13. Carmichael R. D. *Theory of numbers and Diophantine Analysis*. New York, 1959. 118 p.

14. Chor B., Rivest R. A knapsack-Type public key cryptoystem based on arithmetic in finite fields of trades. *IEEE on information theory*, 1988, vol. IT-34, pp. 901–909.

15. Dickson L. E. *History of the Theory of Numbers*. New York, 1971, vol. 2: Diophantine Analysis.

16. Gloden A. *Mehgradige Gleichungen*. Groningen, 1944, p. 104.

17. Gurari E. M., Ibarra O. H. An NP-complete number theoretic problem. *Proc. 10th Ann. ACM. Symp. On Theory of computing*, New York, 1978, pp. 205–215.

18. Koblitz N. A Course in Number Theory and Cryptography. New York, Springer-Verlag Publ., 1987. 235 p.

19. Lenstra A. K., Lenstra H. W., Lovasz L. Factoring polynomials with rational coefficients. *Mathematische annalen*, 1982, vol. 261, pp. 515–534.

20. Lin C. H., Chang C. C., Lee R. C. T. A new public-key cipher system based upon the diophantine equations. *IEEE Transactions on Computers*, 1995, Jan., vol. 44, issue 1.

21. Merkle R., Hellman M. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 1978, vol. IT-24, pp. 525–530.

22. Mordell L. J. *Diophantine equations*. London, New York, Acad. Press, 1969, 312 p.

23. Osipyan V. O. Buiding of alphabetic data protection cryptosystems on the base of equal power knapsacks with Diophantine problems. *Sinconf 2012 – 5th International Conference on Security of Information and Networks*. Jaipur, ACM Publ., 2012, pp. 124–129.

24. Osipyan V. O. Mathematical modelling of cryptosystems based on Diophantine problem with gamma superposition method. *SIN'15 Proceedings of the 8th International Conference on Security of Information and Networks*. Sochi, ACM Publ., 2015, pp. 338–341.

25. Sierpinski W. *Elementary Theory of Numbers*. Warszawa, Hafner Publ. Company, 1964. 480 p.

26. Shannon C. Communication theory of secrecy systems. *Bell System Techn. J.*, 1949, vol. 28, no. 4, pp. 656–715.

УДК 004.056.53

МОДЕЛЬ И ПРОЕКТНЫЕ ДИАГРАММЫ СИСТЕМЫ ЦЕНТРАЛИЗОВАННОЙ ЗАЩИТЫ ПОДСЕТИ ХОСТОВ ПОД УПРАВЛЕНИЕМ UNIX-ПОДОБНЫХ ОПЕРАЦИОННЫХ СИСТЕМ

Статья поступила в редакцию 06.03.2019, в окончательном варианте – 05.04.2019.

Горкавенко Владимир Сергеевич, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, Татищева, 20а,

магистрант, e-mail: slipmetal@mail.ru

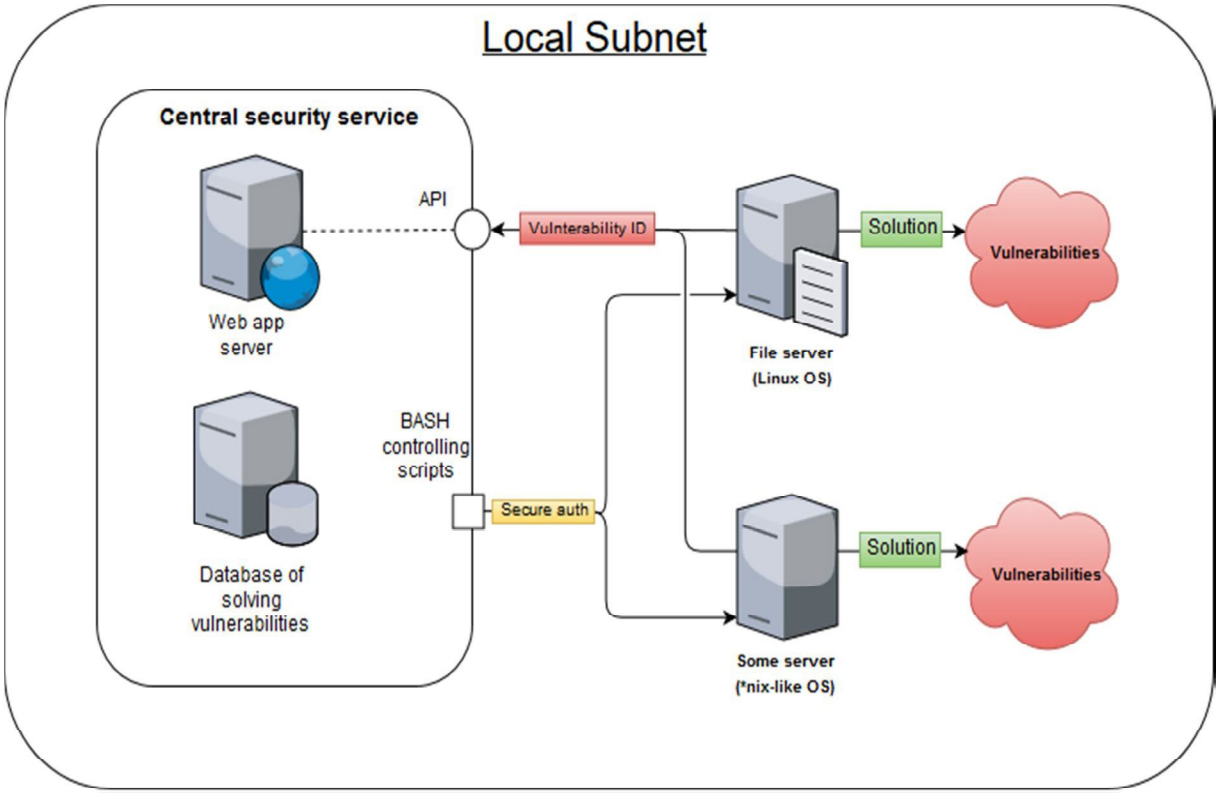
Ажмухамедов Искандар Маратович, Россия, г. Астрахань, Татищева, 20а, Астраханский государственный университет,

доктор технических наук, профессор, заведующий кафедрой информационной безопасности, e-mail: iskander_agm@mail.ru

