DOI 10.36622/VSTU.2023.26.2.012
УДК 004.056.47

СОЗДАНИЕ КИБЕРПОЛИГОНА: ФОРМИРОВАНИЕ БЛОКА ЭМУЛЯЦИИ И СКАНИРОВАНИЯ ИНФРАСТРУКТУРЫ

С.С. Куликов, А.И. Саушкин

В работе предлагается вариант реализации программно-технического комплекса (ПТК), являющегося блоком анализа безопасности киберполигона. Блок позволяет эмулировать инфраструктуру заказчика и проводить ее сканирование на предмет наличия уязвимостей. Компонент эмуляции инфраструктуры обеспечивает ее быструю инициализацию и, при необходимости, оперативное восстановление. За счет использования технологий, обеспечивающих быстрое развертывание, инфраструктура может быть поднята на различных вычислительных ресурсах. Компонент анализа защищенности позволяет пользователю провести сканирование инфраструктуры на наличие уязвимостей и получить подробную информацию о них. Чтобы обеспечить прозрачную для пользователя интеграцию компонентов блока и простоту его использования, реализовано специальное приложение, выполняющее роль пользовательского интерфейса. ПТК позволяет реализовать различные способы размещения его компонентов даже на менее ресурсообеспеченных платформах за счет использования технологий контейнеризации.

Ключевые слова: киберполигон, информационная безопасность, эмуляция, технология быстрого развертывания.

Введение

Согласно исследованию кибер-атак, зарегистрированных в 2022-м году [1], количество инцидентов кибербезопасности в 2022 выросло на 20,8%. При этом наибольшему воздействию подверглись государственные учреждения, количество атак на сайты которых возросло более чем в 2 раза. Также большому количеству атак подверглась IT-сфера, что привело к множеству межотраслевых последствий как в связи с эффектом домино, затронувшим инфраструктуру клиентов и подрядчиков, так и в связи с нарушением нормального течения установленных бизнес-процессов клиентов из-за прерывания работы сервисов. Не обошлось и без ущерба конечным пользователям: в большом количестве атак на веб-ресурсы целью атакующих было не нарушение непрерывности бизнес-процессов путем приведения ресурса в недоступное состояние, а кража содержимого баз данных, представленного информацией о пользователях, с целью дальнейшей продажи или использования в других атаках, направленных непосредственно на пользователей.

Существенный всплеск количества атак на государственные ресурсы обуславливается политической направленностью. Атаки проводятся силами информационных операций иностранных государств, хакерских группировок и отдельных хактивистов.

Текущая ситуация оправдывает большой спрос на киберполигоны как в коммерческой, так и в государственной сфере, что подкреплено готовностью компаний тратить до 25% годового бюджета на создание и эксплуатацию киберполигонов [2].

Многообразие киберполигонов

В первую очередь, киберполигон является ПТК для обучения противодействия кибер-атакам и может включать множество блоков с различным функционалом. Однако использование киберполигона только для одной цели не является рентабельным. Площадки киберполигонов используются для проведения научно-исследовательских работ в области проверки защищенности новых технологий, для сертификации разработок ПО или проведения нагрузочных тестирований.

Множество существующих киберполигонов, нацеленных на коммерческое применение, используют метод моделирования «цифровой двойник» для воссоздания инфраструктуры заказчика.

Катализатором быстрого развития подобных решений для киберполигонов можно считать возросшие темпы развития облачных технологий, которые позволили сделать процесс создания инфраструктуры более доступным и быстрым.

Западный рынок киберполигонов представляет собой множество разработок, предназначенных в первую очередь для применения в военной сфере для обучения персонала проведению успешных кибератак и защите от них [3, 4]. На основе совместной разработки в области киберполигонов министерства обороны США и университета Джона Хопкинса регулярно проводятся киберучения «Locked Shields». Однако, существуют и коммерческие решения, позволяющие компаниям организовать киберучения или обучение персонала, провести функциональное тестирование или анализ защищенности.

