УДК 004.771 + 004.75 Вестник УрФО № 1(47) / 2023, с. 15–26

Баринов А.Е., Морозов И.А., Мельников К.В., Руденко Е.А.

DOI: 10.14529/secur230102

БЕЗАГЕНТСКИЙ АННОТИРУЕМЫЙ СБОР ПОВЕДЕНЧЕСКОЙ ИНФОРМАЦИИ ПОЛЬЗОВАТЕЛЕЙ В КОНТЕЙНЕРНЫХ ИНФРАСТРУКТУРАХ [1]

Широкое внедрение технологий контейнеризации и приложений микросервисной архитектуры несет в себе новые специфичные риски при использовании и обслуживании. Технически и организационно обеспечение информационной безопасности в контейнерных средах сильно отличается от привычных подходов. В статье описывается специфика организации аудита действий системных администраторов при обслуживании контейнерных инфраструктур. Авторами предлагается технология безагентского аннотируемого сбора поведенческой информации на основе перехвата взаимодействия пользователей с псевдотерминалом и журналирования запуска приложений, описываются ее преимущества и связанные с ней ограничения.

Ключевые слова: *контейнеризация, аудит, ввод-вывод информации, системные вызовы, мультитенантность, облачная инфраструктура, технология BPF.*

Barinov A.E., Morozov I.A., Melnikov K. V., Rudenko E.A.

AGENTLESS ANNOTATED MONITORING OF USER BEHAVIORAL INFORMATION IN CONTAINER INFRASTRUCTURES

The widespread adoption of containerization technologies and microservice architecture applications brings new specific risks in use and maintenance. Technically and organizationally, ensuring information security in container environments is very different from the usual approaches. The paper describes specifics of organizing an audit of the actions of system administrators when servicing container infrastructures. The authors propose a technology for agent-

[1] Исследование поддержано грантом Российского научного фонда (проект № 22-71-10095).

less annotated collection of behavioral information based on intercepting user interaction with a pseudo-terminal and logging application launches, describe its advantages and limitations associated with it.

Keywords: *containers, auditing process, input and outputs, multi tenants, system calls, cloud infrastructure, Berkeley packet filters.*

Компании все чаще используют технологии контейнеризации для запуска приложений микросервисной архитектуры. Контейнерные приложения легче в обслуживании, их проще переносить, они запускаются в любой среде, где возможно развернуть контейнерезатор (например, платформа Docker портирована практически на все популярные операционные системы). Однако, технически и организационно обеспечение информационной безопасности в контейнерных средах сильно отличается от привычных подходов. Большое количество новых сущностей в микросервисных архитектурах и совершенно другой подход при доставке ПО создают свою специфику [1]. Одним из направлений информационной безопасности является мониторинг и аудит, и несмотря на кажущуюся простоту (прямой сбор информации, например, журналы событий, или вывод аварийных сообщений в некоторых случаях даже упрощаются по сравнению с традиционными или виртуальными инфраструктурами, например, сбор информации с консоли контейнера через dockerlogs), имеются определенные сложности со сбором информации о поведении пользователя в контейнере. В частности, в [2–3] отмечается, что зрелые средства анализа действий пользователя в контейнерных средах в настоящее время отсутствуют в открытом доступе. С одной стороны, если сервис является простым, и настраивается исключительно через консоль хостинг-провайдера и не предполагает интерактивной настройки, то для него нет необходимости в поведенческом мониторинге, однако если же внутри контейнера осуществляется сложное взаимодействие приложений тенанта, и его системный администратор имеет доступ к консольной настройке через SSH или WebSSH, то вопрос организаций мониторинга его действий должен лежать на стороне провайдера инфраструктуры (например, в целях расследования инцидентов, вызванных некорректной настройкой или злонамеренными действиями).

В настоящее время существующие на рынке IaaS провайдеры активно развивают решения по журналированию и аудиту действий пользователя в облачных инфраструктурах, в частности для централизованного журналирования применяются такие решения, как Yandex Cloud Logging [4], AWS CloudTrail [5], Google Cloud Logging [6], однако такие решения больше ориентированы на централизованный сбор событий об ошибках приложений (журналирование потоков stderr, stdout), обработку журналов операционных систем и приложений; AWS CloudTrail [5] кроме того обеспечивает журналирование доступа пользователей (системных администраторов) к API платформы; Google Cloud Logging [6] обладает широкими возможностями по телеметрии используемых ресурсов и журналированию запуска процессов. Некоторые IaaS провайдеры, такие как AWS и Yandex Cloud предоставляют также дополнительные сервисы по аудиту действий пользователей такие, как Yandex Audit Trails [7] и AWS Config [8]. При этом Yandex Audit Trails [7] в большей мере ориентирован на мониторинг действий пользователей (системных администраторов) к объектам инфраструктуры тенанта на основе обращений к API. AWS Config в свою очередь представляет собой сервис декларативного описания конфигурации требуемой виртуальной инфраструктуры и обеспечивает непрерывный мониторинг и журналирование потенциальных отклонений от требуемого состояния и связанные с этим действия. При этом вышеуказанные сервисы оперируют лишь со столь высокоуровневыми объектами, как API платформы, и в некоторых случаях журналы приложений и служб, лишь Google Cloud Logging [6] обеспечивает журналирование запуска процессов, при этом в случае использования виртуальных машин с несколькими контейнерами не обеспечивает аннотирование журнала запуска процессов к контейнерам. Отметим, что ни один из вышеупомянутых провайдеров не обеспечивает журналирование взаимодействия пользователей с ПО контейнера посредством командной строки.