Рассмотрено одно из возможных решений задачи защиты от несанкционированного доступа рабочих станций под управлением unix-подобных операционных систем, находящихся в одной подсети, посредством разработки централизованной системы устранения уязвимостей, позволяющей повысить уровень защищенности. Приведена формализация задачи. Предложено реализовать пять основных этапов, которые будет включать в себя алгоритм поиска и применения решения для устранения диагностированной уязвимости на рабочей станции. Согласно предложенному алгоритму, рабочая станция передает на сервер текст, содержащий информацию о диагностированной уязвимости, затем сервер, согласно информации, полученной от рабочей станции, инициирует поиск решения в базе знаний. Если решение найдено, то сервер передает его на рабочую станцию. Если решение не найдено, тогда централизованная система устранения уязвимостей уведомляет об этом лицо, принимающее решение. После передачи найденного решения для устранения диагностированной уязвимости, сервер инициирует его применение на рабочей станции. Во время применения решений для устранения диагностированных уязвимостей, рабочая станция передает на сервер информацию о процессе их применения. Если они по какой-либо причине не были реализованы, централизованная система устранения уязвимостей уведомляет лицо, принимающее решение о том, что уязвимости не были устранены. Предложено логическое представление централизованной системы устранения уязвимостей, а также диаграмма вариантов использования с описанием взаимодействия внутренних модулей и сервисов. Рассмотрено также одно из возможных решений задачи обеспечения защищенного соединения между рабочей станцией и сервером.

Ключевые слова: операционная система, несанкционированный доступ, информационная безопасность, централизованная защита, unix, linux, ubuntu

Графическая аннотация (Graphical annotation)



MODEL AND PROJECT DIAGRAMS OF THE SYSTEM OF CENTRALIZED PROTECTION OF A SUBNET OF HOSTS UNDER THE MANAGEMENT OF UNIX-LIKE OPERATIONAL SYSTEMS

The article was received by editorial board on 06.03.2019, in the final version – 05.04.2019.

Gorkavenko Vladimir S., Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation,
graduate student, e-mail: slipmetal@mail.ru
Azhmukhamedov Iskandar M., Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation,
Doct. Sci. (Engineering), Professor, Head of the Department of Information Security, e-mail: iskander_agm@mail.ru

One of the possible solutions to the task of protecting against unauthorized access of workstations running unix-like operating systems that are on the same subnet is considered through the development of a centralized vulnerability prevention system that allows to increase the level of security. The task formalization is given. It was proposed to implement five main stages, which will include an algorithm for finding and applying a solution to eliminate the diagnosed vulnerability at the workstation. According to the proposed algorithm, the workstation transmits to the server text containing information about the diagnosed vulnerability, then the server, according to information received from the workstation, initiates a search for a solution in the knowledge base. If a solution is found, the server sends it to the workstation. If no solution is found, then the centralized vulnerability prevention system notifies the decision maker. After the transfer of the found solution to eliminate the diagnosed vulnerability, the server initiates its application at the workstation. During the application of solutions to eliminate diagnosed vulnerabilities, the workstation transmits to the server information about the process of their application. If, for any reason, they were not implemented, the centralized vulnerability prevention system notifies the decision maker that the vulnerabilities have not been fixed. A logical representation of the central control system, as well as a diagram of use cases with a description of the interaction of internal modules and services, are proposed. One of the possible solutions to the problem of providing a secure connection between the workstation and the server is also considered.

Key words: operating system, unauthorized access, information security, centralized protection, unix, linux, ubuntu

Введение. Прогресс в области информационных технологий неизбежно сопровождается необходимостью обеспечения защищенности информации. При этом одной из основных проблем является защита информационных систем (ИС) от НСД [1–2, 11, 12].

В большинство операционных систем (ОС) уже встроены определенные подсистемы защиты от НСД. Например, механизм аутентификации пользователей при входе в операционные системы семейства Windows или Linux; механизмы защиты от исправлений ядра; способы ограничения режимов работы, предотвращающие выполнения нежелательных команд [10]. Также важными компонентами комплексной защиты операционных систем являются: расширенный аудит, контроль над учетными записями, защита пользователей и инфраструктуры от вторжений и вредоносного программного обеспечения [4].

Однако само по себе присутствие интегрированных средств обеспечения защищенности информации полностью не гарантирует ее безопасность. Должна быть осуществлена соответствующая настройка этих средств. Для этого, в свою очередь, от специалиста, который производит данную операцию, требуется высокий уровень профессионализма [3, 6, 8].

Далеко не все кампании могут позволить себе такого специалиста, так как высокий уровень его компетенций требует обеспечения соответствующего ему уровня заработной платы. Обращаясь к статистике использования десктопных и консольных ОС в мире, можно утверждать, что шанс найти и рекрутировать высококвалифицированного специалиста по устранению уязвимостей Unix-подобных ОС крайне мал [5], а известных общедоступных средств для автоматизации процесса централизованного обнаружения и устранения уязвимостей на хосте для таких ОС не существует [9]. При этом под термином «хост» будем понимать любой компьютер, подключенный к локальной сети (рис. 1).

Рисунок 1 – Обмен данными в централизованной системе устранения уязвимостей

Из вышесказанного можно сделать вывод, что разработка централизованной системы устранения уязвимостей (ЦСУУ) подсети хостов под управлением Unix-подобных ОС является весьма актуальной задачей.

В связи с этим ***целью данной работы*** стала разработка проекта такой системы.

Для достижения поставленной цели необходимо решить ***следующие задачи***:

- формализовать алгоритм решения задачи по устранению уязвимостей;
- разработать математическую модель наполнения базы знаний ЦСУУ множеством пар <уязвимость/способ ее устранения>;
- построить логическую структуру и проектные диаграммы разрабатываемого проекта;
- выбрать способ обеспечения защищенного соединения между сервером и хостами в одной подсети.

Формализация задачи. Пусть имеется: некоторое множество всех возможных уязвимостей – $\{U\}$; множество уязвимостей, которые можно диагностировать – $\{D\} \subseteq \{U\}$; множество уязвимостей, для которых есть решение для их устранения – $\{K\} \subseteq \{U\}$.