Отечественный рынок киберполигонов представлен следующими ПТК:

– «Ampire» – отечественный комплекс киберполигона от компании «Перспективный мониторинг»,

– «Jet CyberCamp» – платформа для киберучений от компании «Инфосистемы Джет»,

– «The Standoff 365» – решение от компании Positive Technologies,

– «Киберполигон» - разработка ООО «Киберполигон»,

– «Cyber Polygon» – решение компании BI.ZONE в сфере киберполигонов,

– «Кибермир» – решение «Ростелеком» в сфере отработки практических навыков по кибербезопасности.

Требования к блоку

При проведении анализа отечественных киберполигонов [5-10] выявлен набор требований к разрабатываемому блоку киберполигона (табл. 1). В частности, реализация настоящего блока должна обеспечивать:

– возможность универсального размещения блока,

– полную доступность администрирования компонентов блока для пользователя,

– возможность многоцелевого использования,

– возможность быстрой инициализации инфраструктуры,

– возможность проведения анализа защищенности эмулируемой инфраструктуры.

Таблица 1

Сравнение возможностей отечественных киберполигонов

Наименование киберполигона	Возможность универсального размещения	Доступность	Многоцелевое использование киберполигона	Быстрое развертывание
Ampire	+	Покупка/Аренда	+	+
Jet CyberCamp	+	Покупка/Аренда	-	+
The Standoff 365	-	В рамках соревнований	-	-
«Киберполигон»	+	Покупка/Аренда	+	-
Cyber Polygon	-	Свободное	-	-
Кибермир	+	Покупка/Аренда	+	+

Возможность универсального размещения, т. е. возможность развертывания компонентов на множестве операционных систем, на различных платформах, будь то облачные решения, физические сервера или пользовательские виртуальные машины. Для этого у модуля необходимо обеспечить полную независимость его компонентов от окружения развертывания. Это реализуется путем предоставления всех необходимых для работоспособности модуля зависимостей, как программных (в виде библиотек, компиляторов или интерпретаторов), так и архитектурных (например, в виде сетевой доступности). Первым способом реализации

данного свойства является обычная настройка окружения на выбранной платформе, как путем проведения ручной конфигурации, так и с помощью средств полуавтоматического управления конфигурацией на подобие «Ansible» или «Chef». Очевидным минусом данного решения является высокая трудоемкость: ручная конфигурация требует много времени, также при проведении ручной настройки высока вероятность неверной конфигурации. Минусы ручной настройки нивелируются средствами управления конфигурацией, позволяющими описать пользовательские процессы с использованием понятного специалистам синтаксисе, например, языка «YAML». Команды выполняются последовательно в формате сценариев. И хотя данный способ является достаточно удобным, сценарии требуют доработки и поддержки, также при их разработке необходимо учитывать все возможные конфигурации платформы. Например, процессы установки пакетов отличаются на разных операционных системах, что требует отражения данных условий в сценариях. Второй способ обеспечения наличия данного свойства блока исключает возможность использовать физические серверы в качестве хоста установки, поскольку требует установки гипервизора. Второй способ представляет собой создание образов виртуальных машин с предопределенной конфигурацией. Процесс создания образа заключается в конфигурировании окружения с сохранением состояния настроек в виде образа виртуальной машины, что позволяет достаточно быстро установить необходимый компонент путем поднятия виртуальной машины с уже выполненными настройками. Однако за данным решением кроется множество проблем. Первым минусом данного решения является фиксированное размещение компонентов по образам по заранее определенной архитектуре, что характеризуется малой гибкостью расположения самих компонентов. Да, возможно вынести каждый компонент в отдельный образ, что безусловно увеличит гибкость, но в это же время потребуется большее количество виртуальных машин. Второй минус – сложность поддержки: обслуживание блока требует досконального знания размещения компонентов по образам, а обновление зависимостей компонентов требует создания новых образов. Третий минус – ресурсоемкость данного решения. Каждая виртуальная машина требует множество ресурсов от хостовой машины, а каждый образ занимает большой объем памяти. Третий способ, заключающийся в контейнеризации приложения, является наиболее удобным и характеризуется простотой реализации. Контейнеризация заключается в создании изолированных сред, использующих ядро хостовой машины. Удобство заключается в возможности создавать отдельные контейнеры на основе выбранной среды контейнеризации (containerd, docker и т.д.) с предопределенными настройками и наборами зависимостей. Контейнеры создаются из образов, которые и представляют набор зависимостей. Благодаря созданию собственных образов для каждого из компонентов возможно обеспечить необходимый набор зависимостей, в том числе и конфликтующих между собой для разных компонентов, а затем поместить туда компонент и каждый раз ожидать одного результата вне зависимости от количества запусков за счет свойства идемпотентности контейнеров. Использование контейнеризации позволяет управлять компонентами как отдельно, так и с помощью оркестратора. Контейнеры используют систему контроля ресурсов, чтобы гарантировать, что каждый контейнер получает только те ресурсы, которые ему необходимы для работы. Контейнеры имеют меньший объем и большую скорость работы, по сравнению с виртуальными машинами, и могут использоваться для развертывания приложений в более эффективном и гибком режиме. Для гарантии быстрого процесса разработки модулей, их конфигурации и идемпотентности в данном блоке был выбран способ обеспечения свойства гибкого развертывания, реализуемый путем использования технологий контейнеризации.

