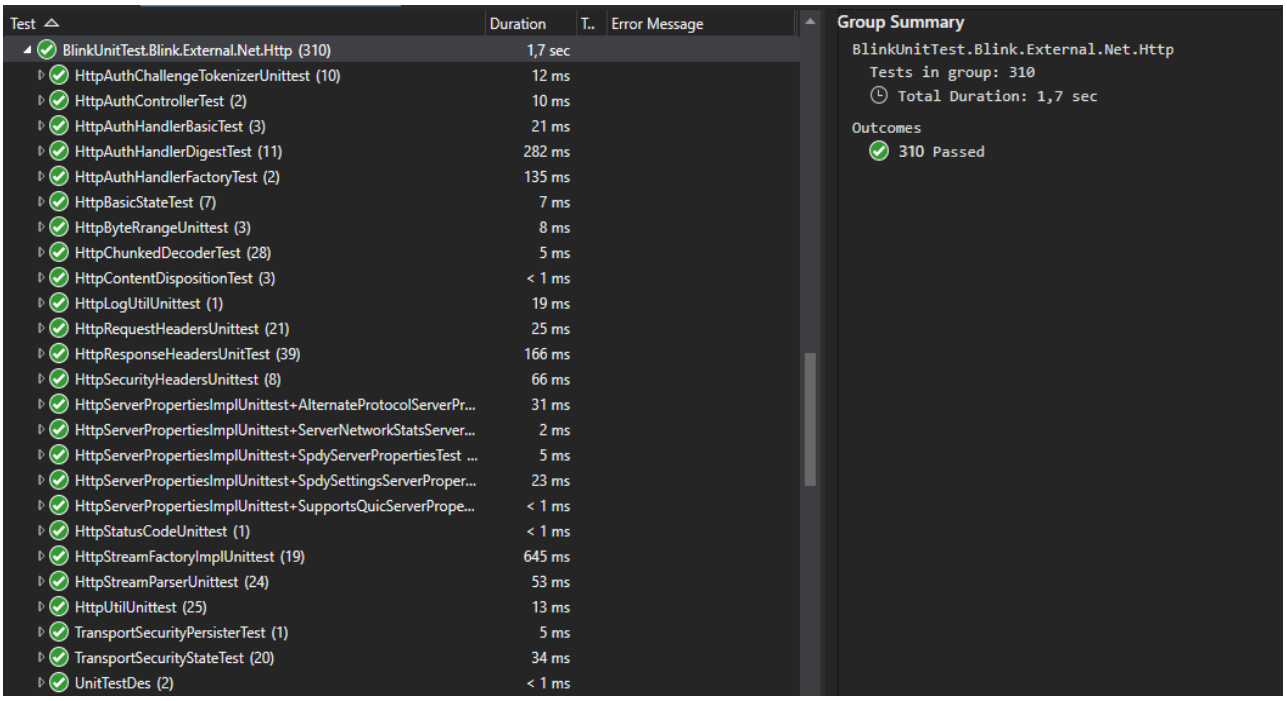

Рисунок 17 – Результат модульного тестирования

Все модульные тесты были успешно пройдены, что свидетельствует о корректности работы программного средства.

4.3 Метрики кода

4.3.1 Цикломатическая метрика Мак-Кейба

При вычислении цикломатической сложности используется граф потока управления программы. Узлы графа соответствуют неделимым группам команд программы, они соединены ориентированными рёбрами, если группа команд, соответствующая второму узлу, может быть выполнена непосредственно после группы команд первого узла.

Проведем подсчет данной метрики для функции ChooseBestChallenge, которая выбирает схему аутентификации из поддерживаемых. Функция приведена в листинге 2.

Листинг 2 – Функция ChooseBestChallenge

```
public static void ChooseBestChallenge(
     HttpAuthHandlerFactory http_auth_handler_factory,
     HttpResponseHeaders headers,
     Target target,
     GURL origin,
     HashSet<Scheme> disabled_schemes,
     BoundNetLog net_log,
     ref HttpAuthHandler handler)
{
     Debug.Assert(http_auth_handler_factory != null);
     Debug.Assert(handler == null);

     HttpAuthHandler best = null;
     String header_name = GetChallengeHeaderName(target);
     String cur_challenge;

     var iter = new Ref<Int32>(0);
     while      (headers.EnumerateHeader(iter,      (StringPiece)
header_name, out cur_challenge))
     {
          HttpAuthHandler cur;
          int                          rv                          =
http_auth_handler_factory.CreateAuthHandlerFromString(
               cur_challenge, target, origin, net_log, out cur);
          if (rv != (int) NetErrors.Error.OK)
          {
               Debug.WriteLine($"VLOG(1)  <<  Unable  to  create
AuthHandler.  Status:  {NetErrors.ErrorToString(rv)}  Challenge:
{cur_challenge}");
               continue;
          }

          if (cur != null && (best == null || best.score() <
cur.score()) && !disabled_schemes.Contains(cur.auth_scheme()))
               best = cur;
     }

     handler = best;
 }
```

Граф потока управления программы для рассматриваемой функции приведен на рисунке 18.

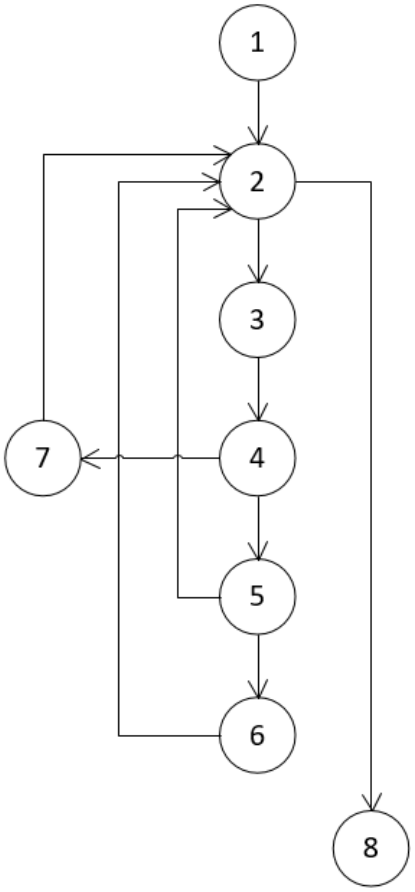

Рисунок 18 – Граф потока управления программы

Для вычисления цикломатической сложности V(G) применяется следующая формула:

$$V(G) = e - n + 2p,$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит 8 вершин, 10 дуг. Для простой программы, или подпрограммы, или метода число компонентов связанности графа всегда равно 1. Подставляем полученные значения и получаем:

$$V(G) = 10 - 8 + 2 = 4.$$

Таким образом значение цикломатической сложности равно 4. Это значение не превышает 10, следовательно, функция не является структурно сложной.

4.3.2 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы. Основу метрики Холстеда составляют четыре измеряемые характеристики программы [24]:

– число уникальных операторов программы, включая символыразделители, имена процедур и знаки операций (словарь операторов);

– число уникальных операндов программы, включая константы и имена переменных (словарь операндов);

– общее число операторов в программе;

– бщее число операндов в программе;

– теоретическое число уникальных операторов;

– теоретическое число уникальных операндов.

Подсчет операторов и операндов для функции ChooseBestChallenge приведен в таблицах 8 и 9 соответственно.

Таблица 8 – Операторы

Операторы	Число вхождений
==	2
!=	2
=	9
&&	2
||	1
вызов функции ()	7
приведение типов ()	2
new	1
out	2
ref	1
var	1

Продолжение таблицы 8

Операторы	Число вхождений
while	1
continue	1
if	2
;	12
{}	5
,	12
.	13
$	2
<	1
!	1
GetChallengeHeaderName	1
EnumerateHeader	1
CreateAuthHandlerFromString	1
WriteLine	1
ErrorToString	1
score	2
Contains	1
auth_scheme	1
n1 = 29	N1 = 89

Таблица 9 – Операнды

Операнды	Число вхождений
null	5
0	1
HttpAuthHandlerFactory	1
http_auth_handler_factory	3
HttpResponseHeaders	1

Продолжение таблицы 9

Операнды	Число вхождений
headers	2
Target	1
target	3
GURL	1
origin	2
HashSet<Scheme>	1
disabled_schemes	2
BoundNetLog	1
net_log	2
HttpAuthHandler	3
handler	3
Ref<Int32>	1
StringPiece	2
OK	1
best	5
String	2
header_name	2
cur_challenge	4
iter	2
rv	3
n2 = 25	N2 = 54

Сложность программы рассчитывается по формуле:

$$HD = (n1 / 2) * (n2 / N2) = (29 / 2) * (54 / 25) = 31.32.$$

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 89 + 54 = 143.$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 29 + 25 = 54.$$

Теоретический словарь программы вычисляется по формуле:

$$n' = n_1' + n_2' = 3.$$

Объем программы рассчитывается по формуле:

$$HV = N * \log_2(n) = 143 * \log_2(54) = 823.$$

Потенциальный объем программы рассчитывается по формуле:

$$V^* = (2 + n`) * \log_2(2 + n`) = 5 * \log_2(5) = 11.6.$$

Теорический уровень программы рассчитывается по формуле:

$$L` = V^* / HV = 11.6 / 823 = 0.01404.$$

Значение уровня программы согласно метрике Холстеда равняется 0.01404.

Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда, однако остается в пределах допустимого [23].

4.3.3 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

– вводимые переменные для расчетов и для обеспечения вывода (P);

– модифицируемые, или создаваемые внутри программы переменные (M);

– переменные, участвующие в управлении работой программного модуля (C);

– не используемые в программе («паразитные») переменные (T).

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Метрика Чепина:

$$Q = aP + bM + cC + dT,$$

где a, b, с, d– весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы.

По мнению автора метрики, наибольший вес, равный 3, имеет функциональная группа C, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a = 1$, $b = 2$, $d = 0.5$. Весовой коэффициент группы T не равен 0, поскольку «паразитные» переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание.

С учетом весовых коэффициентов выражение принимает вид:

$$Q = P + 2 * M + 3 * C + 0.5 * T.$$

В представленной функции характеристики равны:

$$P = 7,$$

$$M = 7,$$

$$C = 4,$$

$$T = 0.$$

Подставим полученные значения в формулу:

$$Q = 7 + 2 * 7 + 3 * 4 + 0 = 33.$$

Значение метрики Чепина равняется 33. Если в результате изменения рассматриваемой функции значение метрики значительно увеличится, это будет означать низкое качество изменений.

Сравнение значения метрики Чепина для различных программ обычно неэффективно, обычно она используется для анализа изменений одного и того же участка кода.

По результатам расчета метрик кода можно сделать вывод, что сложность данной программы низкая, она является достаточно легкой для понимания.

Это означает, что код имеет достаточно высокий уровень качетсва, что позволит легко поддерживать и изменять его при дальнейшей поддержке программных средств.

Заключение

В ходе выполнения выпускной квалификационной работы были разработаны программные средства аутентификации в браузере для операционной системы QP OS.

Был проведен анализ предметной области и произведена постановка задач, в ходе чего было проанализированы программные обеспечения аналогичного назначения, рассмотрены их особенности и выделены недостатки, анализ существующих разработок показал необходимость разработки собственного аналога. Также были разработаны функциональные и нефункциональные требования.

Было проведено планирование разработки и оценка бюджета. Длительность реализации проекта составляет 95 дня, а предполагаемые затраты – 216970 руб.

На этапе проектирования данных программных средств применялся язык моделирования UML с помощью среды разработки диаграмм «ArgoUML».

Для реализации программных средств использовался язык программирования C#, код на котором писался в среде программирования Microsoft Visual Studio 2019.

Было проведено функциональное и модульное тестирование разработанного приложения. На основе результатов тестирования можно сделать вывод, что разработанное приложение соответствует поставленным требованиям и работает корректно. В приложении Д представлен акт о внедрении данного продукта в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

Таким образом, в результате выполнения выпускной квалификационной работы были разработаны программные средства, отвечающее всем требованиям технического задания.

Список использованных источников

1. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf (дата обращения: 23.04.2021).

2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 23.04.2021).

3. Стин ван М., Таненбаум Э.С. Распределенные системы / пер. с англ. В.А. Яроцкого. – М.: ДМК Пресс, 2021. – 584 с.: ил.

4. Методы исследования надёжности удалённой аутентификации. [Электронный ресурс]. URL: https://www.aladdin-rd.ru/company/pressroom/articles/metody_issledovania_nadeznosti_udalennoj_autentifikacii (дата обращения: 22.04.2021).

5. Аутентификация, идентификация, авторизация. [Электронный ресурс]. URL: http://www.msiit.ru/x/miszki/_4___.html (дата обращения: 24.04.2021).

6. Протоколы аутентификации [Электронный ресурс]. URL: https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema11#p112 (дата обращения: 14.05.2021).

7. Баричев С. Г., Гончаров В. В., Серов Р. Е. Основы современной криптографии. – М.: «Горячая линия – Телеком», 2001 – 120 с.

8. Симметричная криптография [Электронный ресурс]. URL: http://mf.grsu.by/UchProc/livak/b_protect/zok_2.htm (дата обращения: 13.05.2021).

9. Реберн, К. Служба сетевой аутентификации Kerberos / Массачусетский технологический институт, 2005.

10. Протокол аутентификации Нидхэма-Шредера в случаях симметричной и асимметричной системы шифрования [Электронный ресурс]. URL: http://citforum.ru/nets/semenov/6/n_s_p_k.shtml (дата обращения: 14.05.2021).

11. NTLM [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/NTLM (дата обращения: 13.05.2021).

12. Обзор способов и протоколов аутентификации в веб-приложениях [Электронный ресурс]. URL: https://habr.com/ru/company/dataart/blog/262817/ (дата обращения: 10.05.2021).

13. SSPI [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/windows/win32/secauthn/sspi?redirectedfrom=MSDN (дата обращения: 11.05.2021).

14. Сквозная аутентификация в Google Chrome [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/security/choose-an-authentication-mode?view=sql-server-ver15 (дата обращения: 01.04.2021).

15. Анализ требований // sites.google.ru [Электронный ресурс]. URL: https://sites.google.com/site/infoprobusinessanalysis/project-definition/ processrazrabotki-programmnogo-obespecenia/analiz-trebovanij (дата обращения: 16.05.2021).

16. Поддержка удостоверений и настройка Microsoft Edge [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/deployedge/microsoft-edge-security-identity#how-users-can-sign-into-microsoft-edge (дата обращения: 01.04.2021).

17. Боронина, Л. Н. Основы управления проектами / Л. Н. Боронина, З. В. Сенук ; М-во образования и науки рос. Федерации, Урал. Федер. Ун-т. – Екатеринбург : изд-во Урал. ун-та, 2015. — 112 с.

18. Формирование требований и классификация требований// sites.google [Электронный ресурс]. URL: https://analytics.infozone.pro/formation-requirements-and-classificationrequirements/#i-10 (дата обращения: 16.05.2021).

19. Технология RUP // Новостной сайт об информационных системах [Электронный ресурс]. URL: https://wiki.clan.su/publ/primery_ts_po_tekhnologija_rup_rational_unified_process/1-1-0-39 (дата обращения: 05.04.2021).

20. Канер, С. Тестирование программного обеспечения / С. Канер, Д. Фолк, Е.К. Нгуен. – К.: ДиаСофт, 2001. – 544 с.

21. Статья на Wikipedia об архитектуре клиент-сервер // Электронная энциклопедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Клиент_—_сервер (дата обращения: 07.04.2021).

22. Котляров, В.П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. – М.: Бином. Лаборатория знаний, 2006. – 285 с.

23. Azure Active Directory [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/azure/active-directory/fundamentals/active-directory-whatis (дата обращения: 10.05.2021).

24. Холстед, М. Х. Начала науки о программах / М.Х. Холстед; пер. с англ. В. М. Юфы. - М. : Финансы и статистика, 1981. - 128 с., ил.

Приложение А
ГЛОССАРИЙ
(справочное)

Идентификация – процедура распознавания субъекта по его идентификатору.

Аутентификация – процедура проверки подлинности субъекта, которая позволяет достоверно убедиться в том, что субъект, предъявивший свой идентификатор, на самом деле является именно тем субъектом, идентификатор которого он использует.

Авторизация – процедура предоставления субъекту определенных прав доступа к ресурсам системы после прохождения им процедуры аутентификации.

Двусторонняя (взаимная) аутентификация – вид аутентификации, при котором проверяется подлинность как пользователя, так и сервера.

Симметричная криптосистема – способ шифровнаия, в котором для шифрования и дешифрования применяется один и тот же криптографический ключ.