Тогда множество, состоящее из пар <диагностированная уязвимость / способ ее устранения> – $\{R\}$, составляющее базу знаний, можно представить в виде отношения:

$$R = D \subseteq U \cap K \subseteq U , \quad (1)$$

В связи с этим можно утверждать, что диагностированную и переданную на сервер уязвимость хоста ($u_i \in D$) можно устранить при условии, что $u_i \in \{K\}$.

Если $u_i \notin \{K\}$, то в этом случае уязвимость хоста не будет устранена, а система уведомит оператора о том, что на текущий момент в базе знаний ЦСУУ нет готового решения для устранения диагностированной уязвимости.

Представим описанные выше множества и их отношения в виде диаграммы Венна (рис. 2).

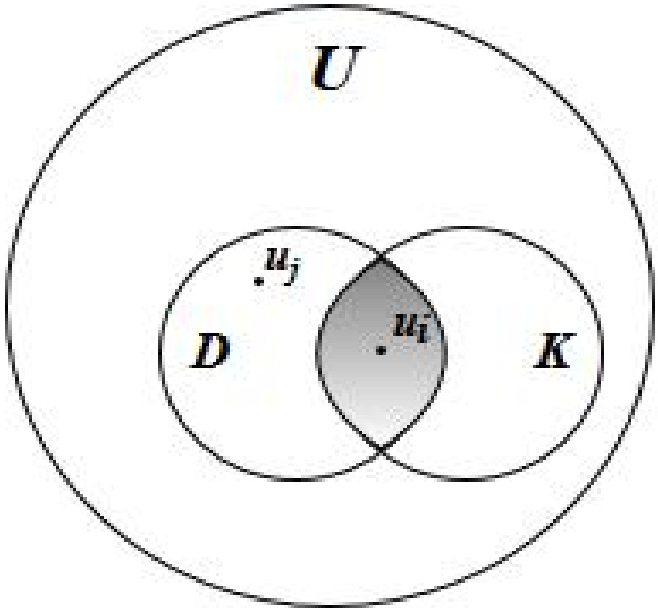


Рисунок 2 – Диаграмма Венна множеств и их отношений: $u_i \in \{D \cap K\}, u_j \notin \{K\}$

Перед построением проектных диаграмм, в которых будет отражено взаимодействие между хостом и сервером централизованной защиты, необходимо формализовать алгоритм решения задачи по устранению уязвимости.

В предлагаемом алгоритме можно выделить 5 основных этапов:

1) передача текста, содержащего информацию о диагностированной уязвимости хоста на сервер централизованной защиты;

2) поиск решения для устранения уязвимости;

3) если решение найдено, то передача информации о необходимых действиях по устранению обнаруженной уязвимости на хост;

4) применение найденного решения для устранения уязвимости на хосте;

5) получение информации от хоста о результатах устранения уязвимостей.

Рассмотрим этап 1 более подробно. Этот этап включает в себя функцию, выполняемую при сетевом взаимодействии во время передачи информации о диагностированной уязвимости хоста на сервер централизованной защиты (F_r). Эта функция может быть представлена в следующем виде:

$$F_r = R, F_{dc}\ V_j \quad , \tag{2}$$

где R – двумерный массив «решений», который можно представить как $[F_c\ V_i\ ; P_i]$; V_i – диагностированная уязвимость; P_i – решение для устранения уязвимости; F_c – функция кодирования информации об уязвимости; V_j – диагностированная и переданная на сервер уязвимость хоста; F_{dc} – функция декодирования закодированной информации об уязвимости хоста.

Для реализации этапа 2, при поиске решения для устранения уязвимости, предлагается использовать алгоритм нечеткого поиска [13] в базе знаний уязвимостей, так как возможна ситуация, когда в базе знаний хранится два разных решения для устранения двух разных уязвимостей, но текст, содержащий информацию об уязвимости, полученный при ее диагностировании может незначительно отличаться по причине присутствия в нем какой-либо переменной.

Если решение уязвимости не было найдено, тогда система должна уведомить лицо, принимающее решения (ЛПР), что диагностированная уязвимость не будет устранена на хосте, пока ее решение не будет добавлено в базу знаний ЦСУУ.

Если в результате выполнения этапа 2 было найдено решение для устранения уязвимости, с помощью защищенного протокола связи произойдет передача информации о решении для устранения уязвимости (этап 3). Далее при установлении защищенного удаленного доступа сервера к хосту произойдет выполнение списка команд, необходимых для устранения уязвимости (этап 4).

В процессе выполнения этапа 3 и 4 сервер будет получать обратную связь от хоста в виде информации об успешности или неуспешности выполнения команд для устранения уязвимостей. Полученная информация будет использована в том числе для уведомления ЛПР о том, что команды по какой-либо причине не были выполнены. Таким образом будет выполнен этап 5 предложенного выше алгоритма.

Формализация решения поставленной задачи позволяет перейти к построению проектных диаграмм, что, в свою очередь, позволит в дальнейшем реализовать ЦСУУ в виде программного продукта.

Проектные диаграммы. Для автоматизации, повышения удобства и надежности процесса защиты хостов от НСД необходимо разработать ПО, требующее минимального количества действий со стороны пользователя и автоматизирующее те действия, которые возможно выполнить без его участия (такие как написание файлов конфигурации, приведение финальных логов в удобный вид для прочтения и анализа и пр.).

Логическое представление проекта такого ПО приведено на рисунке 3.

