УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2015 Том 15 № 6 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2015 Vol. 15 No 6 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.056

РЕАЛИЗАЦИЯ КОНТРОЛЯ ЦЕЛОСТНОСТИ ОБРАЗА ОПЕРАЦИОННОЙ СИСТЕМЫ, ЗАГРУЖАЕМОГО ПО СЕТИ НА ТОНКИЙ КЛИЕНТ

Ю.А. Гатчин[a], О.А. Теплоухова[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: teplouhovaoa@gmail.com

Информация о статье
Поступила в редакцию 08.05.15, принята к печати 26.06.15
doi:10.17586/2226-1494-2015-15-6-1115-1121
Язык статьи – русский

Ссылка для цитирования: Гатчин Ю.А., Теплоухова О.А. Реализация контроля целостности образа операционной системы, загружаемого по сети на тонкий клиент // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15. № 6. С. 1115–1121.

Аннотация

Рассматривается проблема защиты процесса загрузки операционной системы с сервера на бездисковую рабочую станцию по сети, а также проводится анализ существующих способов контроля целостности передаваемой по сетевым протоколам информации. В рамках работы предлагается решение, позволяющее осуществлять контроль целостности загружаемого образа операционной системы до момента передачи на нее управления. Для обеспечения безопасности процесса загрузки выделяются ключевые информационные элементы, целостность которых необходимо гарантировать. Разрабатываемое решение как средство защиты информации должно удовлетворять требованиям информационной безопасности и при этом быть совместимо с остальными аппаратными и программными средствами, используемыми для защиты автоматизированных систем. Предлагаемое решение реализует алгоритм контроля целостности операционной системы, построенный с использованием инфраструктуры открытого ключа. В работе приводится анализ аппаратной конфигурации проектируемого решения с точки зрения удобства его использования и администрирования, а также оцениваются возможности осуществления атак со стороны злоумышленника на защищаемую информацию.

Ключевые слова

тонкий клиент, операционная система, загрузчик, модуль доверенной загрузки, контроль целостности, инфраструктура открытых ключей, электронная подпись.

Благодарности

Победитель программы «Участие молодежного научно-инновационного конкурса» («УМНИК»). Диплом «За лучший доклад» на IV Всероссийском конгрессе молодых ученых (2015 г.).

INTEGRITY MONITORING IMPLEMENTATION FOR THE OPERATING SYSTEM IMAGE LOADED THROUGH A NETWORK TO THE THIN CLIENT

Yu.A. Gatchin[a], O.A. Teploukhova[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: teplouhovaoa@gmail.com

Article info
Received 08.05.15, accepted 26.06.15
doi:10.17586/2226-1494-2015-15-6-1115-1121
Article in Russian

For citation: Gatchin Yu.A., Teploukhova O.A. Integrity monitoring implementation for the operating system image loaded through a network to the thin clients. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 6, pp. 1115–1121.

Abstract

The paper deals with the problem of protection for the process of operating system loading from the server to the diskless workstation through a network and the analysis of the existing ways of integrity monitoring for information transferred under network protocols. Within the scope of research, solution is proposed making it possible to perform integrity monitoring of the operating system loaded image before control is transferred to it. For security protection of loading, key information elements are marked which integrity needs to be guaranteed. The developed solution, as an information security product, should meet the requirements of information security and at the same time be compatible to other hardware and software tools used for protection of the automated systems. The proposed solution implements the algorithm of integrity monitoring for an operating system designed with the use of public key infrastructure. Analysis of hardware configuration for the projected solution from the point of view of its usability and administration ease is provided, and possibilities of intruder's attacks to the protected information are estimated, as well.

Keywords
thin client, operating system, boot loader, module for trusted boot loader, integrity monitoring, public key infrastructure, digital signature.

Acknowledgements
The winner of "Participation of Youth Research and Innovation Competition" ("UMNIK") program with diploma "For the Best Report" at the IV All-Russian Congress of Young Scientists (2015).