Некоторый технологический задел по мониторингу действий пользователей в контейнерах создан в сфере виртуальных лабораторных работ для обучения студентов ИТ-

специальностей в том числе и по кибербезопасности. Виртуальные лаборатории для организации практических занятий по кибербезопасности в последнее время стали очень популярными во всем мире [9–10]. Более того их можно использовать, как при очном обучении (зачастую минуя приобретение дорогостоящего оборудования и обеспечивая масштабируемость на множество обучающихся), так и в обучении с применением дистанционных образовательных технологий. В большинстве случаев проверка корректности выполнения виртуальной лабораторной работы учащимся выполняется посредством тестовых заданий, реже исполняются специальные проверочные программы, скрипты, которые проверяют работоспособность реализованного учащимся задания. Но остается открытым вопрос как нашли студенты правильный ответ, применяя инструменты (под этим можно подразумевать и защиту от списывания). Решения, которые обеспечивают сбор поведенческой информации, также позволяют улучшить обратную связь со студентами. В частности, в виртуальных лабораторных работах по ИТ-дисциплинах может быть несколько путей к правильному решению, и то, как студенты решают эти задания может существенно обогатить их ответы, что, в конечном счете, может быть полезно при оценке [11].

Представленные в открытых источниках способы сбора в основном базируются на двух подходах [12]: сбор ввода и вывода информации на пользовательские виртуальные псевдотерминальные устройства и перехват команд в оболочке при их исполнении. Каждый из этих способов имеет существенные ограничения при сборе информации. Сбор ввода и вывода информации на псевдотерминальные устройства [13] используется в большинстве современных решений по сбору поведенческой информации пользователей. В этом решении предлагается использование давно развиваемого скрипта ttylog.pl, который, в конечном счете, перехватывает информацию с псевдотерминала, используя strace (вызовы read,write), и, таким образом, он успешно регистрирует все входные и выходные данные терминала. Однако существенным недостатком является то, что указанный перехватчик устанавливается внутрь каждого контейнера, что не всегда приемлемо по производительности. Кроме того, указанные инструменты запускаются в пространстве пользователя и всегда могут быть им злонамеренно остановлены. Кроме того, данное программное обеспечение написано на интерпретируемом языке Perl, что несет в себе необходимость также установки самого интерпретатора, а в некоторых решениях [14] производится дополнительная обработка (аннотирование) собранной информации также в контейнере, в том числе и с применением других интерпретируемых языков, в частности Python. В конечном счете, все обработчики помещаются в контейнер и там же исполняются. В итоге, объем установленных скриптов, интерпретаторов и модулей составляет порядка 60-100 Мб (для контейнеров, построенных на базе образов различных ОС), при том что контейнер, разработанный авторами статьи для изучения работы почтового сервера postfix на базе ОС alpine имеет размер всего 33 Мб. При том, что для некоторых инфраструктур требуется запуск более одного контейнера, а количество тенантов в системе может достигать сотен. В итоге для скачивания и дальнейшего анализа посредством инструментов среды контейнеризации скачивается файл, содержащий аннотированные команды.

В решении [15] представлен более легковесный подход, обеспечивающий перехват действий пользователя, основывающийся на переопределении параметра $PS0 действий пользователя (команда, исполняющаяся перед исполнением основной команды), однако он также определяется в пространстве пользователя и может быть злонамеренно отключен им. В дальнейшем информация помещается в системный журнал и отправляется для анализа с помощью демона rsyslogd. К недостаткам этого решения можно отнести то, что сбор информации таким образом можно производить лишь с ограниченного числа командных оболочек, например, bash и его модификации. Более легковесные решения, такие как ash не обладают таким функционалом. Отметим также, что интерпретация параметра $PS0 осуществляется перед исполнением команды, а, следовательно, перехват и анализ вывода команды в этом случае не возможен. Кроме того, при запуске пользователем в рамках сессии альтернативных оболочек, например, ipython и т.д., перехват сессии не осуществляется. При этом авторы решения [15] разработали альтернативные подходы (коннекторы) для специализированных оболочек, например, metasploit, PowerShell и

т.д. Однако на наш взгляд представляется затруднительным разработка таких модулей для всех существующих пользовательских оболочек.

К недостаткам обоих подходов можно отнести – модификацию образа при запуске контейнера, возможность остановки сбора поведенческой информации пользователем. Хотя возможно осуществить запуск и от стороннего пользователя, предоставив соответствующие права, однако для специфической настройки ПО в контейнере зачастую требуются права администратора (суперпользователя). Таким образом, любое из выше рассмотренных средств может быть нелегитимно остановлено. Кроме того, как указано выше, каждый из подходов предполагает избыточную установку стороннего ПО для каждого из контейнеров, что негативно сказывается как на использовании дискового пространства, так и ОЗУ. Кроме того, в [13] отмечается, что подобный сбор информации может пропускать значимые события такие, как запуск сторонних приложений, если они осуществляются пользователем косвенно, например, через сценарий, интерактивную оболочку (например, mc), или из стороннего приложения, для устранения этого недостатка авторы предлагают использовать стороннюю библиотеку [16], основанную на перехвате системного вызова exec на основе технологии LD_PRELOAD (перехват вызовов на основе специализированных библиотек, загружаемых в приоритетном порядке), однако этот подход также имеет ограниченное применение. Во-первых, не все ОС в контейнерах поддерживают эту технологию, например, используемый авторами статьи alpine без существенной дополнительной доработки не обеспечивает работу этой технологии, а кроме того даже в ОС общего назначения, например Ubuntu, если пользователь запустит статически скомпилированную программу этот метод также окажется неработоспособен.

Перед авторами статьи стояла задача обеспечить сбор поведенческой информации пользователей в контейнерной инфраструктуре с минимальным вмешательством в содержание контейнера (или без такового); обеспечить невозможность отключения сбора информации пользователем или изменение собранной информации. При этом помимо сбора информации с терминалов необходимо обеспечить сбор информации о запускаемых процессах.