Асимметричная криптосистема (криптографическая система с открытым ключом) – система шифрования, при которой открытый ключ передается по открытому (незащищенному) каналу и используется для проверки электронной подписи или шифрования сообщений.

Клиент-сервер – вычислительная или сетевая архитектура, в которой сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит использование итеративной модель разработки.

UML (Unified Modeling Language) – язык графического описания, с помощью которого осуществляется объектное моделирование в области разработки программного обеспечения, проектирования систем и отображения организационных структур.

Модульное тестирование (Unit testing) – тестирование каждой атомарной функциональности приложения отдельно, в искусственно созданной среде.

Приложение Б

ИСХОДНЫЙ КОД ПРОГРАММНЫХ СРЕДСТВ

(рекомендуемое)

Листинг Б.1 – HttpAuth.cs

```
using Cryptosoft.Titan.Base.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using Blink.URL;
using Cryptosoft.Titan.Base;
namespace Blink.External.Net
{
      public static class HttpAuth
      {
            public enum Target
            {
                  AUTH_NONE = -1,
                  AUTH_PROXY = 0,
                  AUTH_SERVER = 1,
                  AUTH_NUM_TARGETS = 2,
            }
            public enum AuthorizationResult
            {
                  AUTHORIZATION_RESULT_ACCEPT,
                  AUTHORIZATION_RESULT_REJECT,
                  AUTHORIZATION_RESULT_STALE,
                  AUTHORIZATION_RESULT_INVALID,
                  AUTHORIZATION_RESULT_DIFFERENT_REALM,
            }
            public enum IdentitySource
            {
                  IDENT_SRC_NONE,
                  IDENT_SRC_PATH_LOOKUP,
                  IDENT_SRC_URL,
                  IDENT_SRC_REALM_LOOKUP,
                  IDENT_SRC_EXTERNAL,
                  IDENT_SRC_DEFAULT_CREDENTIALS,
            }
            public enum Scheme
            {
                  AUTH_SCHEME_BASIC = 0,
                  AUTH_SCHEME_DIGEST,
                  AUTH_SCHEME_NTLM,
                  AUTH_SCHEME_NEGOTIATE,
                  AUTH_SCHEME_SPDYPROXY,
                  AUTH_SCHEME_MOCK,
                  AUTH_SCHEME_MAX,
            }
            public class/*struct*/ Identity
            {
                  public IdentitySource source;
                  public bool invalid;
                  public AuthCredentials credentials;
                public Identity()
                {
                    source = IdentitySource.IDENT_SRC_NONE;
                    invalid = true;
                }
            }
            public static String GetChallengeHeaderName(Target target)
            {
                  switch (target)
                  {
                        case Target.AUTH_PROXY:
                              return "Proxy-Authenticate";
                        case Target.AUTH_SERVER:
                              return "WWW-Authenticate";
                        default:
```

```
                        Lermontov.NotReached();
                        return String.Empty;
                }
        }
        public static String GetAuthorizationHeaderName(Target target)
        {
                switch (target)
                {
                        case Target.AUTH_PROXY:
                                return HttpRequestHeaders.kProxyAuthorization;
                        case Target.AUTH_SERVER:
                                return HttpRequestHeaders.kAuthorization;
                        default:
                                Lermontov.NotReached();
                                return String.Empty;
                }
        }
        public static String GetAuthTargetString(Target target)
        {
                switch (target)
                {
                        case Target.AUTH_PROXY:
                                return "proxy";
                        case Target.AUTH_SERVER:
                                return "server";
                        default:
                                Lermontov.NotReached();
                                return String.Empty;
                }
        }
        private static String[] kSchemeNames =
        {
                "basic",
                "digest",
                "ntlm",
                "negotiate",
                "spdyproxy",
                "mock",
        };
        public static String SchemeToString(Scheme scheme)
        {
                if (scheme < Scheme.AUTH_SCHEME_BASIC || scheme >=
Scheme.AUTH_SCHEME_MAX)
                {
                        Lermontov.NotReached();
                        return "invalid_scheme";
                }
                return kSchemeNames[(int)scheme];
        }
        public static void ChooseBestChallenge(
                HttpAuthHandlerFactory http_auth_handler_factory,
                HttpResponseHeaders headers,
                Target target,
                GURL origin,
                HashSet<Scheme> disabled_schemes,
                BoundNetLog net_log,
                ref HttpAuthHandler handler)
        {
                Debug.Assert(http_auth_handler_factory != null);
                Debug.Assert(handler == null);
                HttpAuthHandler best = null;
                String header_name = GetChallengeHeaderName(target);
                String cur_challenge;
                var iter = new Ref<Int32>(0);
```

```
                while (headers.EnumerateHeader(iter, (StringPiece)
header_name, out cur_challenge))
                {
                        HttpAuthHandler cur;
                        int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                                cur_challenge, target, origin, net_log, out
cur);
                        if (rv != (int) NetErrors.Error.OK)
                        {
                                Debug.WriteLine($"VLOG(1) << Unable to create
AuthHandler. Status: {NetErrors.ErrorToString(rv)} Challenge:
{cur_challenge}");
                                continue;
                        }

                        if (cur != null && (best == null || best.score() <
cur.score()) && !disabled_schemes.Contains(cur.auth_scheme()))
                                best = cur;
                }

                handler = best;
            }
            public static AuthorizationResult HandleChallengeResponse(
                HttpAuthHandler handler,
                HttpResponseHeaders headers,
                Target target,
                HashSet<Scheme> disabled_schemes,
                ref String challenge_used)
            {
                Debug.Assert(handler != null);
                Debug.Assert(headers != null);
                Debug.Assert(challenge_used != null);
                challenge_used = null; //.clear();
                HttpAuth.Scheme current_scheme = handler.auth_scheme();
                if (disabled_schemes.Contains(current_scheme))
                        return
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT;

                String current_scheme_name = SchemeToString(current_scheme);
                String header_name = GetChallengeHeaderName(target);
                var iter = new Ref<Int32>(0);
                String challenge;
                HttpAuth.AuthorizationResult authorization_result =
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID;
                while (headers.EnumerateHeader(iter, (StringPiece)
header_name, out challenge))
                {
                        HttpAuthChallengeTokenizer props = new
HttpAuthChallengeTokenizer(challenge, 0, challenge.Length);
                        if
(!External.Base.StringUtils.LowerCaseEqualsASCII(props.scheme(),
current_scheme_name))
                                continue;
                        authorization_result =
handler.HandleAnotherChallenge(props);
                        if (authorization_result !=
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID)
                        {
                                challenge_used = challenge;
                                return authorization_result;
                        }
                }
                return
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT;
```

```
            }
      }
}
```