Полная доступность администрирования компонентов для

пользователя. Данное свойство гарантируется предоставлением полного доступа пользователю к исходному коду блока, что позволяет не только изменять функционал в зависимости от требований, но и развивать блок по модели open-source.

Возможность многоцелевого использования: не только для проведения соревнований, но и для обучения принципам обеспечения информационной безопасности (ИБ) и т.д. Данное свойство гарантируется отсутствием привязанности эмулируемой инфраструктуры к каким-либо типизированным шаблонам инфраструктуры. Пользователи имеют возможность самостоятельно разрабатывать архитектуры требуемых окружений, проводить их настройку в соответствии с требуемым сценарием использования.

Возможность быстрой инициализации инфраструктуры: данное свойство обеспечивается применением технологий быстрого развертывания, позволяющих использовать API провайдеров инфраструктуры для управления состоянием инфраструктуры.

Возможность проведения анализа защищенности эмулируемой инфраструктуры. Данное свойство гарантируется благодаря использованию средств обеспечения безопасности с гарантированно актуальными обновлениями баз данных уязвимостей, гибкими настройками и множеством интеграций, предусматривающих параллельное использование других средств обеспечения информационной безопасности.

Компоненты блока

Блок реализован на базе микросервисной архитектуры и состоит из следующих компонентов:

– scp-emulation-frontend – «frontend» компонент, обеспечивающий блок пользовательским интерфейсом, а именно страницами создания окружения (рис. 2-4) и проведения сканирования (рис. 5),

– scp-infrastructure-backend – «backend» компонент, отвечающий за создание модели инфраструктуры по данным, отправленным пользователем с «frontend» компонента. После обработки данных создает записи в реляционной базе данных, которые соответствуют создаваемым сущностям инфраструктуры,

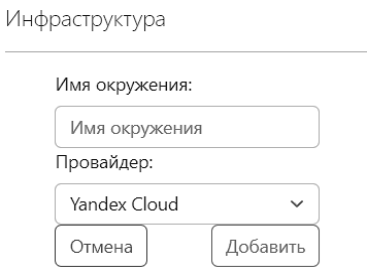

Рис. 2. Элемент пользовательского интерфейса страницы создания окружения

– scp-terraform-backend – «backend» компонент, отвечающий за эмуляцию создаваемой инфраструктуры с использованием выбранного Terraform провайдера,