Для построения исследовательской среды использовалась одна из наиболее популярных Linux-систем Ubuntu 22.04 и система контейнеризации Docker 20.10. Отметим что, двумя основными технологиями контейнеризации являются Docker и LXC. Docker — это легкие автономные исполняемые пакеты, включающие все необходимое для запуска приложений [17]. С одной стороны, LXC обладает большей гибкостью и меньшими накладными ресурсами, поскольку имеет возможность более гибкой настройки сетевых интерфейсов и более простым взаимодействием с ядром хоста, что повышает производительность [18]. Обе технологии обеспечивают изоляцию ресурсов контейнеров посредством механизма контрольных групп (cgroup), реализуемым в ядре Linux. В своей работе мы использовали стек на базе Docker исходя из требований масштабируемости, включая горизонтальную и функциональную. Кроме того, большинство хостинг-провайдеров – AWS, Yandex Cloud, и т.д. также используют технологию Docker в качестве основной. Также в пользу Docker говорит и ее широкое использование для построения комплексов виртуальных лабораторных работ [11].

Для решения рассматриваемой задачи нами было решено использовать технологию Berkley Packet Filter (BPF) [19]. Технология BPF позволяет осуществлять перехват событий ядра ОС, включая осуществление системных вызовов и т.д. В частности, программа strace, упомянутая выше в современных версиях ОС Linux использует эту технологию при отладке процессов и перехвате системных вызовов. Отметим, что для обеспечения работы технологии BPF она должна быть активирована в ядре. Однако для большинства серверных ОС это не является существенным ограничением. В рассматриваемой нами Ubuntu 22.04 все модули, обеспечивающие работу данной технологии активированы в ядре по умолчанию, кроме BPF_PRELOAD. Однако отсутствие этого модуля для рассматриваемой задачи не существенно влияет на функциональность, поскольку он отвечает лишь за предварительную загрузку BPF модулей в ядро ОС и обеспечивает безопасный доступ к ним из пользовательского режима (рис. 1).

Упомянем, что контейнер - это более слабая технология изоляция процессов, чем, например, виртуальные машины, и таким образом все процессы контейнера (в общем случае, без настройки специфичной изоляции со

```
andrey@andrey-virtual-machine:~$ cat /boot/config-5.15.0-46-generic | grep BPF
CONFIG_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_ARCH_WANT_DEFAULT_BPF_JIT=y
# BPF subsystem
CONFIG_BPF_SYSCALL=y
CONFIG_BPF_JIT=y
CONFIG_BPF_JIT_ALWAYS_ON=y
CONFIG_BPF_JIT_DEFAULT_ON=y
CONFIG_BPF_UNPRIV_DEFAULT_OFF=y
# CONFIG_BPF_PRELOAD is not set
CONFIG_BPF_LSM=y
# end of BPF subsystem
CONFIG_CGROUP_BPF=y
CONFIG_IPV6_SEG6_BPF=y
CONFIG_NETFILTER_XT_MATCH_BPF=m
CONFIG_BPFILTER=y
CONFIG_BPFILTER_UMH=m
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_STREAM_PARSER=y
CONFIG_LWTUNNEL_BPF=y
CONFIG_BPF_EVENTS=y
CONFIG_BPF_KPROBE_OVERRIDE=y
CONFIG_TEST_BPF=m
```

Рис. 1. Состав BPF модулей в ОС Ubuntu 22.04

стороны платформы контейнеризации) видны во внешней ОС и доступны для отладки. Таким образом, решение на базе скрипта ttylog.pl на первый взгляд кажется рабочим, однако при активации доступа пользователем к контейнеру через SSH (сетевой протокол прикладного уровня, позволяющий производить удаленное управление операционной системой посредством псевдотерминала) в каждом из контейнеров будет создаваться свой набор псевдотерминалов пользовательских сессий недоступных во внешней ОС, но при этом упоминающийся в аргументах запуска процессов.

Рассмотрим подробнее работу ttylog.pl. Во-первых, этот инструмент на основе регулярных выражений находит процесс sshd (один из популярных SSH серверов в таблице процессов на основе регулярных выражений), а затем при инициализации сессии пользователя определяет номер псевдотерминала используемого им (обычно в ОС Linux номера псевдотерминальных устройств присваиваются последовательно tty1, tty2 и т.д.; pts1, pts2 и т.д.; ttyS0, ttyS1 и т.д.). После чего подключает отладчик на базе strace к процессу (дочерний процесс sshd, создаваемый для каждой сессии) и на основе перехвата вызовов и выборке полезной нагрузки с помощью регулярных выражений осуществляет реконструкцию пользовательской сессии. Однако, в случае контейнерной инфраструктуры каждый из контейнеров будет создавать свой набор псевдотерминалов не видимый во внешней ОС, при этом каждый из этих псевдотерминалов будет, упомянут в аргументах запуска соответствующих процессов sshdв таблице процессов сервера. Таким образом, при запуске двух контейнеров и подключении к ним через sshd решение с помощью регулярного выражения выберет только один процесс (с меньшим PID) для подключения отладчика, поскольку активируются псевдотерминалы с одинаковым именем. Однако по ряду признаков (идентификаторы процессов, IP-адреса контейнеров в строке запуска) возможно, определить к какому контейнеру относится указанный процесс. С учетом этого сценарий ttylog.pl был доработан авторским коллективом (выборка по имени или адресу контейнера) и сбор пользовательского взаимодействия с псевдотерминалом стал возможен без вмешательства в пространство контейнера.

Рассмотрим вопрос сбора информации о запускаемых процессах. Как упомянуто выше, технологии на основе LD_PRELOAD не обеспечивают достоверный перехват системных вызовов exесв контейнерных инфраструктурах. Однако, напомним, что ядро операционной системы является общим для всех контейнеров, а технология BPF имеет возмож-

ность перехвата в том числе и системного вызова exec. Популярный программный пакет BCC [20] предоставляет набор инструментов для создания эффективных программ трассировки ядра и управления им, а также включает в себя несколько полезных инструментов и примеров. Среди включенных в состав пакета инструментов присутствует execsnoop.py, который обеспечивает перехват системных вызовов execв системе. Отметим, что указанный пакет легко устанавливается на Ubuntu 22.04 поскольку присутствует в стандартном репозитории пакетов под именем bpfcc-tools, при этом требуемый инструмент имеет имя execsnoop-bpfcc. При запуске по умолчанию решение выводит в консоль PID запускаемого процесса, родительского процесса и строку запуска. Однако нам для адекватного сопоставления с действиями в консоли, также требуется определять время запуска процесса. Кроме того, крайне затруднительно определить на основе одних лишь родительских процессов принадлежность процессов к тому или иному контейнеру.