Листинг Б.2 – HttpAuthSspiWin.cs

```
using Cryptosoft.Titan.Base.Logging;
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
using Blink.Core;
using Cryptosoft.Titan.PInvoke;
using System.Text;
namespace Blink.External.Net
{
      using SECURITY_STATUS = Int32;
      using SEC_WCHAR = UInt16;   //unsigned short
      using SEC_CHAR = Byte;     //char;
      using TimeStamp = Int64;
      using PTimeStamp = Int64;
      using SECURITY_INTEGER = Int64;
      using SECURITY_HANDLE = SecHandle;
      using CredHandle = SecHandle;
      using PCredHandle = SecHandle;
      using CtxtHandle = SecHandle;
      using PCtxtHandle = SecHandle;
      using PSecBufferDesc = SecBufferDesc;
      using PSecPkgInfoW = WinAPI.SecPkgInfo;
      using static WinAPI;
      public static partial class Static
      {
            internal static void SplitDomainAndUser(String combined, out
String domain, out String user)
            {
                  int backslash_idx = combined.IndexOf('\\');
                  if (backslash_idx == -1)
                  {
                        domain = "";//.clear();
                        user = combined;
                  }
                  else
                  {
                        domain = combined.Substring(0, backslash_idx);
                        user = combined.Substring(backslash_idx + 1);
                  }
            }
            internal static int DetermineMaxTokenLength(
                  SSPILibrary library, string package, ref UInt32
max_token_length)
            {
                  Debug.Assert(library != null);
                  var status = library.QuerySecurityPackageInfo(package, out
var PkgInfoPtr);
                  var pkg_info =
Marshal.PtrToStructure<SecPkgInfo>(PkgInfoPtr);
                  int rv = MapQuerySecurityPackageInfoStatusToError(status);
                  if (rv != (int)NetErrors.Error.OK)
                        return rv;
                  UInt32 token_length = pkg_info.cbMaxToken;
                  status = library.FreeContextBuffer(PkgInfoPtr);
                  rv = MapFreeContextBufferStatusToError(status);
                  if (rv != (int)NetErrors.Error.OK)
                        return rv;
                  max_token_length = token_length;
                  return (int)NetErrors.Error.OK;
            }
```

```
            public static int
MapAcquireCredentialsStatusToError(SECURITY_STATUS status, string package)
            {
                  Debug.WriteLine($"VLOG(1) << AcquireCredentialsHandle
returned 0x{status:X}");
                  switch (status)
                  {
                        case SEC_E_OK:
                              return (int) NetErrors.Error.OK;
                        case SEC_E_INSUFFICIENT_MEMORY:
                              return (int) NetErrors.Error.ERR_OUT_OF_MEMORY;
                        case SEC_E_INTERNAL_ERROR:
                              Debug.WriteLine($"LOG(WARNING) <<
AcquireCredentialsHandle returned unexpected status 0x{status:X}");
                              return (int)
NetErrors.Error.ERR_UNEXPECTED_SECURITY_LIBRARY_STATUS;
                        case SEC_E_NO_CREDENTIALS:
                        case SEC_E_NOT_OWNER:
                        case SEC_E_UNKNOWN_CREDENTIALS:
                              return (int)
NetErrors.Error.ERR_INVALID_AUTH_CREDENTIALS;
                        case SEC_E_SECPKG_NOT_FOUND:
                              return (int)
NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME;
                        default:
                              Debug.WriteLine($"LOG(WARNING) <<
AcquireCredentialsHandle returned undocumented status 0x{status:X}");
                              return (int)
NetErrors.Error.ERR_UNDOCUMENTED_SECURITY_LIBRARY_STATUS;
                  }
            }
            public static int AcquireExplicitCredentials(SSPILibrary library,
                                          /*SEC_WCHAR[]*/string package,
                                          String domain,
                               String user,
                               String password,
                               ref CredHandle cred)
            {
                  SEC_WINNT_AUTH_IDENTITY identity = new
SEC_WINNT_AUTH_IDENTITY();
                  identity.Flags = SEC_WINNT_AUTH_IDENTITY_UNICODE;
                  identity.User = user;
                  identity.UserLength = user.Length;
                  identity.Domain =domain;
                  identity.DomainLength = domain.Length;
                  identity.Password = password;
                  identity.PasswordLength = password.Length;
                  IntPtr IdentityPtr =
Marshal.AllocHGlobal(Marshal.SizeOf(identity));
                  Marshal.StructureToPtr(identity, IdentityPtr, false);
                  TimeStamp expiry = 0;
                  SECURITY_STATUS status = library.AcquireCredentialsHandle(
                        null,       // pszPrincipal
                        package,    //const_cast<SEC_WCHAR*>(package),  //
pszPackage
                        SECPKG_CRED_OUTBOUND,  // fCredentialUse
                        IntPtr.Zero,        // pvLogonID
                        IdentityPtr, //ref identity,  // pAuthData
                        IntPtr.Zero,  // pGetKeyFn (not used)
                        IntPtr.Zero,  // pvGetKeyArgument (not used)
                        ref cred,  // phCredential
                        ref expiry);  // ptsExpiry

                  Marshal.FreeHGlobal(IdentityPtr);
                  return MapAcquireCredentialsStatusToError(status, package);
```

```
            }
            public static int AcquireDefaultCredentials(SSPILibrary library,
/*const SEC_WCHAR**/ref string package, ref CredHandle cred)
            {
                  TimeStamp expiry = 0;
                  SECURITY_STATUS status = library.AcquireCredentialsHandle(
                        null,
                        /package,  // pszPackage
                        SECPKG_CRED_OUTBOUND,  // fCredentialUse
                        IntPtr.Zero,  // pvLogonID
                        IntPtr.Zero/*null*/,  // pAuthData
                        IntPtr.Zero,  // pGetKeyFn (not used)
                        IntPtr.Zero,  // pvGetKeyArgument (not used)
                        ref cred,  // phCredential
                        ref expiry);  // ptsExpiry
                  return MapAcquireCredentialsStatusToError(status, package);
            }
            public static int
MapInitializeSecurityContextStatusToError(SECURITY_STATUS status)
            {
                  Debug.WriteLine($"VLOG(1) << InitializeSecurityContext
returned 0x{status : H}");
                  switch (status)
                  {
                        case WinAPI.SEC_E_OK:
                        case WinAPI.SEC_I_CONTINUE_NEEDED:
                              return (int)NetErrors.Error.OK;
                        case WinAPI.SEC_I_COMPLETE_AND_CONTINUE:
                        case WinAPI.SEC_I_COMPLETE_NEEDED:
                        case WinAPI.SEC_I_INCOMPLETE_CREDENTIALS:
                        case WinAPI.SEC_E_INCOMPLETE_MESSAGE:
                        case WinAPI.SEC_E_INTERNAL_ERROR:
                              Debug.WriteLine($"LOG(WARNING) <<
InitializeSecurityContext returned unexpected status 0x{status:X}");
                              return
(int)NetErrors.Error.ERR_UNEXPECTED_SECURITY_LIBRARY_STATUS;
                        case WinAPI.SEC_E_INSUFFICIENT_MEMORY:
                              return (int)NetErrors.Error.ERR_OUT_OF_MEMORY;
                        case WinAPI.SEC_E_UNSUPPORTED_FUNCTION:
                              Lermontov.NotReached();
                              return (int)NetErrors.Error.ERR_UNEXPECTED;
                        case WinAPI.SEC_E_INVALID_HANDLE:
                              Lermontov.NotReached();
                              return (int)NetErrors.Error.ERR_INVALID_HANDLE;
                        case WinAPI.SEC_E_INVALID_TOKEN:
                              return
(int)NetErrors.Error.ERR_INVALID_RESPONSE;
                        case WinAPI.SEC_E_LOGON_DENIED:
                              return (int)NetErrors.Error.ERR_ACCESS_DENIED;
                        case WinAPI.SEC_E_NO_CREDENTIALS:
                        case WinAPI.SEC_E_WRONG_PRINCIPAL:
                              return
(int)NetErrors.Error.ERR_INVALID_AUTH_CREDENTIALS;
                        case WinAPI.SEC_E_NO_AUTHENTICATING_AUTHORITY:
                        case WinAPI.SEC_E_TARGET_UNKNOWN:
                              return
(int)NetErrors.Error.ERR_MISCONFIGURED_AUTH_ENVIRONMENT;
                        default:
                              Debug.WriteLine($"LOG(WARNING) <<
InitializeSecurityContext returned undocumented status 0x{status:X}");
                              return
(int)NetErrors.Error.ERR_UNDOCUMENTED_SECURITY_LIBRARY_STATUS;
                  }
            }
```

```
            public static int
MapQuerySecurityPackageInfoStatusToError(SECURITY_STATUS status)
            {
                  Debug.WriteLine($"VLOG(1) << QuerySecurityPackageInfo
returned 0x{status:X}");
                  switch (status)
                  {
                        case WinAPI.SEC_E_OK:
                              return (int)NetErrors.Error.OK;
                        case WinAPI.SEC_E_SECPKG_NOT_FOUND:
                              return
(int)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME;
                        default:
                              Debug.WriteLine($"LOG(WARNING) <<
QuerySecurityPackageInfo returned undocumented status 0x{status:X}");
                              return
(int)NetErrors.Error.ERR_UNDOCUMENTED_SECURITY_LIBRARY_STATUS;
                  }
            }

            public static int
MapFreeContextBufferStatusToError(SECURITY_STATUS status)
            {
                  Debug.WriteLine($"VLOG(1) << FreeContextBuffer returned
0x{status:X}");
                  switch (status)
                  {
                        case WinAPI.SEC_E_OK:
                              return (int)NetErrors.Error.OK;
                        default:
                              Debug.WriteLine($"LOG(WARNING) <<
FreeContextBuffer returned undocumented status 0x{status:X}");
                              return
(int)NetErrors.Error.ERR_UNDOCUMENTED_SECURITY_LIBRARY_STATUS;
                  }
            }
      }
      public interface SSPILibrary
      {
            SECURITY_STATUS AcquireCredentialsHandle(string pszPrincipal,

       string pszPackage,
       uint fCredentialUse,
       IntPtr pvLogonId,
       IntPtr pvAuthData,
       IntPtr pGetKeyFn,

       IntPtr pvGetKeyArgument,

       ref PCredHandle phCredential,

       ref PTimeStamp ptsExpiry);

            /SECURITY_STATUS InitializeSecurityContext(ref PCredHandle
phCredential,

      /*PCtxtHandle*/IntPtr phContext,

      /*SEC_WCHAR* */string pszTargetName,

      int fContextReq,

      int Reserved1,

      int TargetDataRep,
```

```
      /*PSecBufferDesc*/IntPtr pInput,

      int Reserved2,

      out PCtxtHandle phNewContext,

      out PSecBufferDesc pOutput,

      out uint contextAttr,

      /*out PTimeStamp*/IntPtr ptsExpiry);
        SECURITY_STATUS QuerySecurityPackageInfo(/*LPWSTR*/string
pszPackageName, out IntPtr pkgInfo);
        SECURITY_STATUS FreeCredentialsHandle(ref PCredHandle phCredential);
        SECURITY_STATUS DeleteSecurityContext(ref PCtxtHandle phContext);
        SECURITY_STATUS FreeContextBuffer(IntPtr pvContextBuffer);
      }
      public class SSPILibraryDefault : SSPILibrary
      {
            public SSPILibraryDefault()
            { }
            public virtual SECURITY_STATUS
AcquireCredentialsHandle(/*LPWSTR*/string pszPrincipal,

/*LPWSTR*/string pszPackage,

UInt32 fCredentialUse,

/*void* */IntPtr pvLogonId,

/*void* */IntPtr pvAuthData,

/*SEC_GET_KEY_FN*/IntPtr pGetKeyFn,

/*void* */IntPtr pvGetKeyArgument,

ref PCredHandle phCredential,

ref PTimeStamp ptsExpiry)
            {
                  return Import.AcquireCredentialsHandle(pszPrincipal,
pszPackage, fCredentialUse,
                                                                  pvLogonId,
pvAuthData, pGetKeyFn,

pvGetKeyArgument, ref phCredential,
                                                                  ref
ptsExpiry);
            }
            public virtual SECURITY_STATUS InitializeSecurityContext(ref
PCredHandle phCredential,

        /*PCtxtHandle*/IntPtr phContext,

        /*SEC_WCHAR* */string pszTargetName,

        Int32 fContextReq,

        Int32 Reserved1,

        Int32 TargetDataRep,

        /*PSecBufferDesc*/IntPtr pInput,
```

```
        Int32 Reserved2,

        out PCtxtHandle phNewContext,

        out PSecBufferDesc pOutput,

        out UInt32 contextAttr,

        /*out PTimeStamp*/IntPtr ptsExpiry)
            {
                  return Import.InitializeSecurityContext(
                        ref phCredential,
                        phContext,
                        pszTargetName,
                        fContextReq,
                        Reserved1,
                        TargetDataRep,
                        pInput,
                        Reserved2,
                        out phNewContext,
                        out pOutput,
                        out contextAttr,
                        ptsExpiry);
            }
            public virtual SECURITY_STATUS
QuerySecurityPackageInfo(/*LPWSTR*/string pszPackageName, out IntPtr pkgInfo)
            {
                  return Import.QuerySecurityPackageInfo(pszPackageName, out
pkgInfo);
            }
            public virtual SECURITY_STATUS FreeCredentialsHandle(ref
PCredHandle phCredential)
            {
                  return Import.FreeCredentialsHandle(ref phCredential);
            }
            public virtual SECURITY_STATUS DeleteSecurityContext(ref
PCtxtHandle phContext)
            {
                  return Import.DeleteSecurityContext(ref phContext);
            }
            public virtual SECURITY_STATUS FreeContextBuffer(IntPtr
pvContextBuffer)
            {
                  return Import.FreeContextBuffer(pvContextBuffer);
            }
      }
      public class HttpAuthSSPI : IDisposable
      {
            public HttpAuthSSPI(SSPILibrary sspi_library,
               string scheme,
               /*SEC_WCHAR* */string security_package,
                     UInt32 max_token_length)
            {
                  library_ = sspi_library;
                  scheme_ = scheme;
                  security_package_ = security_package;
                  max_token_length_ = max_token_length;
                  can_delegate_ = false;

                  Debug.Assert(library_ != null);
                  SecInvalidateHandle(ref cred_);
                  SecInvalidateHandle(ref ctxt_);
            }
            public bool NeedsIdentity()
```

```
            {
                  return decoded_server_auth_token_.Length == 0;//.isEmpty();
            }
            public bool AllowsExplicitCredentials()
            {
                  return true;
            }
            public HttpAuth.AuthorizationResult
ParseChallenge(HttpAuthChallengeTokenizer tok)
            {
                  if (!Base.StringUtils.LowerCaseEqualsASCII(tok.scheme(),
scheme_.ToLower()))
                        return
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID;

                  string encoded_auth_token = tok.base64_param();
                  if (encoded_auth_token.isEmpty())
                  {
                        if (SecIsValidHandle(ctxt_))
                              return
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT;
                        Debug.Assert(decoded_server_auth_token_.Length ==
0/*.isEmpty()*/);
                        return
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_ACCEPT;
                  }
                  else
                  {
                        if (!SecIsValidHandle(ctxt_))
                              return
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID;
                  }
                  decoded_server_auth_token_ =
Convert.FromBase64String(encoded_auth_token);
                  return
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_ACCEPT;
            }
            public int GenerateAuthToken(AuthCredentials credentials,
                                         string spn,
                                         /*ref*/Ref<string> auth_token)
            {
                  int rv;
                  if (!SecIsValidHandle(cred_))
                  {
                        rv = OnFirstRound(credentials);
                        if (rv != (int)NetErrors.Error.OK)
                              return rv;
                  }
                  Debug.Assert(SecIsValidHandle(cred_));
                  /*void**/IntPtr out_buf;
                  int out_buf_len;
                  IntPtr TempServerToken =
Marshal.AllocHGlobal(decoded_server_auth_token_.Length);
                  Marshal.Copy(decoded_server_auth_token_, 0, TempServerToken,
decoded_server_auth_token_.Length);
                  rv = GetNextSecurityToken(
                        spn,
                        TempServerToken,
                        decoded_server_auth_token_.Length/*.length()*/,
                        out /*&*/out_buf,
                        out /*&*/out_buf_len);

                  Marshal.FreeHGlobal(TempServerToken);
                  if (rv != (int)NetErrors.Error.OK)
                        return rv;
```

```
                Byte[] TempOutBuf = new Byte[out_buf_len];
                Marshal.Copy(out_buf, TempOutBuf, 0, out_buf_len);
                string encode_output;
                encode_output = Convert.ToBase64String(TempOutBuf);
                Marshal.FreeHGlobal(out_buf);//free(out_buf);
                auth_token.Value = scheme_ + " " + encode_output;
                return (int)NetErrors.Error.OK;
          }
          public void Delegate()
          {
                can_delegate_ = true;
          }
          private int OnFirstRound(AuthCredentials credentials)
          {
                Debug.Assert(!SecIsValidHandle(cred_));
                int rv = (int)NetErrors.Error.OK;
                if (credentials != null)
                {
                      String domain;
                      String user;
                      Static.SplitDomainAndUser(credentials.username(), out
domain, out user);
                      rv = Static.AcquireExplicitCredentials(library_,
security_package_, domain,
                                                                  user,
credentials.password(), ref cred_);
                      if (rv != (int)NetErrors.Error.OK)
                            return rv;
                }
                else
                {
                      rv = Static.AcquireDefaultCredentials(library_, ref
security_package_, ref cred_);
                      if (rv != (int)NetErrors.Error.OK)
                            return rv;
                }
                return rv;
          }
          private int GetNextSecurityToken(
                string spn,
                /*void* */IntPtr in_token,
                int in_token_len,
                out /*void** */IntPtr out_token,
                out int out_token_len)
          {
          out_token_len = 0;
          out_token = IntPtr.Zero;
                IntPtr ctxt_ptr;
          SecBufferDesc in_buffer_desc = new SecBufferDesc();
          SecBufferDesc out_buffer_desc = new SecBufferDesc();
          /*SecBufferDesc**/IntPtr in_buffer_desc_ptr;
          SecBuffer in_buffer = new SecBuffer();
          SecBuffer out_buffer = new SecBuffer();
          if (in_token_len > 0)
          {
              // Prepare input buffer.
              in_buffer_desc.ulVersion =
(int)SecBufferType.SECBUFFER_VERSION;
              in_buffer_desc.cBuffers = 1;
                      in_buffer_desc.pBuffers =
Marshal.AllocHGlobal(Marshal.SizeOf(in_buffer));
                      Marshal.StructureToPtr(in_buffer,
in_buffer_desc.pBuffers, false);//in_buffer_desc.pBuffers = &in_buffer;
              in_buffer.BufferType = (int)SecBufferType.SECBUFFER_TOKEN;
              in_buffer.cbBuffer = in_token_len;
```

```
                in_buffer.pvBuffer = /*const_cast<void*>*/(in_token);
                        Marshal.StructureToPtr(in_buffer,
in_buffer_desc.pBuffers, false);    //Синхронизируем in_buffer и
in_buffer_desc.pBuffers после изменения in_buffer
                        //ctxt_ptr = &ctxt_;
                        ctxt_ptr =
Marshal.AllocHGlobal(Marshal.SizeOf(ctxt_));
                        Marshal.StructureToPtr(ctxt_, ctxt_ptr, false);
                        //in_buffer_desc_ptr = &in_buffer_desc;
                        in_buffer_desc_ptr =
Marshal.AllocHGlobal(Marshal.SizeOf(in_buffer_desc));
                        Marshal.StructureToPtr(in_buffer_desc,
in_buffer_desc_ptr, false);
                  }
            else
            {
                if (SecIsValidHandle(ctxt_))
                {
                    Lermontov.NotReached();
                    return (int)NetErrors.Error.ERR_UNEXPECTED;
                }
                        ctxt_ptr = IntPtr.Zero;//null;
                        in_buffer_desc_ptr = IntPtr.Zero;//null;
                  }
            out_buffer_desc.ulVersion = (int)SecBufferType.SECBUFFER_VERSION;
            out_buffer_desc.cBuffers = 1;
                  out_buffer_desc.pBuffers =
Marshal.AllocHGlobal(Marshal.SizeOf(out_buffer));
                  Marshal.StructureToPtr(out_buffer, out_buffer_desc.pBuffers,
false);
                  out_buffer.BufferType = (int)SecBufferType.SECBUFFER_TOKEN
                  out_buffer.cbBuffer = (int)max_token_length_;
                  out_buffer.pvBuffer =
Marshal.AllocHGlobal(out_buffer.cbBuffer);//malloc(out_buffer.cbBuffer);
                  Marshal.StructureToPtr(out_buffer, out_buffer_desc.pBuffers,
false);
                  if (out_buffer.pvBuffer == IntPtr.Zero)
                return (int)NetErrors.Error.ERR_OUT_OF_MEMORY;
            UInt32 context_flags = 0;
            if (can_delegate_)
                context_flags |= (ISC_REQ_DELEGATE | ISC_REQ_MUTUAL_AUTH);
            UInt32 context_attribute = 0;
            String spn16 = spn;//Base.ASCIIToUTF16(spn);
            SECURITY_STATUS status = library_.InitializeSecurityContext(
                ref cred_,  // phCredential
                ctxt_ptr,  // phContext
                spn16,//const_cast<Base.char16*>(spn16.c_str()),  //
pszTargetName
                (Int32)context_flags,  // fContextReq
                0,  // Reserved1 (must be 0)
                (Int32)SECURITY_NATIVE_DREP,  // TargetDataRep
                in_buffer_desc_ptr,  // pInput
                0,  // Reserved2 (must be 0)
                out ctxt_,  // phNewContext
                out out_buffer_desc,  // pOutput
                out context_attribute,  // pfContextAttr
                IntPtr.Zero/*NULL*/);  // ptsExpiry
            int rv =
Static.MapInitializeSecurityContextStatusToError(status);
                  out_buffer =
(SecBuffer)Marshal.PtrToStructure(out_buffer_desc.pBuffers,
typeof(SecBuffer));
                  if (rv != (int)NetErrors.Error.OK)
            {
                ResetSecurityContext();
```

```
      Marshal.FreeHGlobal(out_buffer.pvBuffer);//free(out_buffer.pvBuffer);
                return rv;
            }
            if (out_buffer.cbBuffer == 0)
            {
      Marshal.FreeHGlobal(out_buffer.pvBuffer);//free(out_buffer.pvBuffer);
                out_buffer.pvBuffer = IntPtr.Zero;
            }
            out_token = out_buffer.pvBuffer;
            out_token_len = out_buffer.cbBuffer;
            return (int)NetErrors.Error.OK;
        }
            private void ResetSecurityContext()
            {
                  if (SecIsValidHandle(ctxt_))
                  {
                        library_.DeleteSecurityContext(ref ctxt_);
                        SecInvalidateHandle(ref ctxt_);
                  }
            }
            private SSPILibrary library_;
            private string scheme_;
            private /*SEC_WCHAR* */string security_package_;
            private /*string*/Byte[] decoded_server_auth_token_ =
/*String.Empty*/new Byte[0];
            private UInt32 max_token_length_;
            private CredHandle cred_;
            private CtxtHandle ctxt_;
            private bool can_delegate_;
            #region IDisposable Support
            private bool disposedValue = false;
            protected virtual void Dispose(bool disposing)
            {
                  if (!disposedValue)
                  {
                        if (disposing)
                        {
                              ResetSecurityContext();
                              if (SecIsValidHandle(cred_))
                              {
                                    library_.FreeCredentialsHandle(ref cred_);
                                    SecInvalidateHandle(ref cred_);
                              }
                        }
                        disposedValue = true;
                  }
            }
            public void Dispose()
            {
                  Dispose(true);
            }
            #endregion
      }

}
```