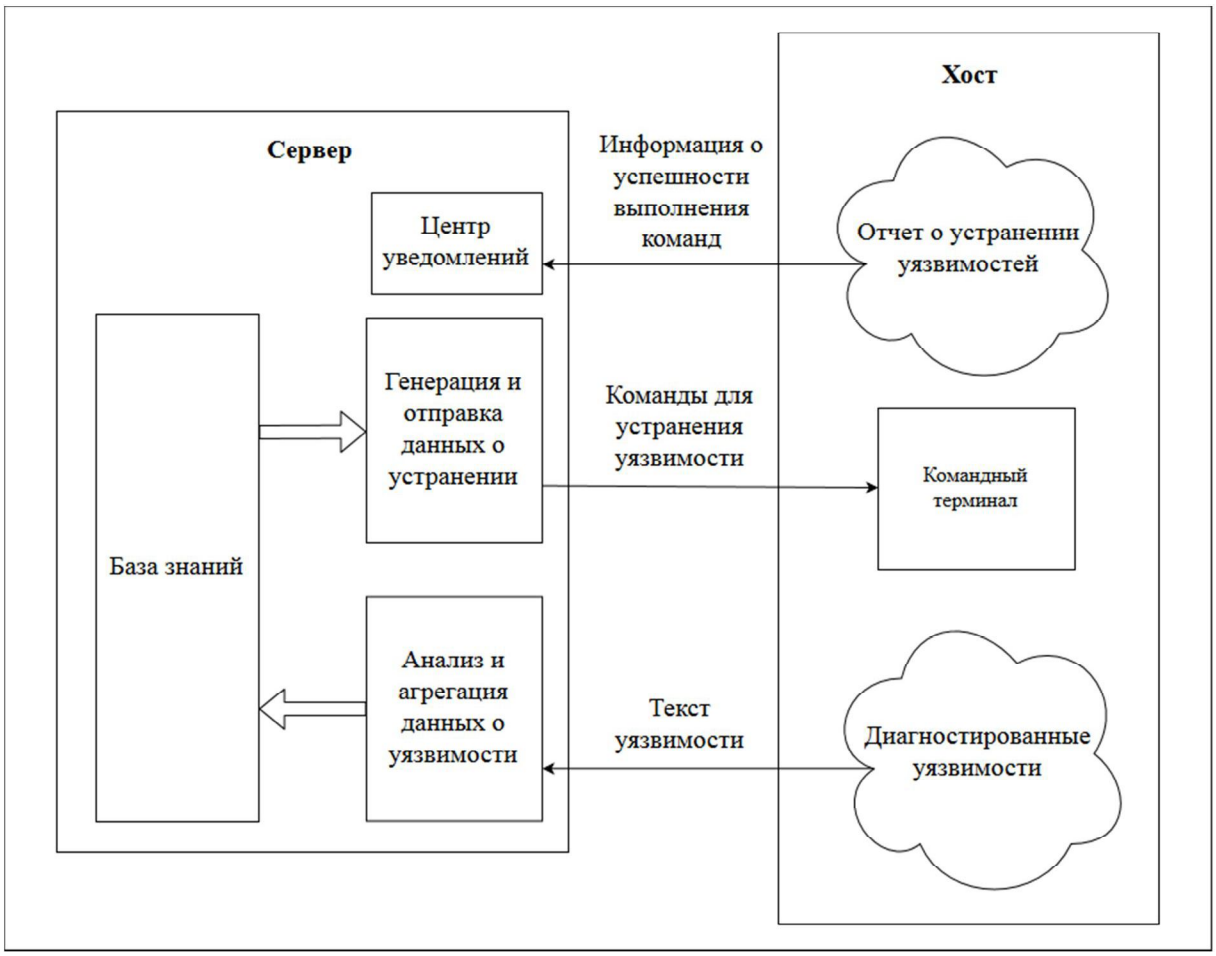


Рисунок 3 – Логическое представление предлагаемой ЦСУУ

В результате дальнейшего анализа были определены прецеденты, возникающие при работе ЦСУУ, что позволило, для описания пользовательских сценариев и вариантов использования, построить диаграмму прецедентов, которая является исходным концептуальным представлением (концептуальной моделью) системы в процессе ее проектирования и разработки (рис. 4).

При этом было выделено три актора: сервер приложения; специалист информационной безопасности; системный администратор. В совокупности своих функциональных возможностей они полностью решают поставленную задачу.

Основная роль системного администратора в проекте данной ЦСУУ – развертывание системы на сервере под управлением Unix-подобной операционной системы, а также организация процесса обслуживания оборудования и программной поддержки сервера приложения.

От специалиста информационной безопасности (ИБ) требуется: контролировать процесс допуска к подключению новых хостов в подсеть; организовать процесс составления протоколов запуска процесса анализа и устранения уязвимостей на хостах подсети, а также исполнять обязанности ЛПР.

Сервер приложения при этом обслуживает программную реализацию формализованного выше алгоритма, а также хранит информацию о хостах, допущенных к подключению в локальную сеть специалистом ИБ. ЦСУУ обязана уведомлять ЛПР о том, что решение по диагностированной уязвимости не

было найдено в базе знаний, а также если решение было найдено, но не было применено на хосте по какой-либо причине.

Далее необходимо произвести декомпозицию ЦСУУ на отдельные модули и более подробно описать работу основных функций и сервисов, а также их связей в предлагаемой системе. Рассмотрим основные блоки модулей и сервисов, необходимых для ее работы, более подробно.