Детальнее рассмотрим возможности инструмента execsnoop.py. Не представляет затруднений добавить в вывод команды время посредством соответствующего аргумента. Кроме того, инструмент имеет возможность выборки запускаемых процессов по контрольным группам (cgroup). Как упомянуто выше, Docker также использует механизм контрольных групп при своей работе. В общем случае новая контрольная группа создается при старте контейнера. При этом с помощью execsnoop.py возможно осуществить выборку процессов, относящихся к задаваемому множеству контрольных групп. Это может быть удобно, в частности в мультитенантных инфраструктурах, а также в виртуальных лабораторных работах, когда изучаемая инфраструктура представлена более чем в одном контейнере. В результате журнал активности (запуска процессов) в контейнерах одного тенанта формируется в едином месте. При этом подмножества контейнеров, над которыми выполняется мониторинг могут быть пересекающимися, это удобно в том случае, когда несколько тенантов разделяют некоторый разделяемый общий ресурс (файловое хранилище, сервис совместной работы и т.д.).

Множество контрольных групп для execsnoop.py задается посредством механизма разделяемой памяти BPF – карт BPF (BPFmaps). Карты BPF представляют собой динамическую, разделяемую ядром и пользовательским пространством структуру хранения данных, организуемую, как массив, хэш-таблицу, стэк, очередь и т.д. [21]. execsnoop.py использует библиотеку пакета BCC [20] containers.py для работы с контрольными группами. В качестве разделяемого ресурса используется хэш-таблица, в которой перечислены идентификаторы cgroupId контрольных групп, мониторинг которых требуется обеспечивать. Поскольку, карта BPF является динамическим ресурсом, то добавление контрольной группы, возможно, организовать без перерыва мониторинга других контейнеров. Таким образом, при запуске очередного контейнера тенантом соответствующий ему cgroupID должен быть помещен в карту BPF, соответствующую тенанту, для обеспечения мониторинга.

Обратим также внимание, что при длительной работе ОС возможно повторное назначение cgroupID, который уже присваивался ранее создаваемой и завершившей свою работу контрольной группе. Это, во-первых, может создать коллизии при мониторинге (в журнал мониторинга тенанта будут поступать записи с другого тенанта или системной контрольной группы), во-вторых, может вызвать переполнение записей в BPF карте (по умолчанию объем ограничен 1024 записями).

Авторами предлагается следующее решение (рис. 2). В данной диаграмме активностей представлен жизненный цикл аккаунта тенанта с точки зрения обеспечения мониторинга его ресурсов. Начнем с того, что администратор инфраструктуры тенанта создает аккаунт с функциями управления контейнерами посредством платформы доставки сервиса (интерфейса или прикладного API) IaaS платформы. На этом этапе нами предлагается создание BPF карты ассоциированной с данным тенантом. После того как администратор тенанта запускает контейнер для подключения к нему мониторинга необходимо добавить его в соответствующую BPF карту. Поскольку принадлежность процессов, определяется идентификатором cgroupID (8 байт), его необходимо определить после запуска контейнера. Этот идентификатор назначается подсистемой cgroup ядра операционной системы после запуска контейнера. Отметим, что все файловые объекты инфраструктуры cgroup во время исполнения контейнера (в случае среды контейнеризации docker) располагаются в специализированной файловой системе