Введение

С каждым днем количество областей, для которых гарантия безопасности – это основное требование к автоматизированным системам обработки информации, неуклонно растет. В связи с этим внимание к вопросам построения защищенных информационных систем уже никого не удивляет. Необходимость наличия в данных системах так называемого ядра безопасности (Trusted Computing Base) – совокупности аппаратных, программных и специальных компонентов вычислительной системы, реализующих функции защиты безопасности, – осознают многие компании-интеграторы, занимающиеся построением этих систем [1].

Наряду с данным фактом необходимо учитывать смещение выбора аппаратной базы для построения вычислительных систем в пользу терминальных решений. Такая конфигурация системы подразумевает использование серверного компонента, на который устанавливается операционная система (ОС) и запускается терминальная служба. В свою очередь, на клиентских местах устанавливаются бездисковые рабочие станции – тонкие клиенты (ТК) [2], загрузка ОС на которые, как правило, осуществляется по сети.

Таким образом, информационная безопасность всех действий пользователя будет основываться на подлинности образа ОС, загружаемого на ТК с сервера. Уязвимости, которые могут при этом присутствовать в ОС, будь то ошибка в программном коде или сознательно помещенная программная закладка, позволяют злоумышленнику получить доступ к устройству и полностью взять его под контроль. Поэтому одна из приоритетных задач, которая должна быть решена на данном этапе, это контроль целостности загружаемого образа.

Постановка задачи

В настоящее время на рынке информационной безопасности в большинстве своем представлены средства защиты информации (СЗИ), контролирующие загрузку ОС с жесткого диска и предназначенные для применения на стационарных компьютерах. Такие средства не могут быть использованы на бездисковых ТК, поскольку реализованные в них механизмы контроля ОС, как правило, не рассчитаны на сетевую загрузку и не учитывают специфику распределенных терминальных систем.

В свою очередь, сам механизм защищенной загрузки ОС на протяжении последних лет все активнее привлекает внимание разработчиков встроенного программного обеспечения (ПО) в целях защиты от вредоносных программ типа «буткитов» и «руткитов». Целью таких программ является реализация раннего старта, позволяющего изменить программный код ОС и системных драйверов (например, установить перехваты) [3]. Одной из самых популярных технологий, реализовавших защищенную загрузку, является Secure Boot, включаемый, начиная с версии 2.2, в модуль UEFI (Unified Extensible Firmware Interface – унифицированный модульный интерфейс), пришедший на смену BIOS [4]. Как известно, компания Microsoft также включила в свою сертификационную программу для Windows 8 требования к производителям аппаратуры по реализации механизма защищенной загрузки UEFI [5]. Данный механизм должен обеспечивать безопасную предзагрузочную среду и осуществлять защиту от запуска неправомерного вредоносного кода из потенциально уязвимых точек, проверяя электронную подпись (ЭП) загружаемых модулей (драйверов, загрузчиков ОС и приложений).

Необходимо отметить, что в такой реализации режим Secure Boot достаточен для защищенной загрузки ОС только при условии, что злоумышленник не может осуществить физический доступ к защищаемому компьютеру, поскольку в большинстве случаев несанкционированного доступа (НСД) есть возможность осуществления подмены криптографических ключей. Кроме того, согласно спецификациям UEFI, в режиме Secure Boot не предусматривается механизмов создания доверительной сети для ключей. Таким образом, данный режим должен быть активирован непосредственно там, где будут использоваться компьютеры.

Важной особенностью решаемой в рамках данной работы задачи контроля целостности загружаемой ОС является тот факт, что сам образ хранится на сервере, и загрузка осуществляется по сети. Одним из самых распространенных стандартов для сетевой загрузки является PXE (Preboot Execution Environment) [6, 7]. Для реализации данного протокола на ТК должна быть установлена сетевая карта с поддержкой этой возможности. Для организации загрузки ОС в PXE используются протоколы IP, UDP, DHCP и TFTP. PXE-код, прописанный в сетевой карте, получает специальный загрузчик (Network Bootstrap Program) с TFTP-сервера в сети, после чего передает ему управление. Загрузчик имеет доступ к программным интерфейсам (API) PXE (Preboot, UDP, TFTP, UNDI) и, используя их, загружает с сервера ОС. Описываемый выше стандарт UEFI также содержит механизм сетевой загрузки по протоколу PXE. Но в слу-