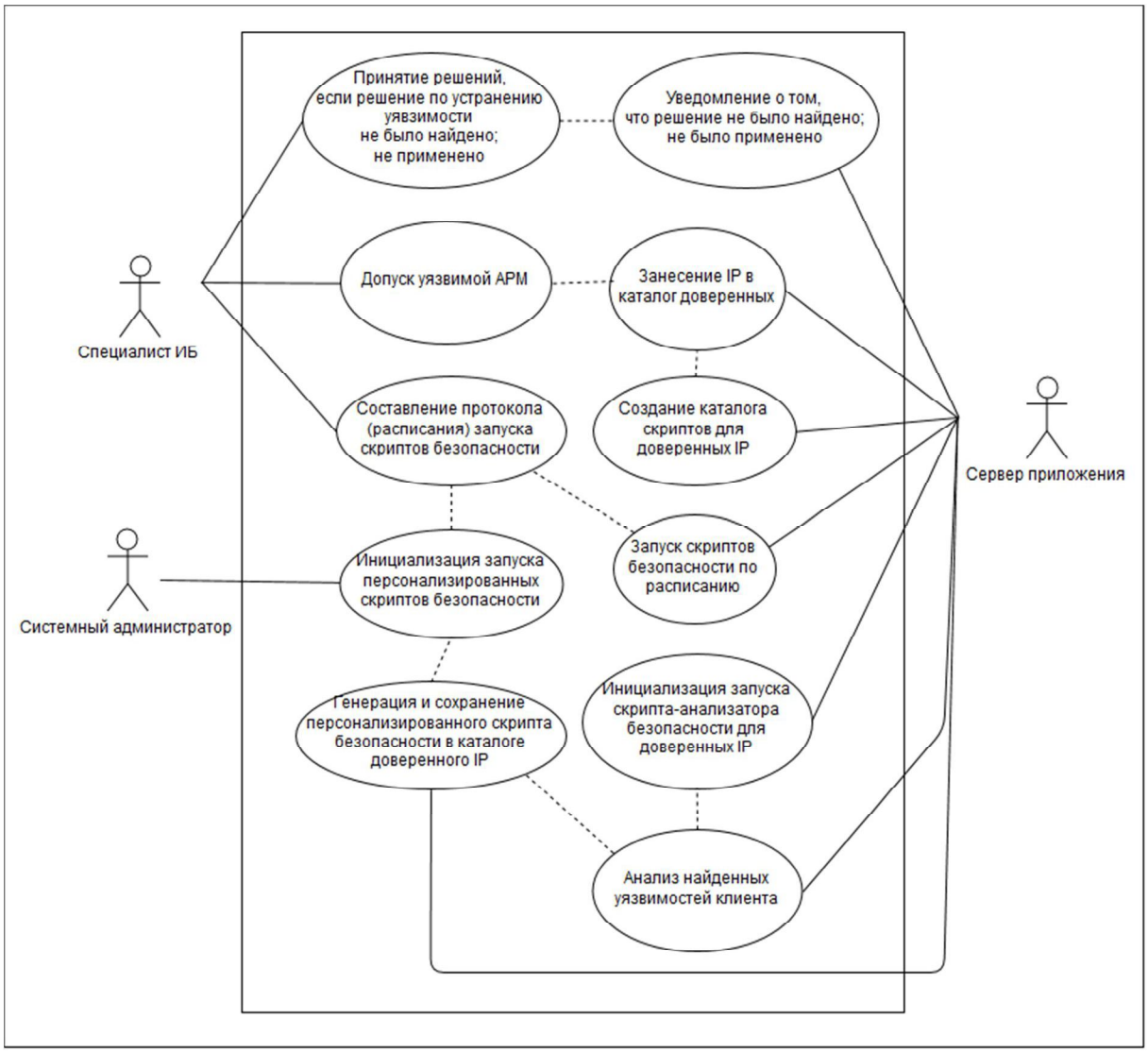


Рисунок 4 – Диаграмма прецедентов предлагаемой ЦСУУ

Описание модулей и сервисов. Одной из важных стадий построения ЦСУУ является описание структуры и схемы взаимодействия следующих модулей и сервисов:

- модуль анализа защищенности;
- модуль анализа и агрегации данных (декомпозиция результирующего лога работы на компоненты-уведомления о уязвимостях в анализируемой системе);
- сервис обслуживания запросов.

Для получения информации об уровне защищенности ОС, ее уязвимостях требуется реализовать модуль анализа защищенности (рис. 5), который содержит в себе имплементацию связи исполняемого клиентского скрипта с анализатором защищенности операционной системы.

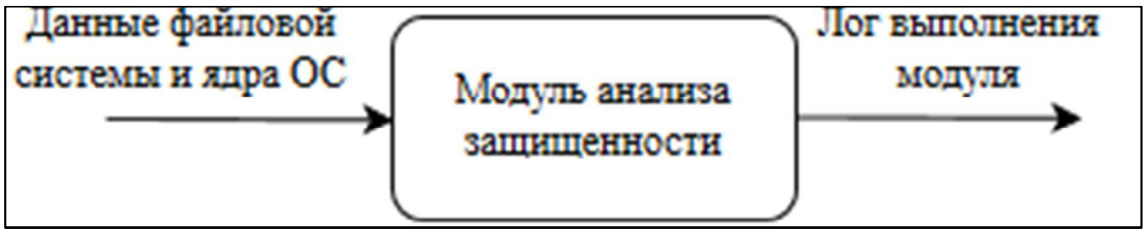


Рисунок 5 – Модуль анализа защищенности

Входными данными такого модуля является информация о файловой системе и ядре ОС для анализа, а результатом выполнения должна стать информация (лог), которая содержит в себе данные об уровне защищенности ОС и ее уязвимостях.

Для последующего анализа полученных данных о состоянии защищенности требуется реализовать модуль обработки результата (рис. 6) работы анализатора (сканера) защищенности. В процессе своей работы он кодирует и сохраняет результат анализа в файловой системе с целью оптимизации процесса ее отправки на сервер для дальнейшей обработки.

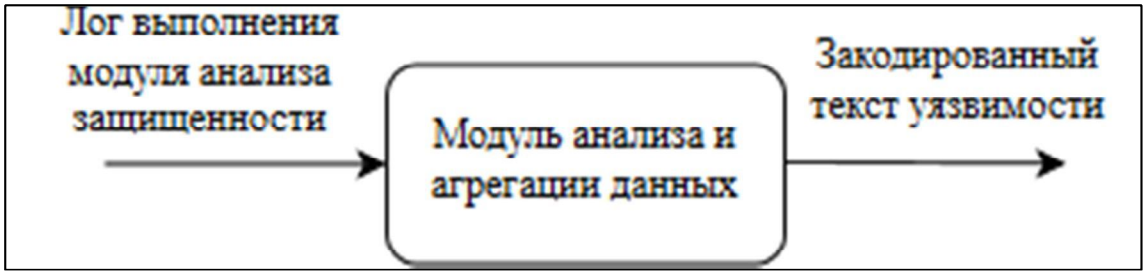


Рисунок 6 – Модуль анализа и агрегации данных

Для получения и обработки переданной с хоста информации об уязвимостях ОС необходимо создать сервис обслуживания запросов (рис. 7), который бы смог декодировать переданные на сервер данные и представить их в более удобном для ЦСУУ формате с целью дальнейшей обработки в программном коде.