Листинг Б.3 – HttpAuthHandlerNTLM.cs

```
#define  NTLM_SSPI //SELF
using Blink.URL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Blink.External.Net
{
```

```
    public partial class HttpAuthHandlerNTLM : HttpAuthHandler
    {
        public static Factory CreateFactory()
        {
#if OS_WIN || OS_QPOS
          return new FactoryWin();
#else
            throw new NotImplementedException();
#endif
        }
        public abstract class Factory : HttpAuthHandlerFactory
        {
            protected Factory()
            {
            }
#if NTLM_SSPI
                  public void set_sspi_library(SSPILibrary sspi_library)
                  {
                        ScopedObjImpl.Reset(ref sspi_library_, sspi_library);
                  }
#endif  // NTLM_SSPI
#if NTLM_SSPI
            protected UInt32 max_token_length_;
            protected bool first_creation_;
            protected bool is_unsupported_;
                  protected /*scoped_ptr*/SSPILibrary sspi_library_;
#endif  // NTLM_SSPI
        }
#if NTLM_PORTABLE
            HttpAuthHandlerNTLM();
#endif
#if NTLM_SSPI
           public HttpAuthHandlerNTLM(SSPILibrary sspi_library, UInt64
max_token_length, URLSecurityManager url_security_manager)
        {
            auth_sspi_ = new HttpAuthSSPI(sspi_library, "NTLM", "NTLM",
(UInt32)max_token_length);
            url_security_manager_ = url_security_manager;
        }
        public override bool NeedsIdentity()
        {
            return auth_sspi_.NeedsIdentity();
        }
        public override bool AllowsDefaultCredentials()
        {
            if (target_ == HttpAuth.Target.AUTH_PROXY)
                return true;
            if (url_security_manager_ == null)
                return false;
            return url_security_manager_.CanUseDefaultCredentials(origin_);
        }
#endif
        public override HttpAuth.AuthorizationResult
HandleAnotherChallenge(HttpAuthChallengeTokenizer challenge)
        {
            return ParseChallenge(challenge, false);
        }
        protected override bool Init(HttpAuthChallengeTokenizer tok)
        {
            auth_scheme_ = HttpAuth.Scheme.AUTH_SCHEME_NTLM;
            score_ = 3;
            properties_ = (int)Property.ENCRYPTS_IDENTITY |
(int)Property.IS_CONNECTION_BASED;
            return ParseChallenge(tok, true) ==
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_ACCEPT;
```

```
        }
        protected override int GenerateAuthTokenImpl(
            AuthCredentials credentials,
            HttpRequestInfo request,
            CompletionCallback callback,
            /*ref*/ Ref<String> auth_token)
        {
            #if NTLM_SSPI
              return auth_sspi_.GenerateAuthToken(
                  credentials,
                  CreateSPN(origin_),
                  auth_token);
            #else  // !NTLM_SSPI
                  // TODO(cbentzel): Shouldn't be hitting this case.
                  if (!credentials)
                  {
                        LOG(ERROR) << "Username and password are expected to
be non-null.";
                        return ERR_MISSING_AUTH_CREDENTIALS;
                  }
                  const void* in_buf;
                  void* out_buf;
                  uint32 in_buf_len, out_buf_len;
                  String decoded_auth_data;
                  String domain;
                  String user;
                  const String&username = credentials.username();
                  const Base.char16 backslash_character = '\\';
                  size_t backslash_idx = username.find(backslash_character);
                  if (backslash_idx == String::npos)
                  {
                        user = username;
                  }
                  else
                  {
                        domain = username.substr(0, backslash_idx);
                        user = username.substr(backslash_idx + 1);
                  }
                  domain_ = domain;
                  credentials_.Set(user, credentials.password());
                  if (auth_data_.empty())
                  {
                        in_buf_len = 0;
                        in_buf = null;
                        int rv = InitializeBeforeFirstChallenge();
                        if (rv != OK)
                              return rv;
                  }
                  else
                  {
                        if (!Base.Base64Decode(auth_data_,
&decoded_auth_data))
                        {
                              LOG(ERROR) << "Unexpected problem Base64
decoding.";
                              return ERR_UNEXPECTED;
                        }
                        in_buf_len = decoded_auth_data.length();
                        in_buf = decoded_auth_data.data();
                  }
                  int rv = GetNextToken(in_buf, in_buf_len, &out_buf,
&out_buf_len);
                  if (rv != OK)
                        return rv;
```

```
                String encode_input(static_cast<char*>(out_buf),
out_buf_len);
                String encode_output;
                Base.Base64Encode(encode_input, &encode_output);
                // OK, we are done with |out_buf|
                free(out_buf);
                *auth_token = String("NTLM ") + encode_output;
                return OK;
            #endif
        }
        public override void Dispose()
        {
            GC.SuppressFinalize(this);
            Dispose(true);
        }
        private /*virtual*/ void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (auth_sspi_ != null)
                {
                    auth_sspi_.Dispose();
                    auth_sspi_ = null;
                }
            }
        }
        ~HttpAuthHandlerNTLM()
        {
            Dispose(false);
        }
#if NTLM_PORTABLE
            private static GenerateRandomProc
SetGenerateRandomProc(GenerateRandomProc proc);
            private static HostNameProc SetHostNameProc(HostNameProc proc);
#endif
        private HttpAuth.AuthorizationResult
ParseChallenge(HttpAuthChallengeTokenizer tok, bool initial_challenge)
        {
            #if NTLM_SSPI
              return auth_sspi_.ParseChallenge(tok);
            #else
                  auth_data_.clear();
                  if (!LowerCaseEqualsASCII(tok.scheme(), "ntlm"))
                        return HttpAuth::AUTHORIZATION_RESULT_INVALID;
                  String base64_param = tok.base64_param();
                  if (base64_param.empty())
                  {
                        if (!initial_challenge)
                              return HttpAuth::AUTHORIZATION_RESULT_REJECT;
                        return HttpAuth::AUTHORIZATION_RESULT_ACCEPT;
                  }
                  else
                  {
                        if (initial_challenge)
                              return HttpAuth::AUTHORIZATION_RESULT_INVALID;
                  }
                  auth_data_ = base64_param;
                  return HttpAuth::AUTHORIZATION_RESULT_ACCEPT;
            #endif  // defined(NTLM_SSPI)
        }
        private static String CreateSPN(GURL origin)
        {
            String Target = "HTTP/";
            Target += NetUtil.GetHostAndPort(origin);
            return Target;
```

```
        }
#if NTLM_SSPI
            private HttpAuthSSPI auth_sspi_;
#endif
#if NTLM_PORTABLE
            private static GenerateRandomProc generate_random_proc_;
            private static HostNameProc get_host_name_proc_;
#endif
        private  String domain_;
        private  AuthCredentials credentials_;
        private String auth_data_;
#if NTLM_SSPI
        private URLSecurityManager url_security_manager_;
#endif
    }
}
```

Листинг Б.4 – HttpAuthHandlerNegotiate.cs

```
using Cryptosoft.Titan.Base.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using Blink.Core;
using Blink.URL;
namespace Blink.External.Net
{
#if OS_WIN
      using AuthLibrary = SSPILibrary;
      using AuthSystem = HttpAuthSSPI;
#elif OS_POSIX
  typedef GSSAPILibrary AuthLibrary;
  typedef HttpAuthGSSAPI AuthSystem;
#endif
      public class HttpAuthHandlerNegotiate : HttpAuthHandler
      {
            public class Factory : HttpAuthHandlerFactory
            {
                  public Factory()
                  {
                        disable_cname_lookup_ = false;
                        use_port_ = false;
                        resolver_ = null;
#if OS_WIN
                        max_token_length_ = 0;
                        first_creation_ = true;
#endif
                        is_unsupported_ = false;
                  }
                  public bool disable_cname_lookup() { return
disable_cname_lookup_; }

                  public void set_disable_cname_lookup(bool
disable_cname_lookup)
                  {
                        disable_cname_lookup_ = disable_cname_lookup;
                  }
                  public bool use_port() { return use_port_; }
                  public void set_use_port(bool use_port) { use_port_ =
use_port; }
                  public void set_host_resolver(HostResolver host_resolver)
                  {
                        resolver_ = host_resolver;
                  }
                  public void set_library(AuthLibrary auth_library)
                  {
                        auth_library_ = auth_library;
```

```
                }
                public override int
CreateAuthHandler(HttpAuthChallengeTokenizer challenge,
                                  HttpAuth.Target target,
                                  GURL origin,
                                  CreateReason reason,
                                  int digest_nonce_count,
                                  BoundNetLog net_log,
                                  out HttpAuthHandler handler)
                {
                      handler = null;
#if OS_WIN
                      if (is_unsupported_ || reason ==
CreateReason.CREATE_PREEMPTIVE)
                            return
(int)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME;
                      if (max_token_length_ == 0)
                      {
                            int rv =
Static.DetermineMaxTokenLength(auth_library_, WinAPI.NEGOSSP_NAME,
                                                            ref max_token_length_);
                            if (rv ==
(int)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME)
                                  is_unsupported_ = true;
                            if (rv != (int)NetErrors.Error.OK)
                                  return rv;
                      }
                      HttpAuthHandler tmp_handler = new
HttpAuthHandlerNegotiate(auth_library_, max_token_length_,

url_security_manager(), resolver_,

disable_cname_lookup_, use_port_);
                      if (!tmp_handler.InitFromChallenge(challenge, target,
origin, net_log))
                            return
(int)NetErrors.Error.ERR_INVALID_RESPONSE;
                      handler = tmp_handler;
                      return (int)NetErrors.Error.OK;
#elif OS_POSIX
                        if (is_unsupported_)
                            return ERR_UNSUPPORTED_AUTH_SCHEME;
                        if (!auth_library_.Init()) {
                            is_unsupported_ = true;
                            return ERR_UNSUPPORTED_AUTH_SCHEME;
                        }
                        scoped_ptr<HttpAuthHandler> tmp_handler(
                              new
HttpAuthHandlerNegotiate(auth_library_.get(), url_security_manager(),

resolver_, disable_cname_lookup_,

use_port_));
                        if (!tmp_handler.InitFromChallenge(challenge,
target, origin, net_log))
                            return ERR_INVALID_RESPONSE;
                        handler.swap(tmp_handler);
                        return OK;
#endif
                }
                private bool disable_cname_lookup_;
                private bool use_port_;
                private HostResolver resolver_;
#if OS_WIN
                private UInt32 max_token_length_;
```

```
                  private bool first_creation_;
#endif
                  private bool is_unsupported_;
                  private AuthLibrary auth_library_;
            }
            public HttpAuthHandlerNegotiate(AuthLibrary auth_library,
#if OS_WIN
                                            UInt32 max_token_length,
#endif
                                            URLSecurityManager
url_security_manager,
                                            HostResolver host_resolver,
                                            bool disable_cname_lookup,
                                            bool use_port)
            {
#if OS_WIN
                  auth_system_ = new AuthSystem(auth_library, "Negotiate",
WinAPI.NEGOSSP_NAME, max_token_length);
#elif OS_POSIX
    auth_system_(auth_library, "Negotiate", CHROME_GSS_SPNEGO_MECH_OID_DESC),
#endif
                  disable_cname_lookup_ = disable_cname_lookup;
                  use_port_ = use_port;
                  resolver_ = host_resolver;
                  already_called_ = false;
                  has_credentials_ = false;
                  auth_token_ = new Ref<String>(null);//null;
                  next_state_ = State.STATE_NONE;
                  url_security_manager_ = url_security_manager;
            }
            public override void Dispose()
            {
                  GC.SuppressFinalize(this);
                  Dispose(true);
            }
            private /*virtual*/ void Dispose(bool disposing)
            {
                  if (disposing)
                  {
                        if (auth_system_ != null)
                        {
                              auth_system_.Dispose();
                              auth_system_ = null;
                        }
                        if (single_resolve_ != null)
                        {
                              single_resolve_.Dispose();
                              single_resolve_ = null;
                        }
                  }
            }
            ~HttpAuthHandlerNegotiate()
            {
                  Dispose(false);
            }
            internal string CreateSPN(AddressList address_list, GURL origin)
            {
                  int port = origin.EffectiveIntPort();
                  string server = address_list.canonical_name();
                  if (server.isEmpty())
                        server = origin.host();
#if OS_WIN
                  const char kSpnSeparator = '/';
#elif OS_POSIX
              static const char kSpnSeparator = '@';
```

```
#endif
                if (port != 80 && port != 443 && use_port_)
                {
                      return String.Format("HTTP{0}{1}:{2:D}",
kSpnSeparator, server, port);
                }
                else
                {
                      return String.Format("HTTP{0}{1}", kSpnSeparator,
server);
                }
            }
            public string spn() { return spn_; }
            public override HttpAuth.AuthorizationResult
HandleAnotherChallenge(HttpAuthChallengeTokenizer challenge)
            {
                  return auth_system_.ParseChallenge(challenge);
            }
            public override bool NeedsIdentity()
            {
                  return auth_system_.NeedsIdentity();
            }
            public override bool AllowsDefaultCredentials()
            {
                  if (target_ == Net.HttpAuth.Target.AUTH_PROXY)
                        return true;
                  if (url_security_manager_ == null)
                        return false;
                  return
url_security_manager_.CanUseDefaultCredentials(origin_);
            }
            public override bool AllowsExplicitCredentials()
            {
                  return auth_system_.AllowsExplicitCredentials();
            }
            protected override bool Init(HttpAuthChallengeTokenizer challenge)
            {
#if OS_POSIX
              if (!auth_system_.Init()) {
                  VLOG(1) << "can't initialize GSSAPI library";
                  return false;
              }
              if (!AllowsDefaultCredentials())
                  return false;
#endif
                  if (CanDelegate())
                        auth_system_.Delegate();
                  auth_scheme_ = Net.HttpAuth.Scheme.AUTH_SCHEME_NEGOTIATE;
                  score_ = 4;
                  properties_ = (int)(Property.ENCRYPTS_IDENTITY |
Property.IS_CONNECTION_BASED);
                  HttpAuth.AuthorizationResult auth_result =
                        auth_system_.ParseChallenge(challenge);
                  return (auth_result ==
Net.HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_ACCEPT);
            }
            protected override int GenerateAuthTokenImpl(AuthCredentials
credentials,
                                                         HttpRequestInfo request,
                                                         CompletionCallback
callback,
                                                         /*ref*/ Ref<String>
auth_token)
            {
                  Debug.Assert(callback_ == null);
```

```
                  Debug.Assert(auth_token_.Value == null);
                  auth_token_ = auth_token;
                  if (already_called_)
                  {
                        Debug.Assert((!has_credentials_ && credentials ==
null) ||
                                     (has_credentials_ &&
credentials.Equals(credentials_)));
                        next_state_ = State.STATE_GENERATE_AUTH_TOKEN;
                  }
                  else
                  {
                        already_called_ = true;
                        if (credentials != null)
                        {
                              has_credentials_ = true;
                              credentials_ = credentials;
                        }
                        next_state_ = State.STATE_RESOLVE_CANONICAL_NAME;
                  }
                  int rv = DoLoop((int)NetErrors.Error.OK);
                  if (rv == (int)NetErrors.Error.ERR_IO_PENDING)
                        callback_ = callback;
                  return rv;
            }
            private enum State
            {
                  STATE_RESOLVE_CANONICAL_NAME,
                  STATE_RESOLVE_CANONICAL_NAME_COMPLETE,
                  STATE_GENERATE_AUTH_TOKEN,
                  STATE_GENERATE_AUTH_TOKEN_COMPLETE,
                  STATE_NONE,
            }
            private void OnIOComplete(int result)
            {
            int rv = DoLoop(result);
                  if (rv != (int)NetErrors.Error.ERR_IO_PENDING)
                        DoCallback(rv);
            }
            private void DoCallback(int rv)
            {
                  Debug.Assert(rv != (int)NetErrors.Error.ERR_IO_PENDING);
                  Debug.Assert(callback_ != null);
                  CompletionCallback callback = callback_;
                  callback_ = null;//.Reset();
                  callback.Invoke(rv);
            }
            private int DoLoop(int result)
            {
                  Debug.Assert(next_state_ != State.STATE_NONE);
                  int rv = result;
                  do
                  {
                        State state = next_state_;
                        next_state_ = State.STATE_NONE;
                        switch (state)
                        {
                              case State.STATE_RESOLVE_CANONICAL_NAME:
                                    Debug.Assert((int)NetErrors.Error.OK ==
rv);
                                    rv = DoResolveCanonicalName();
                                    break;
                              case
State.STATE_RESOLVE_CANONICAL_NAME_COMPLETE:
                                    rv = DoResolveCanonicalNameComplete(rv);
```

```
                                    break;
                              case State.STATE_GENERATE_AUTH_TOKEN:
                                    Debug.Assert((int)NetErrors.Error.OK ==
rv);
                                    rv = DoGenerateAuthToken();
                                    break;
                              case State.STATE_GENERATE_AUTH_TOKEN_COMPLETE:
                                    rv = DoGenerateAuthTokenComplete(rv);
                                    break;
                              default:
                                    Lermontov.NotReached("bad state");
                                    rv = (int)NetErrors.Error.ERR_FAILED;
                                    break;
                        }
                  } while (rv != (int)NetErrors.Error.ERR_IO_PENDING &&
next_state_ != State.STATE_NONE);

                  return rv;
            }
            private int DoResolveCanonicalName()
            {
                  next_state_ = State.STATE_RESOLVE_CANONICAL_NAME_COMPLETE;
                  if (disable_cname_lookup_ || resolver_ == null)
                        return (int)NetErrors.Error.OK;
                  Debug.Assert(single_resolve_ == null);
                  HostResolver.RequestInfo info = new
HostResolver.RequestInfo(new HostPortPair(origin_.host(), 0));
      info.set_host_resolver_flags(HostResolverFlags.HOST_RESOLVER_CANONNAME);
                  single_resolve_ = new SingleRequestHostResolver(resolver_);
                  return single_resolve_.Resolve(
                        info,
                        RequestPriority.DEFAULT_PRIORITY,
                        address_list_,
                        this.OnIOComplete,
                        net_log_);
            }
            private int DoResolveCanonicalNameComplete(int rv)
            {
                  Debug.Assert((int)NetErrors.Error.ERR_IO_PENDING != rv);
                  if (rv != (int)NetErrors.Error.OK)
                  {
                        Debug.WriteLine($"VLOG(1) << Problem finding canonical
name for SPN for host {origin_.host()} : {NetErrors.ErrorToString(rv)}");
                        rv = (int)NetErrors.Error.OK;
                  }
                  next_state_ = State.STATE_GENERATE_AUTH_TOKEN;
                  spn_ = CreateSPN(address_list_, origin_);
                  address_list_ = new AddressList();
                  return rv;
            }
            private int DoGenerateAuthToken()
            {
                  next_state_ = State.STATE_GENERATE_AUTH_TOKEN_COMPLETE;
                  AuthCredentials credentials = has_credentials_ ?
credentials_ : null;
                  return auth_system_.GenerateAuthToken(credentials, spn_,
/*ref*/ auth_token_);
            }
            private int DoGenerateAuthTokenComplete(int rv)
            {
                  Debug.Assert((int)NetErrors.Error.ERR_IO_PENDING != rv);
                  auth_token_ = new Ref<String>(null); //null;
                  return rv;
            }
            private bool CanDelegate()
```

```
            {
                  if (target_ == Net.HttpAuth.Target.AUTH_PROXY)
                        return false;
                  if (url_security_manager_ == null)
                        return false;
                  return url_security_manager_.CanDelegate(origin_);
            }
            private AuthSystem auth_system_;
            private bool disable_cname_lookup_;
            private bool use_port_;
            private HostResolver resolver_;
            private AddressList address_list_ = new AddressList();
            private SingleRequestHostResolver single_resolve_;
            private bool already_called_;
            private bool has_credentials_;
            private AuthCredentials credentials_;
            private string spn_;
            private CompletionCallback callback_;
            private Ref<String> auth_token_;
            private State next_state_;
            private URLSecurityManager url_security_manager_;
      }
}
```