чае обычной реализации протокола, когда клиент посылает PXE-запрос на загрузку, не предусмотрено способа гарантировать, что запрос обслуживается настоящим сервером. Соответственно, клиент не может достоверно знать, откуда он получает ОС и был ли образ изменен в процессе передачи. Например, в результате применения злоумышленником атаки «человек посередине» на TFTP-протокол на клиент вместе с файлами ОС может прийти вредоносный код. И хотя разработчики свободно распространяемого ПО предлагают различные механизмы, позволяющие нейтрализовать описанные угрозы при сетевой загрузке [8], готовых средств на рынке информационной безопасности представлено пока не было.

В настоящей работе предлагается решение, позволяющее осуществлять защищенную загрузку по сети на бездисковую рабочую станцию и предусматривающее способы противодействия перечисленным выше угрозам.

Предлагаемое решение

Разрабатываемое решение предполагает создание аппаратно-программного модуля доверенной сетевой загрузки (АПМДСЗ), предназначенного для работы именно на бездисковых ТК [9]. Такой модуль позволяет разделить встроенное ПО, реализующее механизм безопасной загрузки, и криптографические ключи. Кроме того, данный способ персонифицирует загрузку ОС под конкретных пользователей в соответствии с необходимыми им ПО и настройками.

Основные требования, предъявляемые к разрабатываемому решению [10], состоят в следующем:

1. функционирование на аппаратных ТК, не имеющих в своем составе жесткого диска;
2. реализация функции контроля целостности образа ОС, загружаемого с терминального сервера;
3. реализация механизма разграничения прав доступа к процессу загрузки ТК;
4. обеспечение целостности загрузчика ОС;
5. контроль аутентичности данных, поступающих на ТК;
6. независимость выбора загрузчика для ОС и, следовательно, возможность загрузки ОС семейств Unix/Windows;
7. блокировка несанкционированной загрузки ОС с внешних съемных носителей.

Для реализации разрабатываемого модуля доверенной загрузки используется сетевая карта с поддержкой протокола PXE, при этом алгоритм загрузки ОС закладывается в ее встроенном программном обеспечении. Функция блокировки загрузки ОС с неразрешенных носителей осуществляется путем запрета изменений конфигурации BIOS/UEFI без получения привилегий административного доступа.

Для обеспечения контроля целостности загружаемого образа выделяются два основных объекта, неизменность которых необходимо контролировать: загрузчик ОС размером 2–3 КБ, загружаемый в RAM-память на первоначальном этапе и осуществляющий дальнейший процесс загрузки, и сам образ ОС, передаваемый с сервера. В процессе загрузки выделяется базовая точка доверия [11], на которой будут основываться все остальные этапы. В качестве такой точки выбран защищенный внешний носитель (далее – токен), в памяти которого хранится загрузчик ОС. На таком внешнем носителе размещается собственный микроконтроллер – защищенный чип, имеющий специальную сертифицированную защиту как на аппаратном, так и на программном уровне, что позволяет успешно противостоять всем известным угрозам безопасности самого носителя, методам взлома и клонирования. Доступ к памяти этого устройства предоставляется только после успешной аутентификации пользователя – введения пароля. В результате хранения загрузчика на таком токене его целостность гарантирована. Данное решение не накладывает никаких ограничений на то, ОС какого семейства будет при этом загружаться на ТК.

Кроме того, использование отдельного защищенного носителя позволяет решить задачу разграничения доступа: загрузка ОС на ТК возможна только для пользователей, обладающих токеном и знающих пароль. Для обеспечения взаимной аутентификации токена и ТК предполагается реализация алгоритма, предусматривающего защиту от угроз подмены каждого из участвующих в процессе компонентов (пользователя, токена, ТК, сервера).