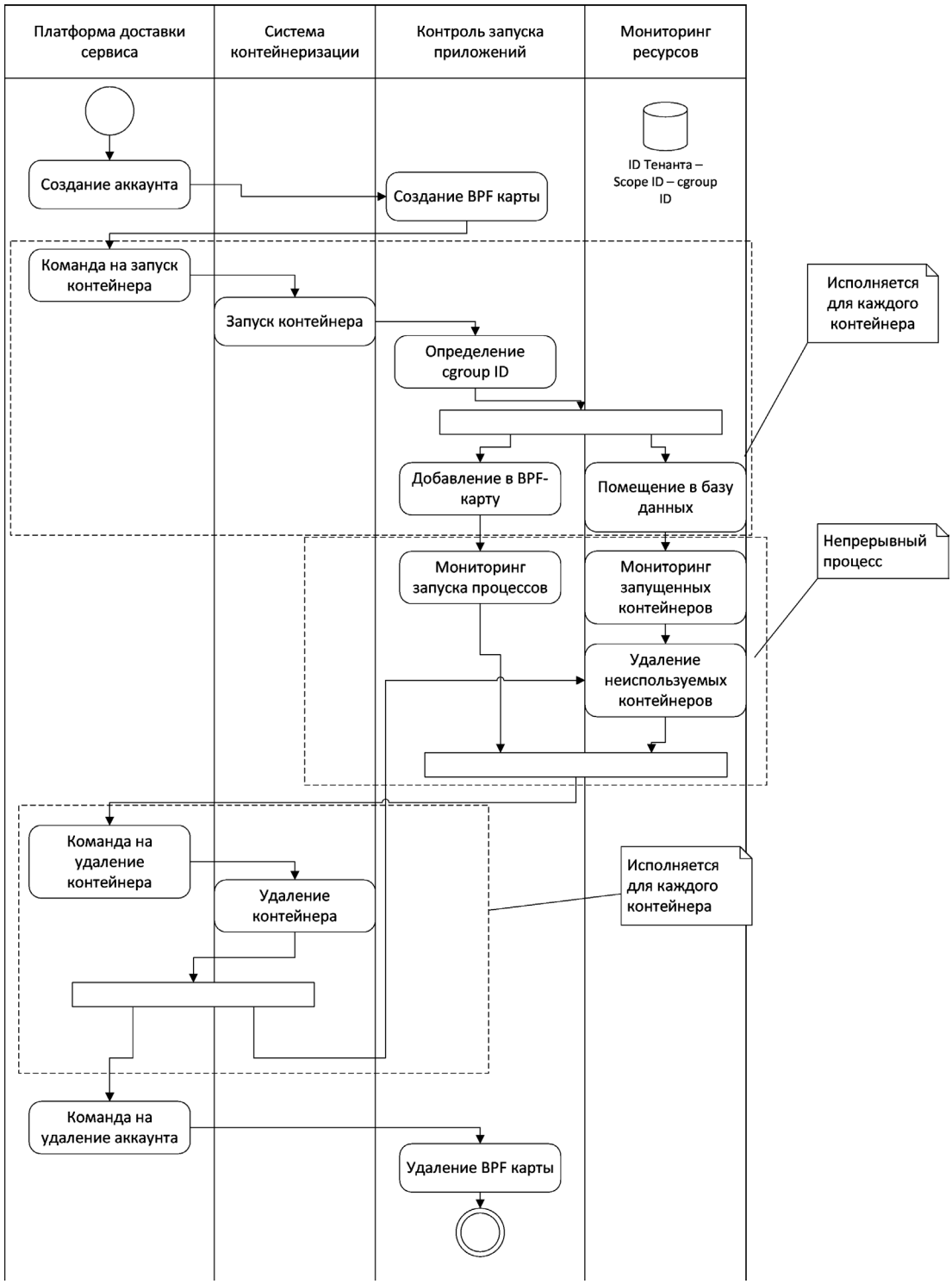

Рис. 2. Диаграмма активностей предложенного решения по запуску процессов

cgroupfs, которая поумолчанию монтируется в /sys/fs/cgroupпри этом пространство контейнера в ней определяется, как /sys/fs/cgroup/user-<uid пользователя, от которого запущен docker>.slice/docker-<ScopeID>.scope или sys/fs/cgroup/system.slice/docker-<ScopeID>.scope. Где ScopeID – это 32 байтный уникальный псевдослучайный идентификатор контейнера, назначаемый сервисом dockerпри создании контейнера. Очевидно, что уникальность идентификации контейнера по ScopeID выше, чем по cgroupID, поэтому

и возникают коллизии (повторные использования cgroupID контейнерами после использования, ранее освобожденного по завершению контейнера). С точки зрения ОС семейства Linux пространство контейнера в файловой системе – это директория, а cgroupID – это ее специализированный атрибут. Обработка специализированных атрибутов не всегда может быть выполнена стандартным ПО дистрибутива, включая ПО из репозиториев. В Ubuntu 22.04 также отсутствует подобное ПО. В [20] предлагается утилита в исходном коде cgroupid для чтения атрибута, и она была пакетирована авторским коллективом.

После определения cgroupID мы добавляем его в соответствующую тенанту BPF-карту, а также размещаем соответствующую запись в служебной базе данных, содержащую также ScopeID и ID тенанта из платформы доставки сервиса (в данном эксперименте мы использовали простой CSV-файл). При удалении контейнера или тенанта штатными средствами мы легко удаляем соответствующие записи из BPF карты или саму карту. Однако если по какой-то причине контейнер не был штатно завершен, и информация об этом не поступила в средство удаления записей мониторинга, то наше решение периодически опрашивает файловую систему на предмет активности пространств, отраженных в таблице. При обнаружении неактивных пространств соответствующие записи удаляются. При этом средству мониторинга запущенных контейнеров необходимо журналировать факт старта контейнера с фиксацией времени, а также PID, запущенных и родительского процесса (containerd-shim-runc-v"1|2"), обеспечивающего запуск контейнера. Также и факт удаления записи об запущенном контейнере должен журналироваться.

Рассмотрим пример работы мониторинга запуска процессов для небольшой системы из двух контейнеров и двух тенантов (для упрощения понимания). Один из ресурсов для них является общим, а один из тенантов (tenant1) также владеет и уникальным контейнером. На рис. 3 представлен вывод терминала содержащий, соответствующие идентификаторы и содержимое BPF карт. На рис. 4 представлен вывод мониторинга процессов с двух тенантов, как можно заметить, встает вопрос определения принадлежности того или иного процесса конкретному контейнеру при этом предлагается решение этой проблемы за счет определения принадлежности родительского процесса первой записи к конкретному контейнеру, а затем последовательной обработки записей. Используя журналы системы мониторинга ресурсов определяем, какой из процессов, к какому контейнеру относился.

```
andrey@andrey-virtual-machine:~$ sudo docker container list
[sudo] password for andrey:
CONTAINER ID   IMAGE                          COMMAND               CREATED          STATUS              PORTS                     NAMES
7861b4f3b686   postfix:latest                 "/usr/sbin/sshd -D"   2 minutes ago    Up About a minute   22/tcp, 25/tcp            nervous_chandrasekhar
f67c823d0c60   edurange2/ubuntu-sshd:16.04    "/usr/sbin/sshd -D"   7 days ago       Up 50 minutes       0.0.0.0:49155->22/tcp   gs_nat
andrey@andrey-virtual-machine:~$ cgroupid hex /sys/fs/cgroup/system.slice/docker-7861b4f3b686b0d5ad822f8fb042caf55fef07bd236be0cef774a6c2027e300c.scope/
4d 2b 00 00 00 00 00 00
andrey@andrey-virtual-machine:~$ cgroupid hex /sys/fs/cgroup/system.slice/docker-f67c823d0c60f1ad545965f2648e052b8e41e460d5217c6bca22486c5dc00daa.scope/
08 2a 00 00 00 00 00 00
andrey@andrey-virtual-machine:~$ sudo bpftool map create /sys/fs/bpf/tenant1 type hash key 8 value 8 entries 128 name cgroupset flags 0
andrey@andrey-virtual-machine:~$ sudo bpftool map create /sys/fs/bpf/tenant2 type hash key 8 value 8 entries 128 name cgroupset flags 0
andrey@andrey-virtual-machine:~$ sudo bpftool map update pinned /sys/fs/bpf/tenant1 key hex 4d 2b 00 00 00 00 00 00 value hex 00 00 00 00 00 00 00 00 any
andrey@andrey-virtual-machine:~$ sudo bpftool map update pinned /sys/fs/bpf/tenant1 key hex 08 2a 00 00 00 00 00 00 value hex 00 00 00 00 00 00 00 00 any
andrey@andrey-virtual-machine:~$ sudo bpftool map update pinned /sys/fs/bpf/tenant2 key hex 4d 2b 00 00 00 00 00 00 value hex 00 00 00 00 00 00 00 00 any
andrey@andrey-virtual-machine:~$ sudo bpftool map dump pinned /sys/fs/bpf/tenant1
key: 4d 2b 00 00 00 00 00 00  value: 00 00 00 00 00 00 00 00
key: 08 2a 00 00 00 00 00 00  value: 00 00 00 00 00 00 00 00
Found 2 elements
andrey@andrey-virtual-machine:~/cgroupid$ sudo bpftool map dump pinned /sys/fs/bpf/tenant2
key: 4d 2b 00 00 00 00 00 00  value: 00 00 00 00 00 00 00 00
Found 1 element
```