Листинг Б.5 – HttpAuthHandlerBasic.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Blink.External.Net
{
      public class HttpAuthHandlerBasic : HttpAuthHandler
      {
            public class Factory : HttpAuthHandlerFactory
            {
                  public Factory()
                  {
                  }
                  public override int CreateAuthHandler(
                        HttpAuthChallengeTokenizer challenge,
                        HttpAuth.Target target,
                        URL.GURL origin,
                        CreateReason reason,
                        int digest_nonce_count,
                        BoundNetLog net_log,
                        out HttpAuthHandler handler)
                  {
                        HttpAuthHandler tmp_handler = new
HttpAuthHandlerBasic();
                        if (!tmp_handler.InitFromChallenge(challenge, target,
origin, net_log))
                        {
                              handler = null;
                              return (int)
NetErrors.Error.ERR_INVALID_RESPONSE;
                        }
                        handler = tmp_handler;
                        return (int)NetErrors.Error.OK;
                  }
            }
            public override HttpAuth.AuthorizationResult
HandleAnotherChallenge(HttpAuthChallengeTokenizer challenge)
            {
                  String realm;
```

```
                  if (!ParseRealm(challenge, out realm))
                        return
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID;
                  return (realm_ != realm) ?
      HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_DIFFERENT_REALM :
      HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT;
            }
        public override void Dispose()
        {
        }
        protected override bool Init(HttpAuthChallengeTokenizer challenge)
            {
                  auth_scheme_ = HttpAuth.Scheme.AUTH_SCHEME_BASIC;
                  score_ = 1;
                  properties_ = 0;
                  return ParseChallenge(challenge);
            }
            protected override int GenerateAuthTokenImpl(
                  AuthCredentials credentials,
                  HttpRequestInfo request,
                  CompletionCallback callback,
                  /*ref*/ Ref<String> auth_token)
            {
                  Debug.Assert(credentials != null);
                  String base64_username_password;
                  List<Byte> DataToEncode = new List<Byte>();
      DataToEncode.AddRange(Encoding.UTF8.GetBytes(credentials.username()));
                  DataToEncode.AddRange(Encoding.ASCII.GetBytes(":"));
      DataToEncode.AddRange(Encoding.UTF8.GetBytes(credentials.password()));
                  base64_username_password =
Convert.ToBase64String(DataToEncode.ToArray());
                  auth_token.Value = "Basic " + base64_username_password;
                  return (Int32)NetErrors.Error.OK;
            }
            private bool ParseChallenge(HttpAuthChallengeTokenizer challenge)
            {
                  if (String.Compare(challenge.scheme(), "basic",
StringComparison.InvariantCultureIgnoreCase) != 0)
                        return false;
                  String realm;
                  if (!ParseRealm(challenge, out realm))
                        return false;
                  realm_ = realm;
                  return true;
            }
            private static bool ParseRealm(HttpAuthChallengeTokenizer
tokenizer, out String realm)
            {
                realm = String.Empty;
            var parameters = tokenizer.param_pairs();
                while (parameters.GetNext())
                {
                if (String.Compare(parameters.name(), "realm",
StringComparison.InvariantCultureIgnoreCase) != 0)
                    continue;
                realm = parameters.value().Normalize();
                if (!realm.IsNormalized())
                    {
                        return false;
                }
                }
            return parameters.valid();
            }
      }
}
```

Листинг Б.6 – HttpAuthHandlerDigest.cs

```
using Cryptosoft.Titan.Base.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Blink.Core;
using static Blink.External.Net.Static;
namespace Blink.External.Net
{
      public class HttpAuthHandlerDigest : HttpAuthHandler
      {
            public interface NonceGenerator
            {
                  String GenerateNonce();
            }
            public class DynamicNonceGenerator : NonceGenerator
            {
                  private const String domain = "0123456789abcdef";
                  private Random mRandom = new
Random((Int32)DateTime.Now.Ticks);
                  public DynamicNonceGenerator()
                  {
                  }
                  public String GenerateNonce()
                  {
                        List<Char> cnonce = new List<Char>(16);
                        for (int i = 0; i < 16; ++i)
                              cnonce.Add(domain[mRandom.Next(0, 15)]);
                        return new String(cnonce.ToArray());
                  }
            }
            public class FixedNonceGenerator : NonceGenerator
            {
                  public FixedNonceGenerator(String nonce)
                  {
                        nonce_ = nonce;
                  }
                  public String GenerateNonce()
                  {
                        return nonce_;
                  }
                  private readonly String nonce_;
            }
            public class Factory : HttpAuthHandlerFactory
            {
                  public Factory()
                  {
                        nonce_generator_ = new DynamicNonceGenerator();
                  }
                  public void set_nonce_generator(NonceGenerator
nonce_generator)
                  {
                        nonce_generator_/*.reset*/ = nonce_generator;
                  }
                  public override int CreateAuthHandler(
                        HttpAuthChallengeTokenizer challenge,
                        HttpAuth.Target target,
                        URL.GURL origin,
                        CreateReason reason,
                        int digest_nonce_count,
                        BoundNetLog net_log,
                        out HttpAuthHandler handler)
```

```
                  {
                        /*scoped_ptr*/HttpAuthHandler tmp_handler = new
HttpAuthHandlerDigest(digest_nonce_count, nonce_generator_);
                        if (!tmp_handler.InitFromChallenge(challenge, target,
origin, net_log))
                        {
                              handler = null;
                              return (int)
NetErrors.Error.ERR_INVALID_RESPONSE;
                        }
                        handler = tmp_handler;
                        return (int)NetErrors.Error.OK;
                  }
                  private /*scoped_ptr*/NonceGenerator nonce_generator_;
            }
            public override HttpAuth.AuthorizationResult
HandleAnotherChallenge(HttpAuthChallengeTokenizer challenge)
            {
                  if (String.Compare(challenge.scheme(), "digest",
StringComparison.InvariantCultureIgnoreCase) != 0)
                        return
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_INVALID;
                  HttpUtil.NameValuePairsIterator parameters =
challenge.param_pairs();
                  String original_realm = String.Empty;
                  while (parameters.GetNext())
                  {
                        if (String.Compare(parameters.name(), "stale",
StringComparison.InvariantCultureIgnoreCase) == 0)
                        {
                              if (String.Compare(parameters.value(), "true",
StringComparison.InvariantCultureIgnoreCase) == 0)
                                    return
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_STALE;
                        }
                        else if (String.Compare(parameters.name(), "realm",
StringComparison.InvariantCultureIgnoreCase) == 0)
                        {
                              original_realm = parameters.value();
                        }
                  }
                  return (original_realm_ != original_realm) ?
      HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_DIFFERENT_REALM :
      HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT;
            }
        public override void Dispose(){}
        protected override bool Init(HttpAuthChallengeTokenizer challenge)
            {
                  return ParseChallenge(challenge);
            }
            protected override int GenerateAuthTokenImpl(
                  AuthCredentials credentials,
                  HttpRequestInfo request,
                  CompletionCallback callback,
                  /*ref*/ Ref<String> auth_token)
            {
                  String cnonce = nonce_generator_.GenerateNonce();
                  String method;
                  String path;
                  GetRequestMethodAndPath(request, out method, out path);
                  auth_token.Value = AssembleCredentials(method, path,
credentials, cnonce, nonce_count_);
                  return (int)NetErrors.Error.OK;
            }
            public enum DigestAlgorithm
```

```
            {
                  ALGORITHM_UNSPECIFIED,
                  ALGORITHM_MD5,
                  ALGORITHM_MD5_SESS,
            }
          public enum QualityOfProtection
            {
                  QOP_UNSPECIFIED,
                  QOP_AUTH,
            }
            private HttpAuthHandlerDigest(int nonce_count, NonceGenerator
nonce_generator)
            {
                  stale_ = false;
                  algorithm_ = DigestAlgorithm.ALGORITHM_UNSPECIFIED;
                  qop_ = QualityOfProtection.QOP_UNSPECIFIED;
                  nonce_count_ = nonce_count;
                  nonce_generator_ = nonce_generator;
                  Debug.Assert(nonce_generator_ != null);
            }
            private bool ParseChallenge(HttpAuthChallengeTokenizer challenge)
            {
                  auth_scheme_ = HttpAuth.Scheme.AUTH_SCHEME_DIGEST;
                  score_ = 2;
                  properties_ = (Int32)Property.ENCRYPTS_IDENTITY;
                  stale_ = false;
                  algorithm_ = DigestAlgorithm.ALGORITHM_UNSPECIFIED;
                  qop_ = QualityOfProtection.QOP_UNSPECIFIED;
                  realm_ = original_realm_ = nonce_ = domain_ = opaque_ =
String.Empty;
                  if (String.Compare(challenge.scheme(), "digest",
StringComparison.InvariantCultureIgnoreCase) != 0)
                        return false;
                  HttpUtil.NameValuePairsIterator parameters =
challenge.param_pairs();
                  while (parameters.GetNext())
                  {
                        if (!ParseChallengeProperty(parameters.name(),
                                    parameters.value()))
                              return false;
                  }
                  if (!parameters.valid())
                        return false;
                  if (String.IsNullOrEmpty(nonce_))
                        return false;
                  return true;
            }
        private bool ParseChallengeProperty(String name, String value)
            {
                  if (String.Compare(name, "realm",
StringComparison.InvariantCultureIgnoreCase) == 0)
                  {
                String realm = value.Normalize();
                     if (!realm.IsNormalized())
                         return false;
                     realm_ = realm;
                     original_realm_ = value;
            }
                  else if (String.Compare(name, "nonce",
StringComparison.InvariantCultureIgnoreCase) == 0)
                  {
                        nonce_ = value;
                  }
                  else if (String.Compare(name, "domain",
StringComparison.InvariantCultureIgnoreCase) == 0)
```

```
                {
                      domain_ = value;
                }
                else if (String.Compare(name, "opaque",
StringComparison.InvariantCultureIgnoreCase) == 0)
                {
                      opaque_ = value;
                }
                else if (String.Compare(name, "stale",
StringComparison.InvariantCultureIgnoreCase) == 0)
                {
                      stale_ = String.Compare(value, "true",
StringComparison.InvariantCultureIgnoreCase) == 0;
                }
                else if (String.Compare(name, "algorithm",
StringComparison.InvariantCultureIgnoreCase) == 0)
                {
                      if (String.Compare(value, "md5",
StringComparison.InvariantCultureIgnoreCase) == 0)
                      {
                            algorithm_ = DigestAlgorithm. ALGORITHM_MD5;
                      }
                      else if (String.Compare(value, "md5-sess",
StringComparison.InvariantCultureIgnoreCase) == 0)
                      {
                            algorithm_ = DigestAlgorithm.ALGORITHM_MD5_SESS;
                      }
                      else
                      {
                          Lermontov.DVLog(1, "Unknown value of algorithm");
                    return false;
                      }
                }
                else if (String.Compare(name, "qop",
StringComparison.InvariantCultureIgnoreCase) == 0)
                {
                      HttpUtil.ValuesIterator qop_values = new
HttpUtil.ValuesIterator(value, 0, value.Length, ',');
                      qop_ = QualityOfProtection.QOP_UNSPECIFIED;
                      while (qop_values.GetNext())
                      {
                            if (String.Compare(qop_values.value(), "auth",
StringComparison.InvariantCultureIgnoreCase) == 0)
                            {
                                  qop_ = QualityOfProtection.QOP_AUTH;
                                  break;
                            }
                      }
                }
                else
                {
                    Lermontov.DVLog(1, "Skipping unrecognized digest
property");
            }
                      return true;
                }
                private static String QopToString(QualityOfProtection qop)
                {
                      switch (qop)
                      {
                            case QualityOfProtection.QOP_UNSPECIFIED:
                                  return String.Empty;
                            case QualityOfProtection.QOP_AUTH:
                                  return "auth";
                            default:
```

```
                        Lermontov.NotReached();
                        return String.Empty;
                }
            }
            private static String AlgorithmToString(DigestAlgorithm algorithm)
            {
                switch (algorithm)
                {
                    case DigestAlgorithm.ALGORITHM_UNSPECIFIED:
                        return String.Empty;
                    case DigestAlgorithm.ALGORITHM_MD5:
                        return "MD5";
                    case DigestAlgorithm.ALGORITHM_MD5_SESS:
                        return "MD5-sess";
                    default:
                        Lermontov.NotReached();
                        return String.Empty;
                }
            }
            private void GetRequestMethodAndPath(HttpRequestInfo request, out
String method, out String path)
            {
                Debug.Assert(request != null);
                URL.GURL url = request.url;
                if (target_ == HttpAuth.Target.AUTH_PROXY &&
(url.SchemeIs("https") || url.SchemeIsWSOrWSS()))
                {
                    method = "CONNECT";
                    path = NetUtil.GetHostAndPort(url);
                }
                else
                {
                    method = request.method;
                    path = HttpUtil.PathForRequest(url);
                }
            }
            private String AssembleResponseDigest(
                String method,
                String path,
                AuthCredentials credentials,
                String cnonce,
                String nc)
            {
            String ha1 = Base.MD5.MD5String(credentials.username() + ":" +
original_realm_ + ":" + credentials.password());

            if (algorithm_ == DigestAlgorithm.ALGORITHM_MD5_SESS)
                ha1 = Base.MD5.MD5String(ha1 + ":" + nonce_ + ":" + cnonce);
            String ha2 = Base.MD5.MD5String(method + ":" + path);
            String nc_part = String.Empty;
            if (qop_ != QualityOfProtection.QOP_UNSPECIFIED)
            {
                nc_part = nc + ":" + cnonce + ":" + QopToString(qop_) + ":";
            }
            return Base.MD5.MD5String(ha1 + ":" + nonce_ + ":" + nc_part +
ha2);
        }
        private String AssembleCredentials(
                String method,
                String path,
                AuthCredentials credentials,
                String cnonce,
                int nonce_count)
            {
                String nc = $"{nonce_count:x8}";
```