Учитывая вышесказанное, использование данного решения, в отличие от механизма Secure Boot, делает угрозу непосредственного доступа злоумышленника к ТК неактуальной, поскольку у него не будет возможности ни загрузиться с собственного носителя, ни заменить сетевую карту на свою (алгоритм аутентификации обнаружит подмену). Кроме того, разрабатываемый модуль – это достаточно гибкое решение, не накладывающее ограничения ни на используемое оборудование, ни на тип ОС. Администрирование системы с такими ТК упрощается за счет того, что администратору нужно работать только с защищенными носителями пользователей.

Таким образом, основной задачей в предлагаемом решении является разработка алгоритма загрузки, обеспечивающего контроль целостности и аутентичности передаваемого с сервера образа и защищенного выполнения каждого шага загрузки, исключающего реализацию угроз безопасности [12].

Анализ существующих способов контроля целостности информации

Под угрозой нарушения целостности понимается любое умышленное изменение информации, хранящейся в вычислительной системе или передаваемой из одной системы в другую [13]. Когда зло-

умышленники преднамеренно изменяют информацию, говорится, что целостность информации нарушена. Целостность также будет нарушена, если к несанкционированному изменению приводит случайная ошибка программного или аппаратного обеспечения.

К основным методам обеспечения целостности относятся:

1. метод контрольных сумм;
2. метод «циклического контрольного кода»;
3. однонаправленные функции хеширования[1];
4. электронная подпись.

Несмотря на простоту реализации и высокую производительность первых двух методов, необходимо отметить их основной недостаток: с точки зрения вычислительной сложности легко найти исходный прообраз для полученного значения функции, в результате чего равенство значений, получаемых на этапе проверки, не гарантирует неизменности передаваемой информации [14]. Данное свойство делает невозможным использование указанным методов в системах с высокими требованиями к защищенности.

В отличие от них, применение однонаправленной хэш-функции позволяет гарантировать стойкость используемого алгоритма к обратному преобразованию. Задача поиска коллизий (наличие нескольких прообразов функции, дающих на выходе одно хэш-значение) является крайне тяжелой по ресурсоемкости и производительности [15].

Но, наряду с данным преимуществом, у метода однонаправленной хэш-функции есть важная особенность: необходимость обеспечивать безопасное хранение или передачу контрольного значения проверяющей стороной. Относительно разрабатываемого решения очень важно, чтобы хэш-значение, вычисляемое для образа ОС в момент формирования на сервере, хранилось таким образом, чтобы исключить угрозу подмены. С точки зрения аппаратной конфигурации ТК единственным доверенным местом для хранения контрольного значения ввиду отсутствия жесткого диска является защищенная память токена [16].

Но такой вариант реализации метода контроля целостности вызывает ряд сложностей: образ ОС, загружаемый по сети на ТК, периодически может изменяться легитимным образом, например, при установке обновлений администратором. В этом случае контрольное значение в памяти токена должно быть перезаписано, поскольку изменится и хэш-значение. Но при достаточно развитой структуре организации, предполагающей наличие нескольких десятков ТК, использование подобного сценария крайне затрудняет работу администратора.

Инфраструктура открытого ключа

Альтернативный вариант контроля целостности передаваемого образа – использование инфраструктуры открытого ключа (ИОК) для вычисления электронной подписи.

ИОК представляет собой комплексную систему, сервисы которой реализуются и предоставляются с использованием технологии открытых ключей, т.е. криптосистему, в которых для прямого и обратного преобразования используются два разных ключа – открытый и закрытый [17].