Рис. 3. Используемые идентификаторы контейнеров.

```
TIME     PCOMM            PID    PPID   RET ARGS
23:56:17 sshd             6115   5679     0 /usr/sbin/sshd -D -R
23:56:23 sshd             6119   5120     0 /usr/sbin/sshd -D -R
23:56:23 bash             6124   6123     0 /bin/bash -c /usr/local/src/ttylog/start_ttylog.sh
23:56:23 start_ttylog.sh  6124   6123     0 /usr/local/src/ttylog/start_ttylog.sh
23:56:23 grep             6127   6125     0 /bin/grep ^sftp
23:56:23 grep             6130   6128     0 /bin/grep ^scp
23:56:23 tty              6132   6124     0 /usr/bin/tty
23:56:23 hostname         6133   6124     0 /bin/hostname
23:56:23 sudo             6134   6124     0 /usr/bin/sudo cat /usr/local/src/logs/count.NAT
23:56:23 cat              6135   6134     0 /bin/cat /usr/local/src/logs/count.NAT
23:56:23 sudo             6137   6124     0 /usr/bin/sudo tee /usr/local/src/logs/count.NAT
23:56:23 tee              6138   6137     0 /usr/bin/tee /usr/local/src/logs/count.NAT

TIME     PCOMM            PID    PPID   RET ARGS
23:56:17 sshd             6115   5679     0 /usr/sbin/sshd -D -R
23:56:46 bash             6202   6201     0 /bin/bash
23:57:03 date             6204   6202     0 /bin/date
```

Рис. 4. Вывод мониторинга процессов для рассматриваемого примера. Слева для tenant1, справа для tenant2

Рассмотрим подробнее рисунок 4. Рамкой показаны процессы, относящиеся к одному контейнеру в одном тенанте. Однако, очевидно, что не все процессы в представленном журнале выстраиваются в единую цепочку родитель-потомок. Такие процессы без явного родителя отмечены на рисунке 4 стрелками. Это объясняется тем, что execsnoop.py

перехватывает лишь системный вызов exec (загрузки исполняемого образа в ОЗУ), в то время как новый процесс может быть создан системным вызовом fork без загрузки нового исполняемого образа в ОЗУ. Однако, в таком случае возможно определить принадлежность к контейнеру процесса посредством systemctl. А вопрос перехвата системного вызова fork является задачей дальнейших исследований. Также недостатком представленного вывода средства мониторинга является формат вывода времени перехваченного события, такой формат не удобен для дальнейшей машинной обработки и сопоставления с действиями пользователя на терминале. Но встроенные средства ОС позволяют его сконвертировать в более удобное целочисленное представление Unix-времени.

Решение [13] в составе системы мониторинга имеет модуль предварительной обработки журнала взаимодействия с терминалом, который конвертирует журнал в формат csvi формирует такие поля, как входная строка пользователя; уникальный идентификатор, формируемый на основе имени пользователя, тенанта и номера; предварительный класс записи; целочисленное представление Unix-времени; имя пользователя; текущая директория; вывод терминала. Поскольку выводной формат предложенной авторами реализации ttylog.pl идентичен, мы используем тот же механизм постобработки. Кроме того, в соответствии с дополнительной информацией, полученной с модуля сбора информации о запускаемых процессах имеется возможность дополнить эту информацию (при этом обработка должна производится в конвейерном режиме и также необходимо определять текущую директорию процесса, определить же имя исполняющего пользователя в контейнере извне не всегда представляется возможным, например, когда пользователь запускает программу, как службу). Для корреляции событий возможно использовать время, а в случае сложных многопользовательских ресурсов, включая разделенные между тенантами, требуется более детальное определение принадлежности процессов пользователям, а в некоторых случаях возможно ручной анализ. В конечном счете, получаем единый журнал действий в начале для каждого контейнера, а затем для тенанта в целом.