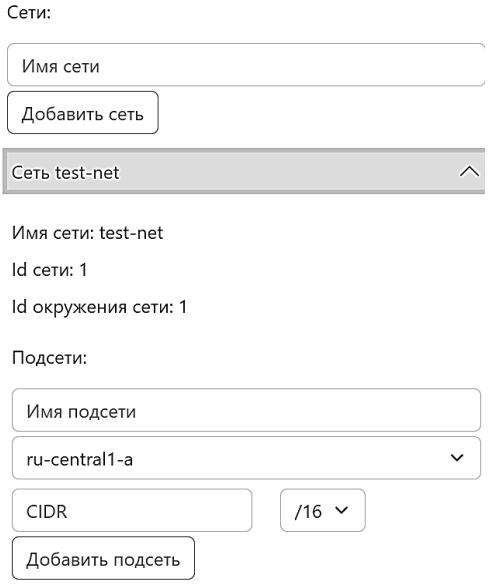

Рис. 3. Элемент пользовательского интерфейса страницы создания окружения

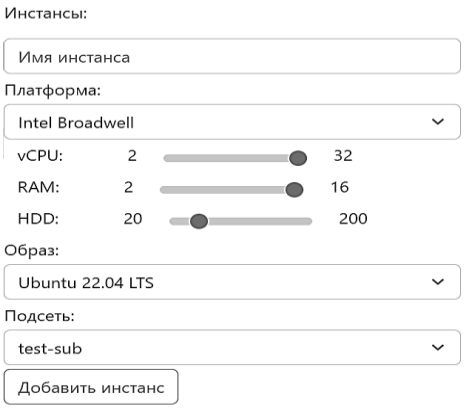

Рис. 4. Элемент пользовательского интерфейса страницы создания окружения

– scp-nessus-backend – «backend» компонент, отвечающий за сканирование пользовательской инфраструктуры с возвратом отчета.

Процесс создания инфраструктуры состоит из следующих действий (рис. 6):

– пользователь переходит в интерфейс создания инфраструктуры,

– пользователь добавляет окружение, выбирая имя и Terraform провайдера, с использованием которого будет эмулировано окружение,

– в добавленном окружении пользователь создает объекты сетей, подсетей и виртуальных машин (инстансов), заполняя формы с параметрами,

– пользователь создает окружение,

– заполненные данные форматируются и «backend» компонент scp-infrastructure-backend создает строки Relation Database Management System (RDBMS) с данными о созданных объектах и их параметрами,

– «backend» компонент scp-terraform-backend получает данные из RDBMS, шаблонизирует с их использованием необходимые манифесты Terraform и применяет конфигурацию.