Одним из основных компонентов ИОК является удостоверяющий центр (УЦ), обеспечивающий изготовление сертификатов открытых ключей и управление (аннулирование, приостановление, возобновление) ими. Сервер, выполняющий функции УЦ, должен располагаться строго внутри контролируемой зоны организации и не иметь внешних подключений либо полностью быть отключенным от сети. При этом обязательно должен быть предусмотрен ресурс, доступный всем пользователям системы, позволяющий проверять статус выданного сертификата (действителен или отозван) в режиме реального времени. Как правило, в качестве такого ресурса используется либо сервер, содержащий список отозванных сертификатов (Certificate Revocation List – CRL), периодически обновляемый вручную администратором, либо сервер, на котором работает служба онлайн-проверки статуса сертификата (Online Certificate Status Protocol – OCSP). Во втором случае информация о статусе сертификатов более актуальна, так как данная служба ее получает напрямую из реестра УЦ, и ответ такой службы фиксирован и сравнительно мал, в отличие от CRL, который может иметь большой объем. Но в этом случае сервер УЦ должен быть в сети и предоставлять возможность обращения службе OCSP к своей базе, что порождает возможность угрозы НСД и компрометации ключа УЦ, которым подписываются все выдаваемые сертификаты. В таком случае предполагается иерархическая архитектура удостоверяющих центров – корневой, не подключенный в сеть организации, и подчиненный, выдающий сертификаты пользователям и позволяющий подключение службы OCSP к своей базе. В разрабатываемом решении терминальный сервер обладает парой ключей: закрытым (ЗК) и открытым (ОК), при этом факт владения открытым ключом и факт того, что этот открытый ключ сопоставлен закрытому, подтверждается сертификатом, выдаваемым УЦ.

При передаче образа ОС сервер подписывает его своим ЗК. В используемом алгоритме электронной подписи в качестве одного из этапов также используется хэш-функция: на вход функции подписи

[1] ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования. Введ. 01.01.2013. М.: Стандартинформ, 2012. 33 с.

подается не сам образ, а его хэш-значение. Но принципиальное отличие данного метода от метода простого хэширования состоит в том, что хэш-значение, вычисленное для ОС, не хранится постоянно на токене, и его не надо периодически обновлять в случае изменения ОС. Оно вычисляется каждый раз на ТК от полученного образа и сравнивается с тем, что было получено в результате расшифрования электронной подписи с помощью ОК сервера (рис. 1).

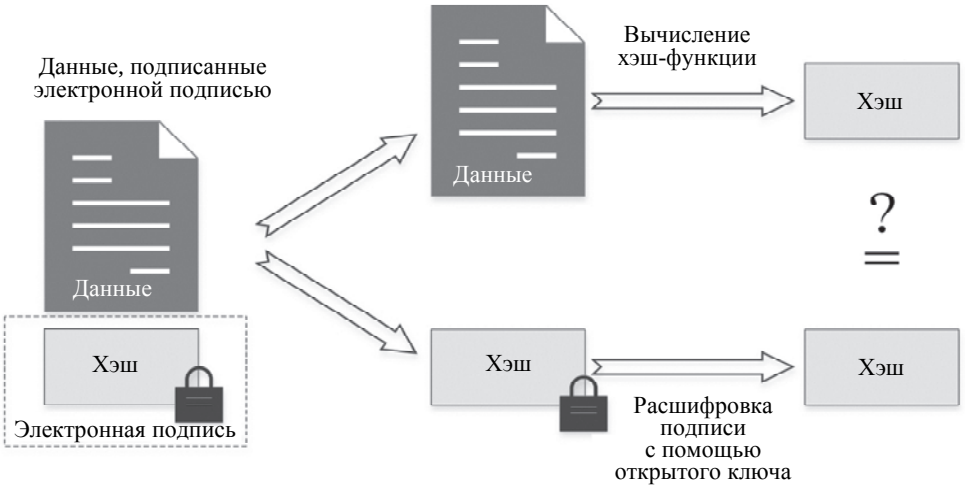

Рис. 1. Механизм проверки электронной подписи

Таким образом, если администратор внес изменения в загружаемый образ (установил ПО или обновления), это не повлияет на успешность проверки.

Реализация алгоритма контроля целостности ОС

Реализация описанного алгоритма контроля целостности передаваемого образа подразумевает формирование на стороне сервера электронной подписи образа и передача на ТК самого образа, подписи и сертификата открытого ключа для осуществления проверки.