Для дальнейшей обработки решение [13] предлагает производить анализ полученных строк специально задаваемыми регулярными выражениями(вехами) на предмет соответствия следующих признаков имя узла, строки ввода и вывода. Изначально решение предложено для проверки хода выполнения виртуальных лабораторных работ, однако на наш взгляд при созданный подход применим и для обнаружения потенциально нежелательных и злонамеренных действий пользователя. Потенциально каждая веха является сигнатурой и позволяет маркировать события (назначать соответствующий класс) как потенциально опасные. Дальнейший анализ данных в CSV не является удобным в силу того, что требуется находить взаимосвязи между представленными действиями пользователя. В работе [22] представлен формат, удобный для машинной обработки на основе JSON, содержащий следующие поля: время, команда, пользователь, имя контейнера, рабочая директория, кроме того, в проекте [13] отмечается подход конвертации сформированных данных в указанный JSON-формат с обогащением его уникальным идентификатором и классом записи. Как правило, после первичного анализа нет необходимости в хранении вывода команд.

В итоге полученные журналы являются хорошо подлежащими машинной обработке и могут быть проанализированы в системах сбора, анализа, уведомлений и визуализации журналов (ELK, Graylog, Grafana и т.д), в том числе на предмет выявления признаков сложных цепочек подозрительных команд. Таким образом, данный подход мог бы найти свое применение в системах аудита сервисов, предоставляющих услуги контейнеризации (IaaS-платформ); специализированных модулях DLP-систем, а также при организации сред виртуальных лабораторных работ. Ключевым преимуществом, разработанного авторами подхода является полная неинтрузивность, относительно контейнеров тенанта, и прозрачность для пользователя. Недостатки подхода отражены по ходу статьи (проблемы с обработкой вызова fork, более сложное аннотирование журналов на стороне операционной системы), также можно отметить определенные проблемы с обслуживанием решений на базе технологии BPF в связи с ее активным развитием, в частности репозиториях Ubuntu 22.04 на момент написания статьи размещен пакет bpfcc-tools версии 0.18, при том, что используется по умолчанию используется ядро Linux линейки 5.15, с которой совместим лишь набор инструмен-

тов версии 0.23. Не все системные структуры BPF обладают хорошей обратной совместимостью, и поскольку strace весьма популярная утилита, ее работа, как правило, гарантируется с текущим ядром разработчиками дистрибутива. Более редкие же инструменты, например, ttysnoop.py (альтернативный инструмент перехвата информации с псевдотерминала) оказались в силу этого неработоспособны. Работа execsnoop.py пока обладает в целом обратной совместимостью (отсутствует лишь возможность фильтрации процессов по PID родительского), при этом работа не гарантируется при обновлении ядра. Ручное же пакетирование является затруднительным в силу некоторых специфичных особенностей реализации ядра в каждой ОС. В дальнейшем авторы планируют усовершенствовать механизмы выборки информации и обработки служебных данных, а также разработать демонстрационную базу сигнатур.

Литература

1. Юрий Семенюков, Контейнеризация в enterprise: взгляд практика. Jetinfo, №7-8 (307-308) декабрь 2021, 36-43, https://www.jetinfo.ru/wp-content/uploads/2021/12/jetinfo_07_08_2021.pdf

2. Антон Гаврилов, 5000 слов о защите контейнеров. Jetinfo, №7-8 (307-308) декабрь 2021, 44-57, https://www.jetinfo.ru/wp-content/uploads/2021/12/jetinfo_07_08_2021.pdf

3. Ghadeer, D. Security in Kubernetes: best practices and security analysis / D. Ghadeer, H. Jaafar, A. A. Vorobeva // Journal of the Ural Federal District. Information Security. – 2022. – No 2(44). – P. 63–69. – DOI 10.14529/secur220209. – EDN ZYDBOK.

4. ООО «Яндекс.Облако», Yandex Cloud Logging, 2022, https://cloud.yandex.ru/docs/logging/

5. Amazon Web Services, Inc., AWS Cloud Trail: Отслеживание действий пользователей и использования API, 2022, https://aws.amazon.com/ru/cloudtrail/

6. Google LLC, Cloud logging documentation, 2022, https://cloud.google.com/logging/docs

7. ООО «Яндекс.Облако», Yandex Audit Trails, 2022, https://cloud.yandex.ru/docs/logging/

8. Amazon Web Services, Inc., AWS Config: Запись и оценка конфигурации ресурсов AWS, 2022, https://aws.amazon.com/ru/config/

9. Taylor, C.; Arias, P.; Klopchic, J.; Matarazzo, C.; Dube, E. CTF: State-of-the-Art and building the next generation. In 2017 USENIX Workshop on Advances in Security Education (ASE 17); 2017. Available online: https://www.usenix.org/conference/ase17/workshop-program/presentation/taylor (accessed on 4 August 2021).

10. Davis, A.; Leek, T.; Zhivich, M.; Gwinnup, K.; Leonard, W. The Fun and Future of CTF. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, San Diego, CA, USA, 18 August 2014.

11. Richard Weiss, Michael E. Locasto, and Jens Mache. 2016. A Reflective Approach to Assessing Student Performance in Cybersecurity Exercises. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). Association for Computing Machinery, New York, NY, USA, 597–602. https://doi.org/10.1145/2839509.2844646

12. Valdemar Švábenský, Richard Weiss, Jack Cook, Jan Vykopal, Pavel Čeleda, Jens Mache, Radoslav Chudovský, and Ankur Chattopadhyay. 2022. Evaluating Two Approaches to Assessing Student Progress in Cybersecurity Exercises. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 787–793. https://doi.org/10.1145/3478431.3499414

13. Jelena Mirkovic, Aashray Aggarwal, David Weinman, Paul Lepe, Jens Mache, and Richard Weiss. 2020. Using Terminal Histories to Monitor Student Progress on Hands-on Exercises. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 866–872. https://doi.org/10.1145/3328778.3366935

14. R. Weiss, F. Turbak, J. Mache and M. E. Locasto, "Cybersecurity Education and Assessment in EDURange," in IEEE Security & Privacy, vol. 15, no. 3, P. 90–95, 2017, doi: 10.1109/MSP.2017.54.