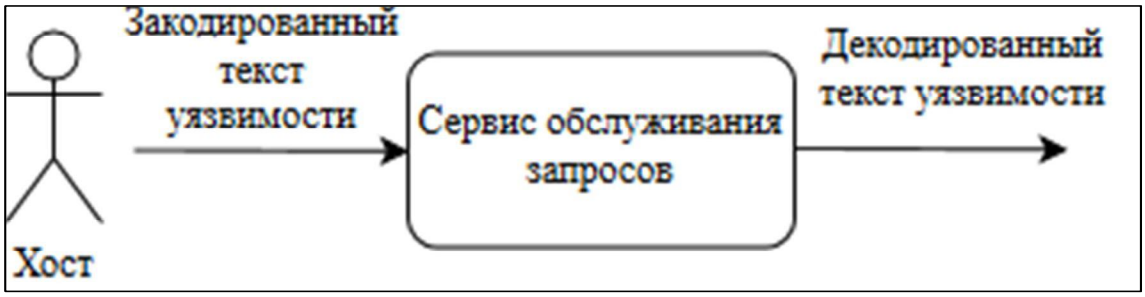


Рисунок 7 – Сервис обслуживания запросов

После получения информации о уязвимостях хоста-клиента серверу необходимо проанализировать полученные данные. Для этого требуется реализовать модуль анализа полученных данных (рис. 8).

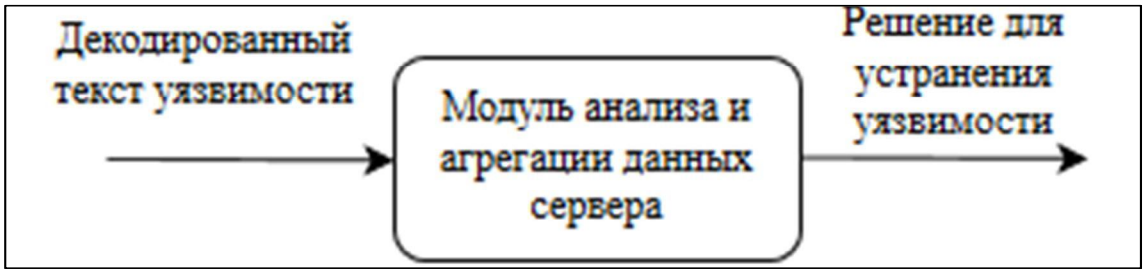


Рисунок 8 – Модуль анализа полученных данных

В результате выполнения модуля анализа полученных данных станет возможным получение хостом списка решений для устранения найденных на нем уязвимостей в оптимизированном формате для дальнейшего исполнения или дополнительной обработки. Если в результате работы модуля не было найдено решение, ЦСУУ уведомит ЛПР о том, что диагностированная уязвимость не будет устранена, так как информация о способе ее устранения отсутствует в базе знаний.

С целью сохранения совместимости с большинством unix-подобных ОС в процессе генерации скрипта, содержащего решения для устранения диагностированных уязвимостей, необходимо использовать один из скриптовых языков программирования, поддерживаемый различными ОС, например, BASH или Python, для последующей передачи исполняемого скрипта на хост (рис. 9). Если скрипт не был исполнен на хосте по какой-либо причине, ЦСУУ уведомит ЛПР об этом.

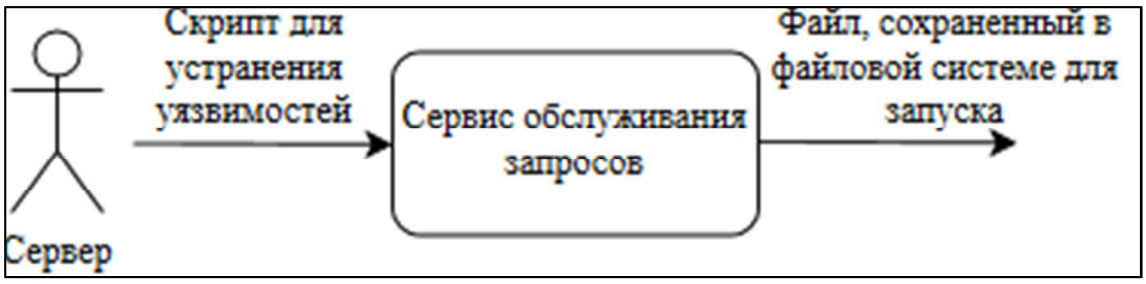


Рисунок 9 – Модуль получения ответа

При выполнении этапов 2, 3 и 4 алгоритма ЦСУУ важное значение имеет выбор способа обеспечения защищенного канала связи хостов в одной подсети для удаленного использования командного терминала при автоматизированной передаче данных в файловую систему сервера и получении информации о процессе исполнения скрипта, в котором содержатся решения для устранения диагностированных уязвимостей.

Способ обеспечения защищенного соединения. В качестве сервиса установки защищенного соединения между центральным хостом и хостами-клиентами в одной сети предлагается использовать протокол SSH.

Преимущества SSH протокола заключаются в возможности удаленного выполнения команд через командную оболочку (терминал). SSH позволяет безопасно передавать практически любой сетевой протокол, при этом шифрование осуществляется с помощью различных алгоритмов (к примеру, AES и RSA). Также предлагаемый протокол перед шифрованием и передачей информации сжимает ее. При этом SSH защищает передачу данных по каналу и предотвращает любую возможность внедрения в установленную сессию и перехвата данных [7].

Алгоритм протокола SSH можно разделить на три уровня, каждый из которых располагается над предыдущим: транспорт (открытие защищённого канала), аутентификация, подключение.

На транспортном уровне происходит сетевое подключение сервера к хосту на TCP-порт, указанный в опции Port в файле конфигурации хоста.

После установки TCP-соединения, хост и сервер обмениваются версиями SSH-протокола и другими вспомогательными данными, необходимыми для выяснения совместимости протоколов и для выбора алгоритмов работы.

На уровне аутентификации предлагается использовать аутентификацию с помощью ключей шифрования. Аутентификация посредством ключей происходит следующим образом:

- сервер отсылает хосту имя пользователя (username) и свой публичный ключ;
- хост проверяет в файле каталога сервиса SSH наличие присланного сервером открытого ключа. Если открытый ключ найден, то сервер генерирует случайное число и шифрует его открытым ключом клиента, после чего результат отправляется клиенту;
- сервер расшифровывает сообщение своим приватным ключом и отправляет результат серверу;
- хост проверяет полученный результат на совпадение с тем числом, которое он изначально зашифровал открытым ключом клиента, и в случае совпадения считает аутентификацию успешной.

После проведения всех вышеперечисленных процедур сервер получает возможность передавать команды хосту, а также управлять данными его файловой системы по защищенному каналу.