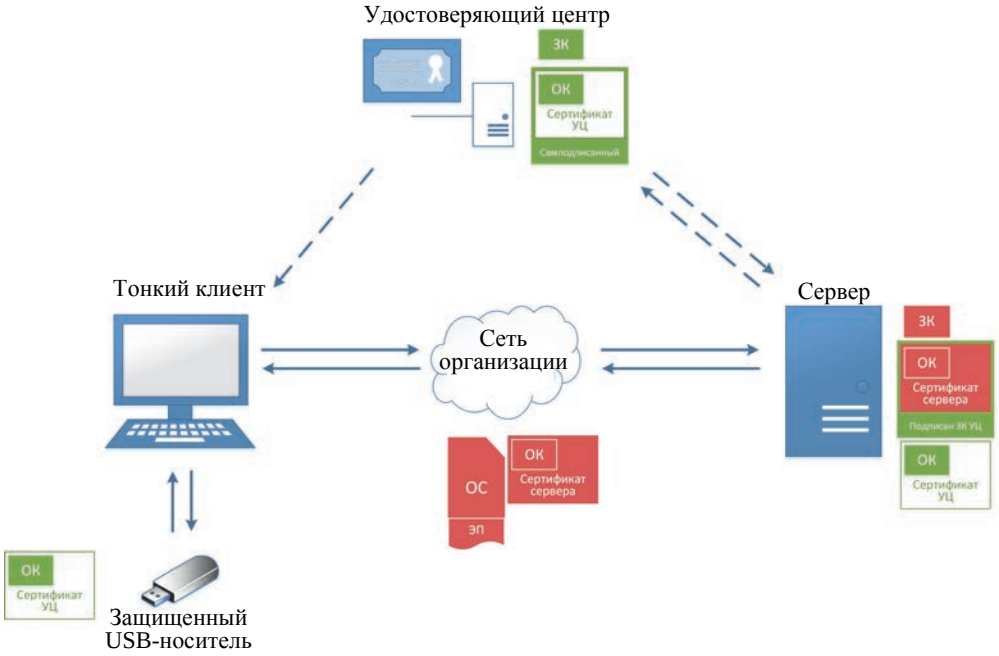

Рис. 2. Компоненты инфраструктуры открытого ключа для разрабатываемого алгоритма контроля целостности

Но в этом случае возникает угроза реализации атаки Man-in-the-middle: злоумышленник может перехватить передаваемый образ, подписать его своим ключом и передать ТК свой сертификат [18]. Для нейтрализации данной угрозы предлагается хранить на внешнем ключевом носителе сертификат открытого ключа УЦ (или цепочку сертификатов в случае иерархической структуры УЦ). При формировании сертификата открытого ключа серверу УЦ заверяет выдаваемый сертификат своим закрытым ключом, что позволяет в момент верификации электронной подписи переданного образа проверить также подлин-

ность сертификата сервера. Таким образом, компоненты, входящие в состав ИОК, необходимые для реализации предлагаемого решения, указаны на рис. 2.

Проверка подписи происходит в два этапа: вычисление хэш-функции образа и собственно математические вычисления, предусмотренные в данном алгоритме подписи, т.е. проверка того или иного соотношения, связывающего хэш-функцию образа, подпись под этим документом и открытый ключ подписывающей стороны. Если требуемое соотношение выполнено, то подпись признается правильной, а сам образ – подлинным, в противоположном случае образ считается измененным, а подпись под ним – недействительной.

Компоненты инфраструктуры открытого ключа, участвующие в протоколе:

- удостоверяющий центр CA;
- тонкий клиент TC;
- терминальный сервер TS;
- закрытый ключ сервера k_s;
- сертификат открытого ключа сервера k_o;
- сертификат открытого ключа удостоверяющего центра k_{CA};
- образ операционной системы OS;
- h – хэш-функция;
- f – функция вычисления электронной подписи;
- v – функция проверки электронной подписи.