```
                  StringBuilder authorization = new StringBuilder();
                  authorization.Append("Digest username=");
      authorization.Append(HttpUtil.Quote(/*Base.UTF16ToUTF8*/(credentials.use
rname())));                                ;
                  authorization.Append(", realm=");
                  authorization.Append(HttpUtil.Quote(original_realm_));
                  authorization.Append(", nonce=");
                  authorization.Append(HttpUtil.Quote(nonce_));
                  authorization.Append(", uri=");
                  authorization.Append(HttpUtil.Quote(path));
                  if (algorithm_ != DigestAlgorithm.ALGORITHM_UNSPECIFIED)
                  {
                        authorization.Append(", algorithm=");
                        authorization.Append(AlgorithmToString(algorithm_));
                  }
                  String response = AssembleResponseDigest(method, path,
credentials, cnonce, nc);
                  authorization.Append(", response=\"");
                  authorization.Append(response);
                  authorization.Append("\"");
                  if (!String.IsNullOrEmpty(opaque_))
                  {
                        authorization.Append(", opaque=");
                        authorization.Append(HttpUtil.Quote(opaque_));
                  }
                  if (qop_ != QualityOfProtection.QOP_UNSPECIFIED)
                  {
                        authorization.AppendFormat(", qop={0}",
QopToString(qop_));
                        authorization.AppendFormat(", nc={0}", nc);
                        authorization.AppendFormat(", cnonce={0}",
HttpUtil.Quote(cnonce));
                  }
                  return authorization.ToString();
            }
          public String nonce_ { get; private set; }
            public String domain_ { get; private set; }
          public String opaque_ { get; private set; }
          public bool stale_ { get; private set; }
          public DigestAlgorithm algorithm_ { get; private set; }
          public QualityOfProtection qop_ { get; private set; }
        String original_realm_;
            int nonce_count_;
            NonceGenerator nonce_generator_;
      }
}
```

Приложение В

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

(рекомендуемое)

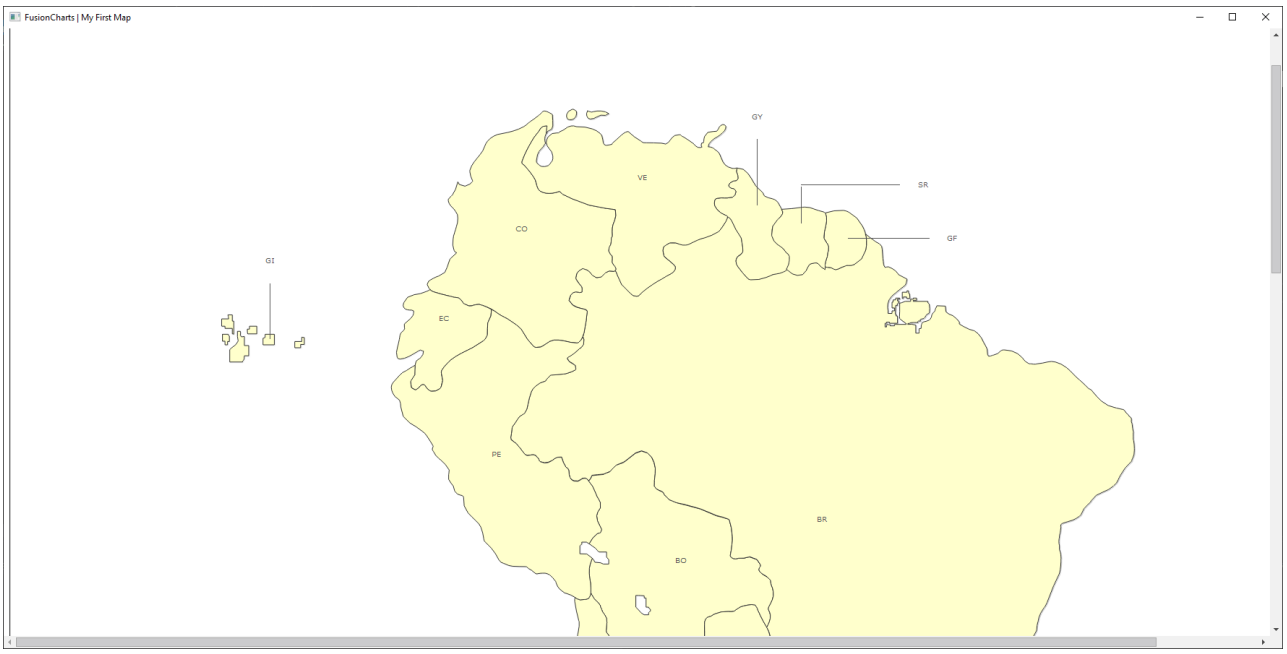

Рисунок В.1 – Страница доступа к ресурсу

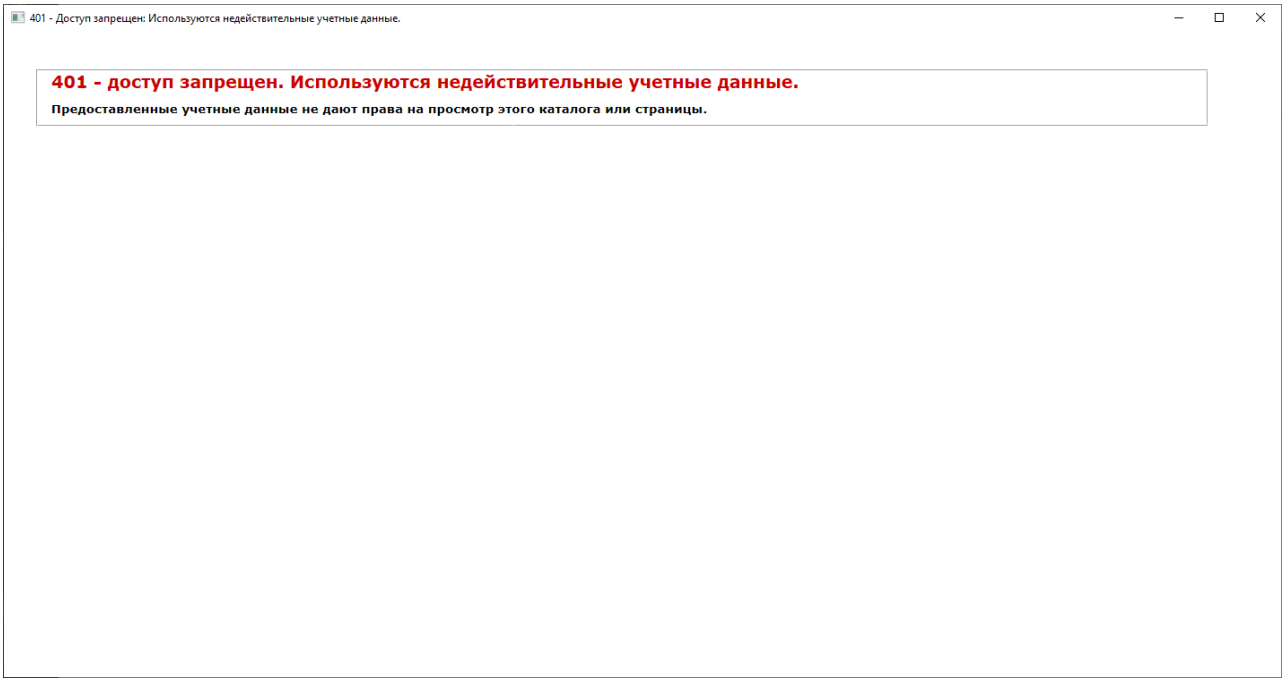

Рисунок В.2 – Страница отказа в доступе

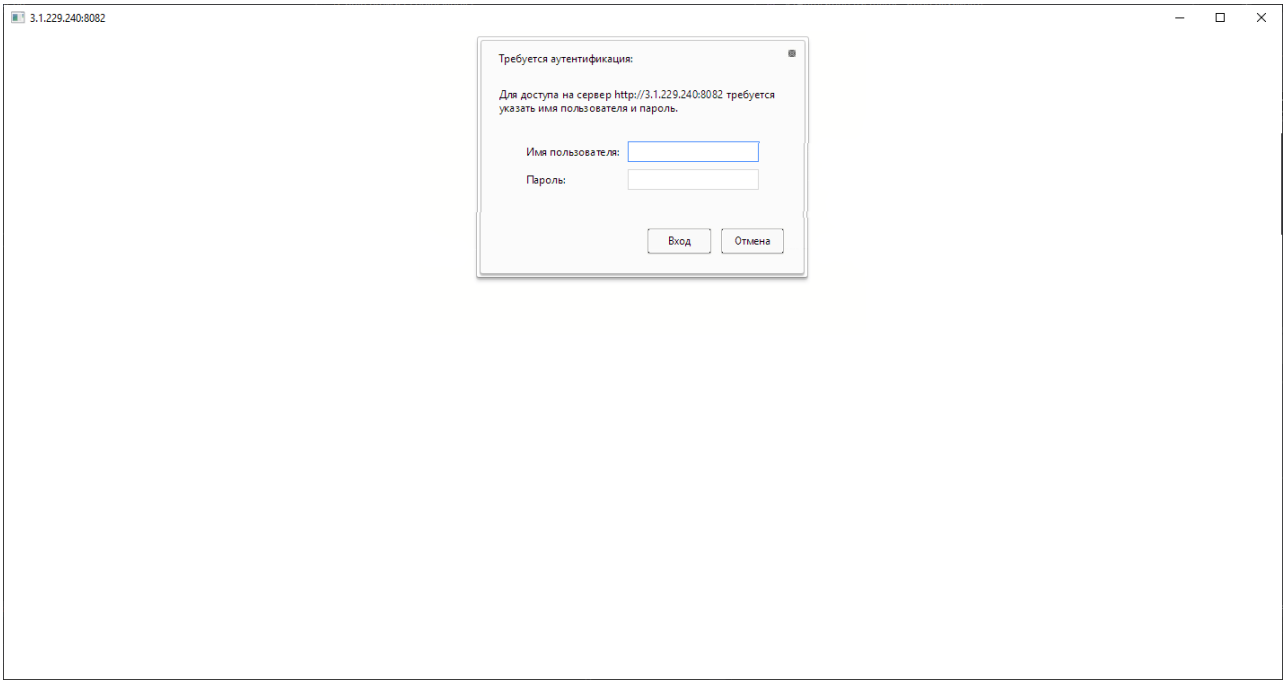

Рисунок В.3 – Страница аутентификации

Приложение Г

ИСХОДНЫЙ КОД ФУНКЦИЙ ТЕСТИРОВАНИЯ

(рекомендуемое)

Листинг Г.1 – HttpAuthControllerTest.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Blink.URL;
using Blink.External.Net;
using Blink.External;
namespace BlinkUnitTest.Blink.External.Net.Http
{
    [TestClass]
    public class HttpAuthControllerTest:Cryptosoft.Titan.TestBase
    {
        enum HandlerRunMode
        {
            RUN_HANDLER_SYNC,
            RUN_HANDLER_ASYNC
        }
        enum SchemeState
        {
            SCHEME_IS_DISABLED,
            SCHEME_IS_ENABLED
        }
        HttpResponseHeaders HeadersFromString(string str)
        {
            string raw_string = str;
            string headers_string = HttpUtil.AssembleRawHeaders(raw_string,
0, raw_string.Length);

            HttpResponseHeaders headers = new
HttpResponseHeaders(headers_string);
            return headers;
        }

        void RunSingleRoundAuthTest(HandlerRunMode run_mode, int handler_rv,
int expected_controller_rv, SchemeState scheme_state)
        {
            BoundNetLog dummy_log = new BoundNetLog();
            HttpAuthCache dummy_auth_cache = new HttpAuthCache();

            HttpRequestInfo request = new HttpRequestInfo();
            request.method = "GET";
            request.url = new GURL("http://example.com");

            HttpResponseHeaders headers = HeadersFromString("HTTP/1.1
407\r\n" + "Proxy-Authenticate: MOCK foo\r\n" + "\r\n");

            HttpAuthHandlerMock.Factory auth_handler_factory = new
HttpAuthHandlerMock.Factory();
            HttpAuthHandlerMock auth_handler = new HttpAuthHandlerMock();
            auth_handler.SetGenerateExpectation((run_mode ==
HandlerRunMode.RUN_HANDLER_ASYNC), handler_rv);
            auth_handler_factory.AddMockHandler(auth_handler,
HttpAuth.Target.AUTH_PROXY);
            auth_handler_factory.set_do_init_from_challenge(true);

            HttpAuthController controller = new
HttpAuthController(HttpAuth.Target.AUTH_PROXY,
                new GURL("http://example.com"),
                dummy_auth_cache, auth_handler_factory);
            Assert.AreEqual((Int32)NetErrors.Error.OK,
controller.HandleAuthChallenge(headers, false, false, dummy_log));
            Assert.IsTrue(controller.HaveAuthHandler());
            controller.ResetAuth(new AuthCredentials());
```

```
            Assert.IsTrue(controller.HaveAuth());

            TestCompletionCallback callback = new TestCompletionCallback();
            Assert.AreEqual((run_mode == HandlerRunMode.RUN_HANDLER_ASYNC) ?
(Int32)NetErrors.Error.ERR_IO_PENDING :
                    expected_controller_rv,
                controller.MaybeGenerateAuthToken(request,
callback.callback(),
                    dummy_log));
            if (run_mode == HandlerRunMode.RUN_HANDLER_ASYNC)
                Assert.AreEqual(expected_controller_rv,
callback.WaitForResult());
            Assert.AreEqual((scheme_state == SchemeState.SCHEME_IS_DISABLED),

controller.IsAuthSchemeDisabled(HttpAuth.Scheme.AUTH_SCHEME_MOCK));
        }

        [TestMethod]
        public void HttpAuthControllerTest_PermanentErrors()
        {
                RunSingleRoundAuthTest(HandlerRunMode.RUN_HANDLER_SYNC,

(Int32)NetErrors.Error.ERR_UNEXPECTED_SECURITY_LIBRARY_STATUS,
                    (Int32)NetErrors.Error.OK,
SchemeState.SCHEME_IS_DISABLED);
                RunSingleRoundAuthTest(HandlerRunMode.RUN_HANDLER_ASYNC,
                    (Int32)NetErrors.Error.ERR_MISSING_AUTH_CREDENTIALS,
                    (Int32)NetErrors.Error.OK,
SchemeState.SCHEME_IS_DISABLED);
                RunSingleRoundAuthTest(HandlerRunMode.RUN_HANDLER_ASYNC,
                    (Int32)NetErrors.Error.ERR_INVALID_AUTH_CREDENTIALS,
                    (Int32)NetErrors.Error.ERR_INVALID_AUTH_CREDENTIALS,
SchemeState.SCHEME_IS_ENABLED);
        }

        class MockHandler : HttpAuthHandlerMock
        {
            public MockHandler(int expected_rv, HttpAuth.Scheme scheme)
            {
                SetGenerateExpectation(false, expected_rv);
            }

            protected override bool Init(HttpAuthChallengeTokenizer
challenge)
            {
                base.Init(challenge);
                set_allows_default_credentials(true);
                set_allows_explicit_credentials(false);
                set_connection_based(true);
                if (challenge.scheme() == "Basic")
                {
                    auth_scheme_ = HttpAuth.Scheme.AUTH_SCHEME_BASIC;
                    --score_;
                    set_allows_explicit_credentials(true);
                }
                Assert.AreEqual(expected_scheme_, auth_scheme_);
                return true;
            }

            protected override int GenerateAuthTokenImpl(AuthCredentials
credentials,
                HttpRequestInfo request,
                CompletionCallback callback,
```

```
                Ref<String> auth_token)
            {
                int result = base.GenerateAuthTokenImpl(credentials, request,
callback, auth_token);
                Assert.IsTrue(result != (Int32)NetErrors.Error.OK ||
!AllowsExplicitCredentials() || !credentials.Empty());
                return result;
            }
            HttpAuth.Scheme expected_scheme_;
        }

        [TestMethod]
        public void HttpAuthControllerTest_NoExplicitCredentialsAllowed()
        {

            BoundNetLog dummy_log = new BoundNetLog();
            HttpAuthCache dummy_auth_cache = new HttpAuthCache();
            HttpRequestInfo request = new HttpRequestInfo();
            request.method = "GET";
            request.url = new GURL("http://example.com");

            HttpRequestHeaders request_headers = new HttpRequestHeaders();
            HttpResponseHeaders headers = HeadersFromString("HTTP/1.1
401\r\n" +
                "WWW-Authenticate: Mock\r\n" +
                "WWW-Authenticate: Basic\r\n" +
                "\r\n");
            HttpAuthHandlerMock.Factory auth_handler_factory = new
HttpAuthHandlerMock.Factory();

            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.OK,
HttpAuth.Scheme.AUTH_SCHEME_MOCK),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.ERR_UNEXPECTED,
HttpAuth.Scheme.AUTH_SCHEME_BASIC),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.ERR_UNEXPECTED,
HttpAuth.Scheme.AUTH_SCHEME_MOCK),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.ERR_UNEXPECTED,
HttpAuth.Scheme.AUTH_SCHEME_BASIC),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.ERR_UNEXPECTED,
HttpAuth.Scheme.AUTH_SCHEME_MOCK),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.AddMockHandler(
                new MockHandler((Int32)NetErrors.Error.OK,
HttpAuth.Scheme.AUTH_SCHEME_BASIC),
                HttpAuth.Target.AUTH_SERVER);
            auth_handler_factory.set_do_init_from_challenge(true);
            HttpAuthController controller = new HttpAuthController(
                HttpAuth.Target.AUTH_SERVER,
                new GURL("http://example.com"),
                dummy_auth_cache,
                auth_handler_factory);
            Assert.AreEqual((Int32)NetErrors.Error.OK,
controller.HandleAuthChallenge(headers, false, false, dummy_log));
            Assert.IsTrue(controller.HaveAuthHandler());
```

```
            controller.ResetAuth(new AuthCredentials());
            Assert.IsTrue(controller.HaveAuth());

            Assert.AreEqual((Int32)NetErrors.Error.OK,
controller.MaybeGenerateAuthToken(
            request, null, dummy_log));
            controller.AddAuthorizationHeader(request_headers);

            Assert.AreEqual((Int32)NetErrors.Error.OK,
            controller.HandleAuthChallenge(headers, false, false,
dummy_log));
            Assert.IsTrue(controller.HaveAuthHandler());
            controller.ResetAuth(new AuthCredentials("Hello", ""));
            Assert.IsTrue(controller.HaveAuth());

Assert.IsTrue(controller.IsAuthSchemeDisabled(HttpAuth.Scheme.AUTH_SCHEME_MOC
K));

Assert.IsFalse(controller.IsAuthSchemeDisabled(HttpAuth.Scheme.AUTH_SCHEME_BA
SIC));
            Assert.AreEqual((Int32)NetErrors.Error.OK,
controller.MaybeGenerateAuthToken(
            request, null, dummy_log));
        }
    }
}
```