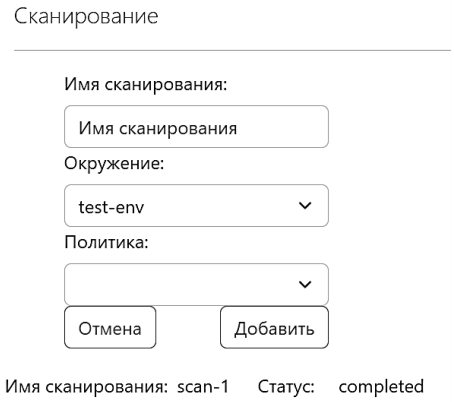

Рис. 5. Элемент пользовательского интерфейса страницы сканирования

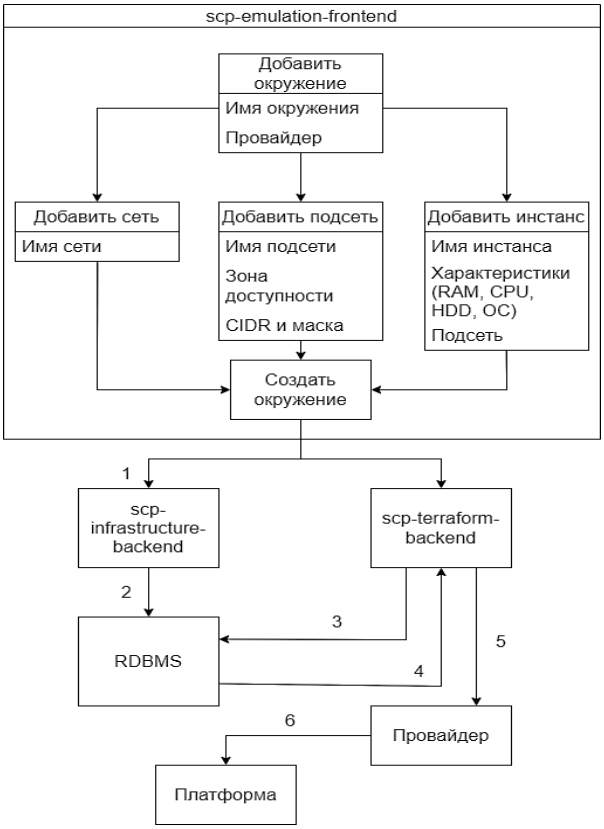

Рис. 6. Алгоритм процесса создания инфраструктуры

Несомненным преимуществом подобного подхода к созданию инфраструктуры по сравнению с использованием Terraform отдельно является возможность одновременно работать с несколькими окружениями и управлять

каждым из них с использованием понятного интерфейса без необходимости вручную управлять множеством директорий и файлов с переменными.

Для хранения данных о создаваемых окружениях и метаданных реализована архитектура для RDBMS (рис. 7), в которой сущности окружения объединены единым элементом, относящимся к таблице «Environment». Таблица «Network» описывает сети, «Subnet» – подсети, «Instance» – создаваемые инстансы, а таблицы «Image» и «Platform» – образы для инстансов и их платформы соответственно.

Процесс сканирования инфраструктуры представлен следующими шагами (рис. 8):

– пользователь переходит в интерфейс сканирования,

– пользователь выбирает созданное окружение,

– пользователь выбирает политику сканирования,

– пользователь создает объект сканирования,

– пользователь выбирает из списка созданных объектов сканирований необходимый и запускает его,

– «backend» компонент scp-nessus-backend получает данные об окружении и проводит сканирование относящихся к нему хостов, результатом которого является отчет.