Таким образом, предложенный способ обеспечения защищенного соединения между сервером и хостом позволяет повысить уровень информационной безопасности при передаче данных, содержащих информацию о диагностированных уязвимостях и решениях для их устранения.

Выводы. В работе проведена формализация задачи защиты от НСД хостов под управлением Unix-подобных ОС, что позволило разработать логическую структуру, построить проектные диаграммы, описать взаимодействия основных модулей и сервисов предлагаемого решения.

Предложенная структура ЦСУУ позволяет проводить централизованный анализ защищенности хостов с целью дальнейшего устранения диагностированных уязвимостей. Также, согласно предложенному алгоритму, ЦСУУ позволяет уведомлять ЛПР в случае, если решение для устранения уязвимости не было найдено в базе знаний или оно по какой-либо причине не выполнено на хосте. Для обеспечения защищенного соединения между сервером и хостами предложено использовать протокол SSH.

После программной реализации ЦСУУ возможна коммерциализация программного продукта, так как предложенная концепция подходит для большинства организаций, в которых используются локальные сети с находящимися в ней хостами под управлением Unix-подобных ОС.

Дальнейшее развитие проекта может предусматривать реализацию алгоритма, выполняющего априорный и апостериорный анализ возможных последствий устранения той или иной уязвимости хоста, что позволит более эффективно выполнять поиск решений и применять их для устранения диагностированных уязвимостей [14]. Например, решение для устранения какой-либо уязвимости может закрыть один из сетевых портов, что может повлиять на работу электронной почты. В этом случае предлагаемый для дальнейшего развития проекта алгоритм поможет предусмотреть возможные последствия применения решений для устранения уязвимостей и учесть их на этапе формирования базы знаний.

Библиографический список

1. Алешников С. И. Проблемы информационной безопасности организации (предприятия) и пути их решения / С. И. Алешников, С. А. Дёмин, С. Б. Федоров, А. С. Федоров // Вестник Балтийского федерального университета им. И. Канта. Сер. Физико-математические и технические науки. – 2013. – Вып. 10. – С. 147–150.

2. Баранов А. П. Актуальные проблемы в сфере обеспечения информационной безопасности программного обеспечения / А. П. Баранов. – Режим доступа: http://emag.iis.ru/arc/infosoc/emag.nsf/BPA /12ccdaa5fd89de1fc32575bd003e2eb1, свободный. – Заглавие с экрана. – Яз. рус.

3. Дагаев А. Ф. Обеспечение информационной безопасности в вычислительной сети предприятия / А. Ф. Дагаев, А. Н. Самойлов, Е. А. Борисова // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. – 2008. – № 38 (4). – С. 1–4. – Режим доступа: https://cyberleninka.ru/article/v/obespechenie-informatsionnoy-bezopasnosti-v-vychislitelnoy-seti-predpriyatiya, свободный. – Заглавие с экрана. – Яз. рус.

4. Девятин П. Н. О проблеме представления формальной модели политики безопасности операционных систем / П. Н. Девятин // Труды Института системного программирования РАН. – 2017. – Т. 29, вып. 3. – С. 7–14.

5. Дорофеев А. В. Менеджмент информационной безопасности: основные концепции / А. В. Дорофеев, А. С. Марков // Вопросы кибербезопасности. – 2014. – № 1 (2). – С. 67–72.

6. Дунин В. С. Модель угроз информационной безопасности комплексной автоматизированной интеллектуальной системы «Безопасный город» / В. С. Дунин, Н. С. Хохлов // Вестник Воронежского института МВД России. – 2011. – № 4. – С. 1–5.

7. Коджешау М. А. Технологии и алгоритмы информационной безопасности / М. А. Коджешау // Вестник Адыгейского государственного университета. Сер. 4. Естественно-математические и технические науки. – 2017. – Вып. 2 (201). – С. 129–132.

8. Медведев Н. В. Модели управления доступом в распределенных информационных системах / Н. В. Медведев, Г. А. Гришин // Машиностроение и компьютерные технологии. – 2011. – № 1. – С. 1–19.

9. Оладько А. Ю. Подсистема мониторинга и аудита информационной безопасности в операционной системе Linux / А. Ю. Оладько // Известия Южного федерального университета. Технические науки. – 2012. – № 12. – С. 22–28.

10. Рукасуева С. Ю. Windows и альтернативные ей операционные системы / С. Ю. Рукасуева, А. П. Багаева // Актуальные проблемы авиации и космонавтики. – 2011. – Т. 1, вып. 7. – С. 459–460.

11. Трубачев Е. С. Проблемы информационной безопасности. Методы и средства защиты информационных ресурсов / Е. С. Трубачев // Вестник Волжского университета им. В.Н. Татищева. – 2009. – № 14. – С. 1–7.

12. Шубин А. Н. Оценка свойств информационных систем в стандартах по информационной безопасности / А. Н. Шубин // Известия Тульского государственного университета. Технические науки. – 2013. – Вып. 3. – С. 336–345.

13. George K. Thiruvathukal. What's in an Algorithm? / George K. Thiruvathukal // Computing in Science & Engineering. – 2013. – Vol. 15. – С. 15–27.

14. Arne Johanson. Software Engineering for Computational Science: Past, Present, Future / Arne Johanson, Wilhelm Hasselbring // Computing in Science & Engineering. – 2018. – Vol. 20. – С. 90–112.

References

1. Aleshnikov S. I., Demin S. A., Fedorov S. B., Fedorov A. S. Problemy informatsionnoy bezopasnosti organizatsii (predpriyatiya) i puti ikh resheniya [Problems of information security of an organization (enterprise) and ways to solve them]. *Vestnik Baltiyskogo federalnogo universiteta im. I. Kanta. Seriya "Fiziko-matematicheskie i tekhnicheskie nauki"* [Bulletin of the Baltic Federal University named after I. Kant. Series "Physico-mathematical and technical sciences"], 2013, no. 10, pp. 147–150.

2. Baranov A. P. *Aktualnye problemy v sfere obespecheniya informatsionnoy bezopasnosti programmnogo obespecheniya* [Actual problems in the field of ensuring information security of software]. Available at: http://emag.iis.ru/arc/infosoc/emag.nsf/BPA/12ccdaa5fd89de1fc32575bd003e2eb1.