Листинг Г.2 – HttpAuthHandlerBasicTest.cs

```
using System;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Blink.URL;
using Blink.External.Net;
using Blink.External;
namespace BlinkUnitTest.Blink.External.Net.Http
{
    [TestClass]
    public class HttpAuthHandlerBasicTest
    {
        [TestMethod]
        public void HttpAuthHandlerBasicTest_GenerateAuthToken()
        {
            var tests = new[]
            {
                new {username = "foo", password = "bar", expected_credentials
= "Basic Zm9vOmJhcg=="},
                new {username = "", password = "foobar", expected_credentials
= "Basic OmZvb2Jhcg=="},
                new {username = "anon", password = "", expected_credentials =
"Basic YW5vbjo="},
                new {username = "", password = "", expected_credentials =
"Basic Og=="}
            };
            GURL origin = new GURL("http://www.example.com");
            HttpAuthHandlerBasic.Factory factory = new
HttpAuthHandlerBasic.Factory();
            for (int i = 0; i < tests.Length; ++i)
            {
                string challenge = "Basic realm=\"Atlantis\"";
                HttpAuthHandler basic;
                Assert.AreEqual((Int32)(NetErrors.Error.OK),
                    factory.CreateAuthHandlerFromString(challenge,
HttpAuth.Target.AUTH_SERVER, origin,
                        new BoundNetLog(), out basic));
```

```
                AuthCredentials credentials =
                    new AuthCredentials(tests[i].username,
tests[i].password);
                HttpRequestInfo request_info = new HttpRequestInfo();
                Ref<String> auth_token = null;
                TestCompletionCallback callback = new
TestCompletionCallback();
                int rv = basic.GenerateAuthToken(credentials, request_info,
callback.callback(), auth_token);
                Assert.AreEqual((Int32)(NetErrors.Error.OK), rv);
                Assert.AreEqual(tests[i].expected_credentials, auth_token);
            }
        }
        [TestMethod]
        public void HttpAuthHandlerBasicTest_HandleAnotherChallenge()
        {
            var tests = new[]
            {
                new { challenge = "Basic realm=\"First\"",
                    expected_rv =
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT},
                new { challenge = "Basic realm=\"Second\"",
                    expected_rv =
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_DIFFERENT_REALM},
                new { challenge = "Basic realm=\"Second\",realm=\"First\"",
                    expected_rv =
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT},
                new { challenge = "basic realm=\"First\",realm=\"Second\"",
                    expected_rv =
HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_DIFFERENT_REALM}
            };
            GURL origin = new GURL("http://www.example.com");
            HttpAuthHandlerBasic.Factory factory = new
HttpAuthHandlerBasic.Factory();
            HttpAuthHandler basic;
            Assert.AreEqual((Int32)(NetErrors.Error.OK),
                factory.CreateAuthHandlerFromString(tests[0].challenge,
HttpAuth.Target.AUTH_SERVER, origin,
                    new BoundNetLog(), out basic));
            for (int i = 0; i < tests.Length; ++i)
            {
                string challenge = tests[i].challenge;
                HttpAuthChallengeTokenizer tok = new
HttpAuthChallengeTokenizer(challenge);
                Assert.AreEqual(tests[i].expected_rv,
basic.HandleAnotherChallenge(tok));
            }
        }
        [TestMethod]
        public void HttpAuthHandlerBasicTest_InitFromChallenge()
        {
            var tests = new[]
            {
                new {challenge = "Basic", expected_rv = (Int32)
(NetErrors.Error.OK), expected_realm = ""},
                new {challenge = "Basic realm=\"\"", expected_rv = (Int32)
(NetErrors.Error.OK), expected_realm = ""},
                new {challenge = "Basic realm=\"test_realm\"", expected_rv =
(Int32) (NetErrors.Error.OK), expected_realm = "test_realm"},
                new {challenge = "Basic
realm=\"test_realm\",unknown_token=foobar", expected_rv = (Int32)
(NetErrors.Error.OK), expected_realm = "test_realm"},
                new {challenge = "Basic
```

```
unknown_token=foobar,realm=\"test_realm\"", expected_rv = (Int32)
(NetErrors.Error.OK), expected_realm = "test_realm"},
                new {challenge = "Negotiate", expected_rv = (Int32)
(NetErrors.Error.ERR_INVALID_RESPONSE), expected_realm = ""},
                new {challenge = "Basic realm=\"foo\",realm=\"bar\"",
expected_rv = (Int32) (NetErrors.Error.OK), expected_realm = "bar"},
                new {challenge = "Basic realm=\"foo-\xE5\"", expected_rv =
(Int32) (NetErrors.Error.OK), expected_realm = "foo-\xC3\xA5"}
            };
            HttpAuthHandlerBasic.Factory factory = new
HttpAuthHandlerBasic.Factory();
            GURL origin = new GURL("http://www.example.com");
            for (int i = 0; i < tests.Length; ++i)
            {
                string challenge = tests[i].challenge;
                HttpAuthHandler basic;
                int rv = factory.CreateAuthHandlerFromString(challenge,
HttpAuth.Target.AUTH_SERVER, origin, new BoundNetLog(), out basic);
                Assert.AreEqual(tests[i].expected_rv, rv);
                if (rv == (Int32)(NetErrors.Error.OK))
                    Assert.AreEqual(tests[i].expected_realm, basic.realm());
            }
        }
    }
}
```

Листинг Г.3 – HttpAuthHandlerDigestTest.cs

```
using System;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Blink.URL;
using Blink.External.Net;
using Blink.External;
namespace BlinkUnitTest.Blink.External.Net.Http
{
    [TestClass]
    public class HttpAuthHandlerDigestTest
    {
        [TestMethod]
        public void HttpAuthHandlerDigestTest_ParseChallenge()
        {
            var tests = new[]
            {
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"Thunder
Bluff\"",
                    parsed_success = true,
                    parsed_realm = "Thunder Bluff",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=ThunderBluff",
                    parsed_success = true,
                    parsed_realm = "ThunderBluff",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
```

```
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\"",
                    parsed_success = true,
                    parsed_realm = "",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest realm=\"\", nonce=\"xyz\"",
                    parsed_success = true,
                    parsed_realm = "",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"foo-\xE5\"",
                    parsed_success = true,
                    parsed_realm = "foo-\xC3\xA5",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest realm=\"Thunder Bluff\"",
                    parsed_success = false,
                    parsed_realm = "",
                    parsed_nonce = "",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=xyz, realm=\"Thunder Bluff\"",
                    parsed_success = true,
                    parsed_realm = "Thunder Bluff",
                    parsed_nonce = "xyz",
```

```
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"Thunder
Bluff\", foo=\"bar\"",
                    parsed_success = true,
                    parsed_realm = "Thunder Bluff",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", algorithm=\"awezum\",
realm=\"Thunder\"",
                    parsed_success = false,
                    parsed_realm = "",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", algorithm=\"mD5\",
realm=\"Oblivion\"",
                    parsed_success = true,
                    parsed_realm = "Oblivion",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_MD5,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", algorithm=\"md5-
sess\", realm=\"Oblivion\"",
                    parsed_success = true,
                    parsed_realm = "Oblivion",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_MD5_SESS,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
```

```
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"Oblivion\",
qop=\"aUth\"",
                    parsed_success = true,
                    parsed_realm = "Oblivion",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_AUTH
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"Oblivion\",
qop=\"auth-int\"",
                    parsed_success = true,
                    parsed_realm = "Oblivion",
                    parsed_nonce = "xyz",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge =  "Digest nonce=\"xyz\", realm=\"Thunder
Bluff\", " +
                        "domain=\"http://intranet.example.com/protection\"",
                    parsed_success =  true,
                    parsed_realm =  "Thunder Bluff",
                    parsed_nonce =  "xyz",
                    parsed_domain =
"http://intranet.example.com/protection",
                    parsed_opaque =  "",
                    parsed_stale =  false,
                    parsed_algorithm =  (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop =  (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
                    challenge = "Digest nonce=\"xyz\", realm=\"Thunder
Bluff\", " +
                    "domain=\"http://intranet.example.com/protection
http://www.google.com\"",
                    parsed_success =  true,
                    parsed_realm =  "Thunder Bluff",
                    parsed_nonce =  "xyz",
                    parsed_domain =  "http://intranet.example.com/protection
http://www.google.com",
                    parsed_opaque =  "",
                    parsed_stale =  false,
                    parsed_algorithm =  (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop =  (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                },
                new {
```

```
                    challenge = "Basic realm=\"foo\"",
                    parsed_success = false,
                    parsed_realm = "",
                    parsed_nonce = "",
                    parsed_domain = "",
                    parsed_opaque = "",
                    parsed_stale = false,
                    parsed_algorithm = (Int32)
HttpAuthHandlerDigest.DigestAlgorithm.ALGORITHM_UNSPECIFIED,
                    parsed_qop = (Int32)
HttpAuthHandlerDigest.QualityOfProtection.QOP_UNSPECIFIED
                }
            };
            GURL origin = new GURL("http://www.example.com");
            HttpAuthHandlerDigest.Factory factory = new
HttpAuthHandlerDigest.Factory();
            for (int i = 0; i < tests.Length; ++i)
            {
                HttpAuthHandler handler;
                int rv =
factory.CreateAuthHandlerFromString(tests[i].challenge,
HttpAuth.Target.AUTH_SERVER, origin,
                    new BoundNetLog(), out handler);
                if (tests[i].parsed_success)
                {
                    Assert.AreEqual((Int32) NetErrors.Error.OK, rv);
                }
                else
                {
                    Assert.IsTrue((Int32) NetErrors.Error.OK != rv);
                    Assert.IsTrue(handler == null);
                    continue;
                }
                Assert.IsTrue(handler != null);
                HttpAuthHandlerDigest digest = (HttpAuthHandlerDigest)
handler;
                Assert.AreEqual(tests[i].parsed_realm, digest.realm());
                Assert.AreEqual(tests[i].parsed_nonce, digest.nonce_);
                Assert.AreEqual(tests[i].parsed_domain, digest.domain_);
                Assert.AreEqual(tests[i].parsed_opaque, digest.opaque_);
                Assert.AreEqual(tests[i].parsed_stale, digest.stale_);
                Assert.AreEqual(tests[i].parsed_algorithm,
(Int32)digest.algorithm_);
                Assert.AreEqual(tests[i].parsed_qop, (Int32)digest.qop_);
                Assert.IsTrue(handler.encrypts_identity());
                Assert.IsFalse(handler.is_connection_based());
                Assert.IsTrue(handler.NeedsIdentity());
                Assert.IsFalse(handler.AllowsDefaultCredentials());
            }
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_AssembleCredentials()
        {
            var tests = new[] {
                new {
                    req_method = "GET",
                    req_path = "/test/drealm1",
                    challenge = "Digest realm=\"DRealm1\", "+
           nonce=\"claGgoRXBAA=7583377687842fdb7b56ba0555d175baa0b800e3\", "+
                    "algorithm=MD5, qop=\"auth\"",
                    username = "foo", password = "bar",
                    cnonce = "082c875dcb2ca740",
                    nonce_count = 1,
```

```
                    expected_creds = "Digest username=\"foo\",
realm=\"DRealm1\", "+

"nonce=\"claGgoRXBAA=7583377687842fdb7b56ba0555d175baa0b800e3\", "+
                    "uri=\"/test/drealm1\", algorithm=MD5, "+
                    "response=\"bcfaa62f1186a31ff1b474a19a17cf57\", "+
                    "qop=auth, nc=00000001, cnonce=\"082c875dcb2ca740\""
                },
                new {
                    req_method = "GET",
                    req_path = "/test/drealm1/",
                    challenge = "Digest realm=\"DRealm1\", "+

"nonce=\"Ure30oRXBAA=7eca98bbf521ac6642820b11b86bd2d9ed7edc70\", "+
                    "algorithm=MD5, qop=\"auth\"",
                    username = "foo bar", password = "",
                    cnonce = "082c875dcb2ca740",
                    nonce_count = 1,
                    expected_creds = "Digest username=\"foo bar\",
realm=\"DRealm1\", "+

"nonce=\"Ure30oRXBAA=7eca98bbf521ac6642820b11b86bd2d9ed7edc70\", "+
                    "uri=\"/test/drealm1/\", algorithm=MD5, "+
                    "response=\"93c9c6d5930af3b0eb26c745e02b04a0\", "+
                    "qop=auth, nc=00000001, cnonce=\"082c875dcb2ca740\""
                },
                new {
                    req_method = "GET",
                    req_path = "/test/drealm1/",
                    challenge = "Digest realm=\"DRealm1\", "+

"nonce=\"7thGplhaBAA=41fb92453c49799cf353c8cd0aabee02d61a98a8\", "+
                    "algorithm=MD5, qop=\"auth\"",
                    username = "", password = "pass",
                    cnonce = "6509bc74daed8263",
                    nonce_count = 1,
                    expected_creds = "Digest username=\"\",
realm=\"DRealm1\", "+

"nonce=\"7thGplhaBAA=41fb92453c49799cf353c8cd0aabee02d61a98a8\", "+
                    "uri=\"/test/drealm1/\", algorithm=MD5, "+
                    "response=\"bc597110f41a62d07f8b70b6977fcb61\", "+
                    "qop=auth, nc=00000001, cnonce=\"6509bc74daed8263\""
                },
                new {
                    req_method = "GET",
                    req_path = "/test/drealm1/",
                    challenge = "Digest realm=\"DRealm1\", "+

"nonce=\"s3MzvFhaBAA=4c520af5acd9d8d7ae26947529d18c8eae1e98f4\", "+
                    "algorithm=MD5, qop=\"auth\"",
                    username = "", password = "",
                    cnonce = "1522e61005789929",
                    nonce_count = 1,
                    expected_creds = "Digest username=\"\",
realm=\"DRealm1\", "+

"nonce=\"s3MzvFhaBAA=4c520af5acd9d8d7ae26947529d18c8eae1e98f4\", "+
                    "uri=\"/test/drealm1/\", algorithm=MD5, "+
                    "response=\"22cfa2b30cb500a9591c6d55ec5590a8\", "+
                    "qop=auth, nc=00000001, cnonce=\"1522e61005789929\""
                },
                new {
```

```
                    req_method = "GET",
                    req_path = "/",
                    challenge = "Digest realm=\"Oblivion\", nonce=\"nonce-
value\"",
                    username = "FooBar", password = "pass",
                    cnonce = "",
                    nonce_count = 1,
                    expected_creds = "Digest username=\"FooBar\",
realm=\"Oblivion\", "+
                    "nonce=\"nonce-value\", uri=\"/\", "+
                    "response=\"f72ff54ebde2f928860f806ec04acd1b\""
                },
                new {
                    req_method = "GET",
                    req_path = "/",
                    challenge = "Digest realm=\"Baztastic\",
nonce=\"AAAAAAAA\", "+
                    "algorithm=\"md5-sess\", qop=auth",
                    username = "USER", password = "123",
                    cnonce = "15c07961ed8575c4",
                    nonce_count = 1,
                    expected_creds = "Digest username=\"USER\",
realm=\"Baztastic\", "+
                    "nonce=\"AAAAAAAA\", uri=\"/\", algorithm=MD5-sess, "+
                    "response=\"cbc1139821ee7192069580570c541a03\", "+
                    "qop=auth, nc=00000001, cnonce=\"15c07961ed8575c4\""
                }
            };
            GURL origin = new GURL("http://www.example.com");
            HttpAuthHandlerDigest.Factory factory = new
HttpAuthHandlerDigest.Factory();
            for (int i = 0; i < tests.Length; ++i)
            {
                HttpAuthHandler handler;
                int rv =
factory.CreateAuthHandlerFromString(tests[i].challenge,
                HttpAuth.Target.AUTH_SERVER,
                origin,
                new BoundNetLog(),
                out handler);
                Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
                Assert.IsTrue(handler != null);

                HttpAuthHandlerDigest digest =
(HttpAuthHandlerDigest)handler;
                PrivateObject PrivateDigest = new PrivateObject(digest);
                string creds =
(string)PrivateDigest.Invoke("AssembleCredentials", tests[i].req_method,
                    tests[i].req_path,
                    new AuthCredentials(tests[i].username,
tests[i].password),
                    tests[i].cnonce,
                    tests[i].nonce_count);
                Assert.AreEqual(tests[i].expected_creds, creds);
            }
        }
        string kSimpleChallenge = "Digest realm=\"Oblivion\", nonce=\"nonce-
value\"";
        bool RespondToChallenge(HttpAuth.Target target,
        string proxy_name,
        string request_url,
        string challenge,
        /*ref*/ Ref<String> token)
```

```
        {
            if (token == null)
            {
                Assert.Fail("|token| must be non-NULL");
                return false;
            }
            Assert.IsTrue(target != HttpAuth.Target.AUTH_PROXY || proxy_name
!= String.Empty);
            Assert.IsFalse(request_url == String.Empty);
            Assert.IsFalse(challenge == String.Empty);
            token.Value = String.Empty;
            HttpAuthHandlerDigest.Factory factory = new
HttpAuthHandlerDigest.Factory();
            HttpAuthHandlerDigest.NonceGenerator nonce_generator = new
HttpAuthHandlerDigest.FixedNonceGenerator("client_nonce");
            factory.set_nonce_generator(nonce_generator);
            HttpAuthHandler handler;
            GURL url_origin = new GURL(target == HttpAuth.Target.AUTH_SERVER
? request_url : proxy_name);
            int rv_create = factory.CreateAuthHandlerFromString(challenge,
target, url_origin.GetOrigin(), new BoundNetLog(), out handler);
            if (rv_create != (Int32)NetErrors.Error.OK || handler == null)
            {
                Assert.Fail("Unable to create auth handler.");
                return false;
            }
            TestCompletionCallback callback = new TestCompletionCallback();
            HttpRequestInfo request = new HttpRequestInfo();
            request.url = new GURL(request_url);
            AuthCredentials credentials = new AuthCredentials("foo", "bar");
            int rv_generate = handler.GenerateAuthToken(credentials, request,
callback.callback(), /*ref*/ token);
            if (rv_generate != (Int32)NetErrors.Error.OK)
            {
                Assert.Fail("Problems generating auth token");
                return false;
            }
            return true;
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_HandleAnotherChallenge()
        {
            HttpAuthHandlerDigest.Factory factory = new
HttpAuthHandlerDigest.Factory();
            HttpAuthHandler handler;
            string default_challenge = "Digest realm=\"Oblivion\",
nonce=\"nonce-value\"";
            GURL origin = new GURL("intranet.google.com");
            int rv = factory.CreateAuthHandlerFromString(default_challenge,
HttpAuth.Target.AUTH_SERVER, origin, new BoundNetLog(), out handler);
            Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
            Assert.IsTrue(handler != null);
            HttpAuthChallengeTokenizer tok_default = new
HttpAuthChallengeTokenizer(default_challenge/*default_challenge.begin(),
default_challenge.end()*/);

Assert.AreEqual(HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT,
handler.HandleAnotherChallenge(tok_default));
            string stale_challenge = default_challenge + ", stale=true";
            HttpAuthChallengeTokenizer tok_stale = new
HttpAuthChallengeTokenizer(stale_challenge/*stale_challenge.begin(),
stale_challenge.end()*/);
```

```
Assert.AreEqual(HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_STALE,
handler.HandleAnotherChallenge(tok_stale));
            string stale_false_challenge = default_challenge + ",
stale=false";
            HttpAuthChallengeTokenizer tok_stale_false = new
HttpAuthChallengeTokenizer(stale_false_challenge/*stale_false_challenge.begin
(), stale_false_challenge.end()*/);

Assert.AreEqual(HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_REJECT,
handler.HandleAnotherChallenge(tok_stale_false));
            string realm_change_challenge = "Digest realm=\"SomethingElse\",
nonce=\"nonce-value2\"";
            HttpAuthChallengeTokenizer tok_realm_change = new
HttpAuthChallengeTokenizer(realm_change_challenge/*realm_change_challenge.beg
in(), realm_change_challenge.end()*/);

Assert.AreEqual(HttpAuth.AuthorizationResult.AUTHORIZATION_RESULT_DIFFERENT_R
EALM, handler.HandleAnotherChallenge(tok_realm_change));
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToServerChallenge()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(HttpAuth.Target.AUTH_SERVER,
String.Empty, "http://www.example.com/path/to/resource", kSimpleChallenge,
/*ref*/ auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", "+
                "nonce=\"nonce-value\", uri=\"/path/to/resource\", "+
                "response=\"6779f90bd0d658f937c1af967614fe84\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToHttpsServerChallenge()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(HttpAuth.Target.AUTH_SERVER,
String.Empty, "https://www.example.com/path/to/resource", kSimpleChallenge,
/*ref*/ auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"/path/to/resource\", " +
                "response=\"6779f90bd0d658f937c1af967614fe84\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToProxyChallenge()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
                HttpAuth.Target.AUTH_SERVER,
                "http://proxy.intranet.corp.com:3128",
                "http://www.example.com/path/to/resource",
                kSimpleChallenge,
                /*ref*/ auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"/path/to/resource\", " +
                "response=\"6779f90bd0d658f937c1af967614fe84\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToProxyChallengeHttps()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
```

```
                HttpAuth.Target.AUTH_PROXY,
                "http://proxy.intranet.corp.com:3128",
                "https://www.example.com/path/to/resource",
                kSimpleChallenge,
                /*ref*/ auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"www.example.com:443\", " +
                "response=\"3270da8467afbe9ddf2334a48d46e9b9\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToProxyChallengeWs()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
                HttpAuth.Target.AUTH_PROXY,
                "http://proxy.intranet.corp.com:3128",
                "ws://www.example.com/echo",
                kSimpleChallenge,
                auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"www.example.com:80\", " +
                "response=\"aa1df184f68d5b6ab9d9aa4f88e41b4c\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToProxyChallengeWss()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
                HttpAuth.Target.AUTH_PROXY,
                "http://proxy.intranet.corp.com:3128",
                "wss://www.example.com/echo",
                kSimpleChallenge,
                auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"www.example.com:443\", " +
                "response=\"3270da8467afbe9ddf2334a48d46e9b9\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToChallengeAuthQop()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
                HttpAuth.Target.AUTH_SERVER,
                String.Empty,
                "http://www.example.com/path/to/resource",
                "Digest realm=\"Oblivion\", nonce=\"nonce-value\",
qop=\"auth\"",
                auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"/path/to/resource\", " +
                "response=\"5b1459beda5cee30d6ff9e970a69c0ea\", " +
                "qop=auth, nc=00000001, cnonce=\"client_nonce\"",
                auth_token.Value);
        }
        [TestMethod]
        public void HttpAuthHandlerDigestTest_RespondToChallengeOpaque()
        {
            Ref<String> auth_token = null;
            Assert.IsTrue(RespondToChallenge(
            HttpAuth.Target.AUTH_SERVER,
```

```
                String.Empty,
                "http://www.example.com/path/to/resource",
                "Digest realm=\"Oblivion\", nonce=\"nonce-value\", " +
                "qop=\"auth\", opaque=\"opaque text\"",
                auth_token));
            Assert.AreEqual("Digest username=\"foo\", realm=\"Oblivion\", " +
                "nonce=\"nonce-value\", uri=\"/path/to/resource\", " +
                "response=\"5b1459beda5cee30d6ff9e970a69c0ea\", " +
                "opaque=\"opaque text\", " +
                "qop=auth, nc=00000001, cnonce=\"client_nonce\"",
                auth_token.Value);
    }
}
```