Алгоритм загрузки и проверки целостности образа:

1. инициализация протокола загрузки образа OS с терминального сервера TS после того, как eToken подключили к тонкому клиенту TC;
2. формирование запроса к TS на получение образа OS и сертификата открытого ключа k_o сервера;
3. формирование на стороне сервера электронной подписи образа:
   $f(h(OS), k_s)$, где $h(OS)$ – хэш-функция от образа OS;
4. получение сертификата k_o сервера образа OS, подписанного закрытым ключом k_s сервера;
5. проверка целостности образа и подлинности подписи:
   $v(h(OS), k_o)$;
6. обращение к eToken за корневым сертификатом удостоверяющего центра k_{CA};
7. проверка подлинности сертификата открытого ключа сервера k_o:
   $v(h(k_o), k_{CA})$;
8. в случае успешности обеих проверок образ OS загружается в RAM-память, и управление передается на него.

Еще одним важным требованием к функционалу разрабатываемого решения является доверенная аутентификация всех участвующих компонентов: сервера, ТК, токена и пользователя. Для реализации такого механизма взаимной аутентификации перед непосредственно загрузкой ОС разрабатывается алгоритм, позволяющий выполнить следующие требования:

1. токен должен аутентифицировать ТК, т.е. ТК должен доказать токену, что входит в состав системы;
2. ТК должен аутентифицировать токен, т.е. токен должен доказать ТК, что является легитимным носителем и был выпущен администратором системы;
3. ТК должен аутентифицировать пользователя, т.е. пользователь должен доказать системе, что предъявленный токен принадлежит ему и он является легитимным пользователем системы.

При этом необходимо понимать, что процесс дальнейшей аутентификации пользователя в ОС, а также для выполнения специальных операций, предусмотренных в рамках процессов, требующих повышенного уровня защиты, является обязательным этапом в данном решении. Несмотря на то, что добавление требований к пользователю по хранению дополнительных секретов усложняет его работу, альтернативный вариант (передача образа ОС уже с осуществленной авторизацией) представляет существенную угрозу НСД к информации [19].

Заключение

В результате реализации описанного алгоритма в рамках разрабатываемого решения обеспечивается выполнение основных функциональных требований к аппаратно-программным модулям доверенной загрузки, за исключением блокировки несанкционированной загрузки операционной системы с внешних съемных носителей. Последнее достигается дополнительными внешними мерами. Предлагаемый модуль, таким образом, позволяет избежать организации шифрованных каналов связи (например, VPN-туннелей) в решении задачи доверенной загрузки ОС на бездисковый тонкий клиент, что является его серьезным экономическим преимуществом. Кроме того, предлагаемое решение может быть интересно организациями, превращающим старый парк компьютеров в бездисковые станции с сетевой загрузкой, так как весь модуль с аппаратной точки зрения представляет собой комплекс сетевой карты и защищенного носителя.