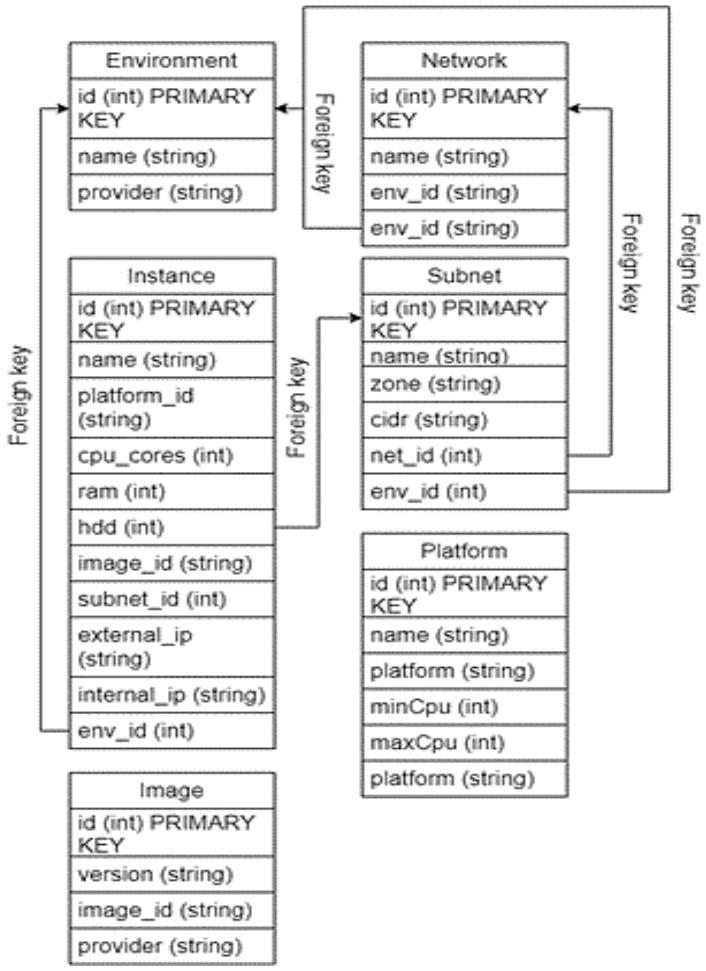

Рис. 7. Архитектура RDBMS

Отчет предоставляется конечному пользователю в формате «html» для удобства ознакомления. Однако, отчет также можно экспортировать из компонента Nessus в множестве других форматов (nessus, CSV, pdf, NessusDB), в том числе поддающихся программному анализу, что позволяет использовать результаты сканирований для дальнейшего анализа в других блоках киберполигона.

В реализованном модуле сканирования в качестве шаблона предоставляемого отчета о результатах сканирования выбран наиболее полный и информативный шаблон. Отчет, составленный по данному шаблону, имеет следующее содержание:

– каждый хост, относящийся к выбранному для сканирования окружению, отображается отдельно и имеет свой набор метаданных (сетевой адрес хоста, операционная система хоста и т. д.),

– каждый хост, относящийся к выбранному для сканирования окружению, имеет множество выявленных уязвимостей, принадлежащих к одному из пяти классов: «Info», «Low», «Medium», «High», «Critical». Nessus определяет класс выявленной уязвимости, используя ряд факторов, включая тип уязвимости, возможные последствия эксплуатации уязвимости для системы, наличие публичных эксплойтов и доступность исправлений. Каждый класс имеет свой порог оценки уязвимости. Например, если уязвимость может привести к компрометации системы и ее данных без необходимости аутентификации, то она будет отнесена к классу «Critical» уязвимостей. Если же уязвимость требует аутентификации или других дополнительных шагов для эксплуатации, то она может быть отнесена к более низкому классу, например, «Medium» или «Low». К классу «Info» относятся уязвимости, позволяющие собрать информацию о системе, например, наличие виртуальных хостов,

– каждая выявленная уязвимость для хоста имеет множество полей с данными о ней, включая краткое описание, подробное описание, полезные ссылки, способы устранения, риск-фактор, оценки по CVSS, VPR и вывод плагина, выявившего уязвимость.

Техническая составляющая

Набор языков программирования, используемый при создании компонентов блока, включает в себя Python и JavaScript, для гипертекстовой разметки документов используется язык HTML.

Набор технологий, использованных при создании блока, включает в себя Terraform, RDBMS, Minio, Docker, Nessus.

Список используемых библиотек и фреймворков включает: Flask, Bootstrap 5, SQLAlchemy, JQuery, Jinja 2, python-terraform, PyTenable.

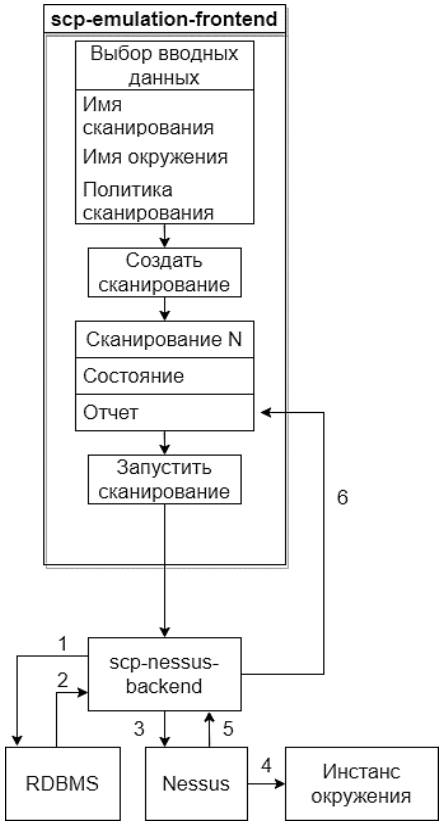

Рис. 8. Алгоритм процесса сканирования

Заключение

Если говорить о возможностях и перспективах развития разработанного прототипа ПТК, являющегося блоком эмуляции инфраструктуры и анализа кибербезопаности в киберполигоне, уместно отметить следующее:

– реализованный ПТК позволяет создавать инфраструктуру киберполигона и предоставлять к ней свободный доступ пользователям для использования как в учебных целях, так и в целях проведения соревнований,

– реализованный ПТК позволяет производить сканирование инфраструктуры на наличие уязвимостей,

– с точки зрения применения ПТК, его установка и использование актуальна в учебных заведениях для обеспечения процесса обучения основам ИБ, а также в качестве инфраструктуры для производственной практики,

– для обеспечения большего выбора провайдеров Terraform необходимо реализовать их поддержку в ПТК.

Список литературы

1. Актуальные киберугрозы: итоги 2022 года. URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2022/ (дата обращения 16.05.2023).

2. Большинство российских компаний готовы тратить до 25% годового бюджета на киберполигоны ИБ. URL: https://www.ptsecurity.com/ru-ru/about/news/bolshinstvo-rossijskih-kompanij-gotovy-tratit-na-do-25-godovogo-byudzheta-na-kiberpoligony-ib/ (дата обращения 16.05.2023).

3. Хочешь мира – готовься к войне или зачем нужен киберполигон. URL: https://www.securitylab.ru/analytics/512874.php (дата обращения 16.05.2023).

4. Обзор рынка киберполигонов. URL: https://www.anti-malware.ru/analytics/Market_Analysis/Cyber-Polygons (дата обращения 16.05.2023).

5. Киберполигон AMPIRE. URL: https://amonitoring.ru/product/ampire/ (дата обращения 16.05.2023).

6. Jet CyberCamp. URL: https://jetcybercamp.ru/ (дата обращения 16.05.2023).

7. Standoff. URL: https://standoff365.com/ (дата обращения 16.05.2023).

8. Киберполигон. URL: https://cyberpoly.ru/ (дата обращения 16.05.2023).

9. Cyber Polygon. URL: https://cyberpolygon.com/ (дата обращения 16.05.2023).

10 Кибермир. URL: https://cybermir.ru/ (дата обращения 16.05.2023).

Воронежский государственный технический университет
Voronezh State Technical University

Поступила в редакцию18.05.2023

Информация об авторах

Куликов Сергей Сергеевич – канд. техн. наук, доцент, Воронежский государственный технический университет, e-mail: alexanderostapenkoias@gmail.com

Александр Иванович Саушкин – студент, Воронежский государственный технический университет, e-mail: madpancake@yandex.ru

CREATION OF A CYBERPOLYGON: FORMATION OF THE UNIT OF EMULATION AND SCANNING OF INFRASTRUCTURE

S.S. Kulikov, A.I. Saushkin

The paper proposes an implementation option for a software and hardware complex (SHC), which is a block for analyzing the security of a cyberpolygon. The block allows you to emulate the customer's infrastructure and scan it for vulnerabilities. The infrastructure emulation component ensures its quick initialization and, if necessary, quick recovery. Through the use of technologies that provide rapid deployment, the infrastructure can be raised on various computing resources. The security analysis component allows the user to scan the infrastructure for vulnerabilities and obtain detailed information about them. To ensure transparent integration of the block components for the user and ease of use, a special application has been implemented that acts as a user interface. SHC allows you to implement various ways of placing its components even on less resource-provided platforms through the use of containerization technologies

Keywords: cyberpolygon, information security, emulation, rapid deployment technology.

Submitted 18.05.2023

Information about the authors

Sergey S. Kulikov – Cand. Sc. (Technical), Associated Professor, Voronezh State Technical University, e-mail: alexanderostapenkoias@gmail.com

Aleksandr I. Saushkin – Student, Voronezh State Technical University, e-mail: madpancake@yandex.ru