Листинг Г.4 – HttpAuthHandlerFactoryTest.cs

```
using System;
using Blink.Platform;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Blink.External.Net;
using Blink.External.Base;
using Blink.URL;
namespace BlinkUnitTest.Blink.External.Net.Http
{
    class MockHttpAuthHandlerFactory : HttpAuthHandlerFactory
    {
        public MockHttpAuthHandlerFactory(int return_code)
        {
            return_code_ = return_code;
        }
        public override Int32 CreateAuthHandler(HttpAuthChallengeTokenizer
challenge,
            HttpAuth.Target target,
            GURL origin,
            CreateReason reason,
            int nonce_count,
            BoundNetLog net_log,
            out HttpAuthHandler handler)
        {
            handler = null;
            return return_code_;
        }
        int return_code_;
    }
    [TestClass]
    public class HttpAuthHandlerFactoryTest
    {
        [TestMethod]
        public void HttpAuthHandlerFactoryTest_RegistryFactory()
        {
            HttpAuthHandlerRegistryFactory registry_factory = new
HttpAuthHandlerRegistryFactory();
            GURL gurl = new GURL("www.google.com");
            const int kBasicReturnCode =
(Int32)NetErrors.Error.ERR_INVALID_SPDY_STREAM;
            MockHttpAuthHandlerFactory mock_factory_basic = new
MockHttpAuthHandlerFactory(kBasicReturnCode);

            const int kDigestReturnCode =
(Int32)NetErrors.Error.ERR_PAC_SCRIPT_FAILED;
            MockHttpAuthHandlerFactory mock_factory_digest = new
MockHttpAuthHandlerFactory(kDigestReturnCode);

            const int kDigestReturnCodeReplace =
(Int32)NetErrors.Error.ERR_SYN_REPLY_NOT_RECEIVED;
```

```
            MockHttpAuthHandlerFactory mock_factory_digest_replace =
                new MockHttpAuthHandlerFactory(kDigestReturnCodeReplace);
            HttpAuthHandler handler;
      Assert.AreEqual((Int32)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME,
                registry_factory.CreateAuthHandlerFromString(
                    "Basic", HttpAuth.Target.AUTH_SERVER, gurl, new
BoundNetLog(), out handler));
            registry_factory.RegisterSchemeFactory("Basic",
mock_factory_basic);
            Assert.AreEqual(kBasicReturnCode,
                registry_factory.CreateAuthHandlerFromString("Basic",
HttpAuth.Target.AUTH_SERVER, gurl,
                    new BoundNetLog(), out handler));
Assert.AreEqual((Int32)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME,
                registry_factory.CreateAuthHandlerFromString("Digest",
HttpAuth.Target.AUTH_SERVER, gurl,
                    new BoundNetLog(), out handler));
            registry_factory.RegisterSchemeFactory("Digest",
mock_factory_digest);
            Assert.AreEqual(kBasicReturnCode,
                registry_factory.CreateAuthHandlerFromString(
                    "Basic", HttpAuth.Target.AUTH_SERVER, gurl, new
BoundNetLog(), out handler));
            Assert.AreEqual(kDigestReturnCode,
                registry_factory.CreateAuthHandlerFromString("Digest",
HttpAuth.Target.AUTH_SERVER, gurl,
                    new BoundNetLog(), out handler));
            Assert.AreEqual(kBasicReturnCode,
                registry_factory.CreateAuthHandlerFromString(
                    "basic", HttpAuth.Target.AUTH_SERVER, gurl, new
BoundNetLog(), out handler));
            registry_factory.RegisterSchemeFactory("Digest",
mock_factory_digest_replace);
            Assert.AreEqual(kBasicReturnCode,
                registry_factory.CreateAuthHandlerFromString(
                    "Basic", HttpAuth.Target.AUTH_SERVER, gurl, new
BoundNetLog(), out handler));
            Assert.AreEqual(kDigestReturnCodeReplace,
                registry_factory.CreateAuthHandlerFromString("Digest",
HttpAuth.Target.AUTH_SERVER, gurl,
                    new BoundNetLog(), out handler));
        }
        [TestMethod]
        public void HttpAuthHandlerFactoryTest_DefaultFactory()
        {
            HostResolver host_resolver = new MockHostResolver();
            MockAllowURLSecurityManager url_security_manager = new
MockAllowURLSecurityManager();
            HttpAuthHandlerRegistryFactory http_auth_handler_factory =
                HttpAuthHandlerFactory.CreateDefault(host_resolver);
            http_auth_handler_factory.SetURLSecurityManager("negotiate",
url_security_manager);
            GURL server_origin = new GURL("http://www.example.com");
            GURL proxy_origin = new GURL("http://cache.example.com:3128");
            {
                HttpAuthHandler handler;
                int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                    "Basic realm=\"FooBar\"",
                    HttpAuth.Target.AUTH_SERVER,
                    server_origin,
                    new BoundNetLog(),
                    out handler);
```

```
                Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
                Assert.IsFalse(handler == null);
                Assert.AreEqual(HttpAuth.Scheme.AUTH_SCHEME_BASIC,
handler.auth_scheme());
                Assert.AreEqual("FooBar", handler.realm());
                Assert.AreEqual(HttpAuth.Target.AUTH_SERVER,
handler.target());
                Assert.IsFalse(handler.encrypts_identity());
                Assert.IsFalse(handler.is_connection_based());
            }
            {
                HttpAuthHandler handler;
                int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                    "UNSUPPORTED realm=\"FooBar\"",
                    HttpAuth.Target.AUTH_SERVER,
                    server_origin,
                    new BoundNetLog(),
                    out handler);

Assert.AreEqual((Int32)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME, rv);
                Assert.IsTrue(handler == null);
            }
            {
                HttpAuthHandler handler;
                int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                    "Digest realm=\"FooBar\", nonce=\"xyz\"",
                    HttpAuth.Target.AUTH_PROXY,
                    proxy_origin,
                    new BoundNetLog(),
                    out handler);
                Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
                Assert.IsFalse(handler == null);
                Assert.AreEqual(HttpAuth.Scheme.AUTH_SCHEME_DIGEST,
handler.auth_scheme());
                Assert.AreEqual("FooBar", handler.realm());
                Assert.AreEqual(HttpAuth.Target.AUTH_PROXY,
handler.target());
                Assert.IsTrue(handler.encrypts_identity());
                Assert.IsFalse(handler.is_connection_based());
            }
            {
                HttpAuthHandler handler;
                int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                    "NTLM",
                    HttpAuth.Target.AUTH_SERVER,
                    server_origin,
                    new BoundNetLog(),
                    out handler);
                Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
                Assert.IsFalse(handler == null);
                Assert.AreEqual(HttpAuth.Scheme.AUTH_SCHEME_NTLM,
handler.auth_scheme());
                Assert.AreEqual("", handler.realm());
                Assert.AreEqual(HttpAuth.Target.AUTH_SERVER,
handler.target());
                Assert.IsTrue(handler.encrypts_identity());
                Assert.IsTrue(handler.is_connection_based());
            }
            {
                HttpAuthHandler handler;
```

```
                int rv =
http_auth_handler_factory.CreateAuthHandlerFromString(
                    "Negotiate",
                    HttpAuth.Target.AUTH_SERVER,
                    server_origin,
                    new BoundNetLog(),
                    out handler);
#if USE_KERBEROS
                Assert.AreEqual((Int32)NetErrors.Error.OK, rv);
                Assert.IsFalse(handler == null);
                Assert.AreEqual(HttpAuth.Scheme.AUTH_SCHEME_NEGOTIATE,
handler.auth_scheme());
                Assert.AreEqual("", handler.realm());
                Assert.AreEqual(HttpAuth.Target.AUTH_SERVER,
handler.target());
                Assert.IsTrue(handler.encrypts_identity());
                Assert.IsTrue(handler.is_connection_based());
#else

Assert.AreEqual((Int32)NetErrors.Error.ERR_UNSUPPORTED_AUTH_SCHEME, rv);
                Assert.IsTrue(handler == null);
#endif
            }
        }
    }
}
```

Приложение Д

АКТ О ВНЕДРЕНИИ

(рекомендуемое)

АКТ

о внедрении результатов
выпускной квалификационной работы
Шириковой Вероники Михайловны

Программные средства аутентификации в браузере для операционной системы QPOS, разработанные инженером-программистом Шириковой В.М. – сотрудником ООО НТП «Криптософт», в ходе выполнения выпускной квалификационной работы, включено в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

Заместитель директора

«08» июня 2021 г. _____ А.А. Вартанов

М.П.