References

1. Novikov S.V., Zima V.M., Andrushkevich D.V. Approach to building securer distributed networks of data processing based on trusted infrastructure. *SPIIRAS Proceedings*, 2015, vol. 38, no. 1, pp. 34–51. (In Russian)
2. Shpunt Ya. Using thin clients. Benefits, costs, and pitfalls. *Intelligent Enterprise/RE*, 2011, no. 5(227), pp. 54–55. (In Russian)
3. Smith R. *Managing EFI Boot Loaders for Linux: Dealing with Secure Boot*. Available at: http://www.rodsbooks.com/efi-bootloaders/secureboot.html (accessed 26.05.2015).
4. Wilkins R., Richardson B. *UEFI Secure Boot in Modern Computer Security Solutions*. Available at: http://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf (accessed 26.05.2015).
5. Manan V., van der Hoeven A. *Windows 8.1 Secure Boot Key Creation and Management Guidance*. Available at: https://msdn.microsoft.com/en-gb/en-en/library/dn747883.aspx (accessed 26.05.2015)
6. *iPXE – Open Source Boot Firmware*. Available at: http://ipxe.org (accessed 26.05.2015).
7. Cherepenin S., Chubin I. *Zagruzka Bezdiskovykh Linux-Stantsii s Pomoshch'yu PXE*. Available at: http://www.opennet.ru/base/sys/pxe_diskless.txt.html (accessed 26.05.2015).
8. *Secure Boot-Compatible UEFI Netboot Over IPv4 and IPv6*. Available at: https://wiki.kubuntu.org/UEFI/SecureBoot-PXE-IPv6 (accessed 26.05.2015).
9. Gatchin Yu.A., Teploukhova O.A. Developing of controls of the operating system integrity with network load on the thin clients. *Trudy Kongressa po Intellektual'nym Sistemam i Iinformatsionnym Tekhnologiyam IS&IT'14* [Proc. Congress on Intelligent Systems and Information Technology IS&IT'14]. Moscow, 2014, vol. 3, pp. 243–248.
10. Gatchin Yu.A., Teploukhova O.A. Monitoring methods for the operating system integrity image at startup on a remote thin client terminal access systems. *Sbornik Tezisov Dokladov II Vserossiiskogo Kongressa Molodykh Uchenykh* [Proc. II All-Russian Congress of Young Scientists]. St. Petersburg, 2013, pp. 52–53. (In Russian)
11. Nesteruk F.G. To develop the technology of adaptive security systems based on intelligent agents. *Voprosy Zashchity Informatsii*, 2009, no. 1, pp. 50–56. (In Russian)
12. Gaidamakin N.A. Zone access control model in distributed computer systems. *Nauchno-Tekhnicheskaya Informatsiya. Seriya 2: Informatsionnye Protsessy i Sistemy*, 2002, no. 12, pp. 15–22
13. Alferov A.P., Zubov A.Yu., Kuz'min A.S., Cheremushkin A.V. *Osnovy Kriptografii* [Basics of Cryptography]. Moscow, Gelios ARV, 2005, 480 p.
14. Ivanov M.A. *Kriptograficheskie Metody zZashchity Informatsii v Komp'yuternykh Sistemakh i Setyakh* [Cryptographic Methods of Information Protection in Computer Systems and Networks]. Moscow, Kudits-Obraz, 2001, 363 p.
15. Panasenko S.P. *Obzor Atak na Algoritm Kheshirovaniya MD5: Poisk Kollizii. Chast' 1*. Available at: http://daily.sec.ru/2012/11/09/Obzor-atak-na-algoritm-heshirovaniya-MD5-poisk-kolliziy-CHast-1.html (accessed 18.02.2015).
16. Mukha M.D. Integrity and authenticity control system of operating system loaded through the network. *Sbornik Materialov XII Mezhdunarodnoi Konferentsii Kompleksnaya Zashchita Informatsii* [Proc. XII Int. Conf. Comprehensive Information Protection]. Yaroslavl', 2008, pp. 139–140. (In Russian)
17. Moldovyan N.A., Moldovyan A.A. *Vvedenie v Kriptosistemy s Otkrytym Klyuchom* [Introduction to Public Key Cryptosystems]. St. Petersburg, BKhV-Peterburg Publ, 2005, 288 p.
18. Cheremushkin A.V. *Kriptograficheskie Protokoly. Osnovnye Svoistva i Uyazvimosti* [Cryptographic Protocols. The Basic Properties and Vulnerability]. Moscow, Izdatel'skii Tsentr "Akademiya", 2009, 272 p.
19. Teploukhova O.A. Building a model of security threats operating system image loaded over the network to the thin client terminal access systems. *Sbornik Tezisov Dokladov III Vserossiiskogo Kongressa Molodykh Uchenykh* [Proc. III All-Russian Congress of Young Scientists]. St. Petersburg, 2014, no. 1, pp. 235–237. (In Russian)

Гатчин Юрий Арменакович – доктор технических наук, профессор, заведующий кафедрой, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, gatchin@mail.ifmo.ru

Теплоухова Ольга Александровна – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, teplouhovaoa@gmail.com

Yuri A. Gatchin – D.Sc., Professor, Head of Chair, ITMO University, Saint Petersburg, 197101, Russian Federation, gatchin@mail.ifmo.ru

Olga A. Teploukhova – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, teplouhovaoa@gmail.com