3. Dagaev A. F., Samoylov A. N., Borisova Ye. A. *Obespechenie informatsionnoy bezopasnosti v vychislitelnoy seti predpriyatiya* [Ensuring information security in the enterprise's computer network]. Available at: https://cyberleninka.ru/article/v/obespechenie-informatsionnoy-bezopasnosti-v-vychislitelnoy-seti-predpriyatiya.

4. Devyatin P. N. O probleme predstavleniya formalnoy modeli politiki bezopasnosti operatsionnykh sistem [On the problem of presenting a formal model of the security policy of operating systems]. *Trudy Instituta sistemnogo programmirovaniya RAN* [Works of the Institute for System Programming of the RAS], 2017, vol. 29, no. 3, pp. 7–14.

5. Dorofeev A. V., Markov A. S. Menedzhment informatsionnoy bezopasnosti: osnovnye kontseptsii [Management of Information Security: Basic Concepts]. *Voprosy kiberbezopasnosti* [Cybersecurity Issues], 2014, no. 1 (2), pp. 67–72.

6. Dunin V. S., Khokhlov N. S. Model ugroz informatsionnoy bezopasnosti kompleksnoy avtomatizirovannoy intellektualnoy sistemy "Bezopasnyy gorod" [The Model of Information Security Threats of the Integrated Automated Intelligent System "Safe City"]. *Vestnik Voronezhskogo instituta MVD Rossii* [Bulletin of Voronezh Institute of the Ministry of Internal Affairs of Russia], 2011, no. 4, pp. 1–5.

7. Kodzheshau M. A. Tekhnologii i algoritmy informatsionnoy bezopasnosti [Technologies and algorithms for information security]. *Vestnik Adygeyskogo gosudarstvennogo universiteta. Seriya 4 "Yestestvenno-matematicheskie i tekhnicheskie nauki"* [Bulletin of Adyghe State University. Series 4 "Natural-mathematical and technical sciences"], 2017, no. 2 (201), pp. 129–132.

8. Medvedev N. V., Grishin G. A. Modeli upravleniya dostupom v raspredelennykh informatsionnykh sistemakh [Access control models in distributed information systems]. *Mashinostroenie i kompyuternye tekhnologii* [Mechanical Engineering and Computer Technologies], 2011, no. 1, pp. 1–19.

9. Oladko A. Yu. Podsistema monitoringa i audita informatsionnoy bezopasnosti v operatsionnoy sisteme Linux [Subsystem of monitoring and auditing information security in the Linux operating system]. *Izvestiya Yuzhnogo federalnogo universiteta. Tekhnicheskie nauki* [News of the Southern Federal University. Technical science], 2012, no. 12, pp. 22–28.

10. Rukasueva S. Yu., Bagaeva A. P. Windows i alternativnye ey operatsionnye sistemy [Windows and operating systems alternative to it]. *Aktualnye problemy aviatsii i kosmonavtiki* [Actual problems of aviation and cosmonautics], 2011, vol. 1, no. 7, pp. 459–460.

11. Trubachev Ye. S. Problemy informatsionnoy bezopasnosti. Metody i sredstva zashchity informatsionnykh resursov [Problems of information security. Methods and means of protection of information resources]. *Vestnik Volzhskogo universiteta imeni V.N. Tatishcheva* [Bulletin of Volzhsky University named after V.N. Tatishchev], 2009, no. 14, pp. 1–7.

12. Shubin A. N. Otsenka svoystv informatsionnykh sistem v standartakh po informatsionnoy bezopasnosti [Estimation of the properties of information systems in the standards of information security]. *Izvestiya Tulskogo gosudarstvennogo universiteta. Tekhnicheskie nauki* [News of Tula State University. Technical Science], 2013, vol. 3, pp. 336–345.

13. George K. Thiruvathukal. What's in an Algorithm? *Computing in Science & Engineering*, 2013, vol. 15, pp. 15–27.

14. Arne Johanson, Wilhelm Hasselbring. Software Engineering for Computational Science: Past, Present, Future. *Computing in Science & Engineering*, 2018, vol. 20, pp. 90–112.

УДК 004.032.26

ИДЕНТИФИКАЦИЯ DDOS-АТАК НА WEB-СЕРВЕРЫ

Статья поступила в редакцию 01.03.2019, в окончательном варианте – 08.04.2019.

Власенко Александра Владимировна, Кубанский государственный технологический университет, 350072, Российская Федерация, г. Краснодар, ул. Московская, 2,

кандидат технических наук, заведующая кафедрой компьютерных технологий и информационной безопасности Института информационных технологий и безопасности, e-mail: Vlasenko@kubstu.ru

Дзьобан Павел Игоревич, Кубанский государственный технологический университет, 350072, Российская Федерация, г. Краснодар, ул. Московская, 2.

кандидат технических наук, старший преподаватель, Краснодар, e-mail: antiemoboy@mail.ru

В настоящее время жизнь обычного человека связана с активным использованием всевозможных онлайн-сервисов. Они помогают не только повысить скорость получения услуг, но и улучшить качество жизни. За комфорт принято платить, поэтому актуальность вопросов обеспечения безопасности оказания онлайн-услуг не вызывает сомнений. Невозможность использования сервисов может приводить к моральному ущербу и в некоторых случаях к существенным материальным потерям. Одним из возможных нарушений нормальной работы web-сервера могут быть атаки типа DOS/DDOS. В данной статье предлагается механизм обнаружения DDOS-атак. Для уменьшения трудоемкости обнаружения вторжений предлагается производить автоматизированный контроль состояния защищенности информационно-телекоммуникационных ресурсов, выполняя на постоянной основе комплекс следующих мероприятий: а) анализ log-файлов web-сервера (в работе рассмотрен пример сервера Apache); б) выявление различных параметров из необработанных запросов (используется для распознавания входящего запроса на web-сервер как «разрешенного» и «вредоносного»); в) проверка каждого входящего запроса (его параметров) к web-серверу по коррелированности с идентифицированными параметрами из log-файлов. Этот этап и приводит к выявлению вредоносного запроса к web-серверу, который делает возможной потенциальную DDOS-атаку.

Ключевые слова: протокол, отказ в обслуживании (DOS), распределенный отказ в обслуживании (DDOS), IP-спуфинг, log-файлы, flood, web-серверы, информационная безопасность