15. Jan Vykopal, Valdemar Švábenský, Pavel Seda, and Pavel Čeleda. 2022. Preventing Cheating in Hands-on Lab Assignments. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 78–84. https://doi.org/10.1145/3478431.3499420

16. M. A. Eriksen and M. Baker. Snoopy Logger. https://github.com/a2o/snoopy.

17. Karagiannis, S.; Ntantogian, C.; Magkos, E.; Ribeiro, L.L.; Campos, L. PocketCTF: A Fully Featured

Approach for Hosting Portable Attack and Defense Cybersecurity Exercises. Information 2021, 12, 318. https://doi.org/10.3390/info12080318

18. Moravcik, M.; Segec, P.; Kontsek, M.; Uramova, J.; Papan, J. Comparison of LXC and Docker Technologies. In Proceedingsof the 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), Košice, Slovenia,12–13 November 2020; P. 481–486

19. David Calavera, Lorenzo Fontana Linux Observability with BPF: Advanced Programming for Performance Analysis and Networking, ISBN: 9781492050209, December 24th, 2019

20. IOVisor community, BPF Compiler Collection https://github.com/iovisor/bcc.

21. Антон Протопопов, BPF для самых маленьких, часть первая: extended BPF, 11 августа 2020. https://habr.com/ru/post/514736/

22. Valdemar Švábenský, Jan Vykopal, Pavel Seda, Pavel Čeleda, Dataset of shell commands used by participants of hands-on cybersecurity training, Data in Brief, Volume 38, 2021, 107398, ISSN 2352-3409, https://doi.org/10.1016/j.dib.2021.107398.

References

1. Yuriy Semenyukov, Konteynerizatsiya v enterprise: vzglyad praktika. Jetinfo, №7-8 (307-308) December 2021, 36-43, https://www.jetinfo.ru/wp-content/uploads/2021/12/jetinfo_07_08_2021.pdf

2. Anton Gavrilov, 5000 slov o zashchite konteynerov. Jetinfo, №7-8 (307-308) December 2021, 44–57, https://www.jetinfo.ru/wp-content/uploads/2021/12/jetinfo_07_08_2021.pdf

3. Ghadeer, D. Security in Kubernetes: best practices and security analysis / D. Ghadeer, H. Jaafar, A. A. Vorobeva // Journal of the Ural Federal District. Information Security. – 2022. – No 2(44). – P. 63–69. – DOI 10.14529/secur220209. – EDN ZYDBOK.

4. Yandex.Cloud LLC, Yandex Cloud Logging, 2022, https://cloud.yandex.com/en-ru/docs/logging/

5. Amazon Web Services, Inc., AWS Cloud Trail: Track user activity and API usage, 2022, https://aws.amazon.com/cloudtrail/

6. Google LLC, Cloud logging documentation, 2022, https://cloud.google.com/logging/docs

7. Yandex.Cloud LLC, Yandex Audit Trails, 2022, https://cloud.yandex.ru/docs/logging/

8. Amazon Web Services, Inc., AWS Config: Record and evaluate configurations of your AWS resources, 2022, https://aws.amazon.com/config/

9. Taylor, C.; Arias, P.; Klopchic, J.; Matarazzo, C.; Dube, E. CTF: State-of-the-Art and building the next generation. In 2017 USENIX Workshop on Advances in Security Education (ASE 17); 2017. Available online: https://www.usenix.org/conference/ase17/workshop-program/presentation/taylor (accessed on 4 August 2021).

10. Davis, A.; Leek, T.; Zhivich, M.; Gwinnup, K.; Leonard, W. The Fun and Future of CTF. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, San Diego, CA, USA, 18 August 2014.

11. Richard Weiss, Michael E. Locasto, and Jens Mache. 2016. A Reflective Approach to Assessing Student Performance in Cybersecurity Exercises. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). Association for Computing Machinery, New York, NY, USA, 597–602. https://doi.org/10.1145/2839509.2844646

12. Valdemar Švábenský, Richard Weiss, Jack Cook, Jan Vykopal, Pavel Čeleda, Jens Mache, Radoslav Chudovský, and Ankur Chattopadhyay. 2022. Evaluating Two Approaches to Assessing Student Progress in Cybersecurity Exercises. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 787–793. https://doi.org/10.1145/3478431.3499414

13. Jelena Mirkovic, Aashray Aggarwal, David Weinman, Paul Lepe, Jens Mache, and Richard Weiss. 2020. Using Terminal Histories to Monitor Student Progress on Hands-on Exercises. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 866–872. https://doi.org/10.1145/3328778.3366935

14. R. Weiss, F. Turbak, J. Mache and M. E. Locasto, "Cybersecurity Education and Assessment in EDURange," in IEEE Security & Privacy, vol. 15, no. 3, P. 90–95, 2017, doi: 10.1109/MSP.2017.54.

15. Jan Vykopal, Valdemar Švábenský, Pavel Seda, and Pavel Čeleda. 2022. Preventing Cheating in Hands-on Lab Assignments. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 78–84. https://doi.org/10.1145/3478431.3499420

16. M. A. Eriksen and M. Baker. Snoopy Logger. https://github.com/a2o/snoopy.

17. Karagiannis, S.; Ntantogian, C.; Magkos, E.; Ribeiro, L.L.; Campos, L. PocketCTF: A Fully Featured

Approach for Hosting Portable Attack and Defense Cybersecurity Exercises. Information 2021, 12, 318. https://doi.org/10.3390/info12080318

18. Moravcik, M.; Segec, P.; Kontsek, M.; Uramova, J.; Papan, J. Comparison of LXC and Docker Technologies. In Proceedingsof the 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), Košice, Slovenia,12–13 November 2020; P. 481–486

19. David Calavera, Lorenzo Fontana Linux Observability with BPF: Advanced Programming for Performance Analysis and Networking, ISBN: 9781492050209, December 24th, 2019

20. IOVisor community, BPF Compiler Collection https://github.com/iovisor/bcc.

21. Anton Protopopov, BPF dlya samykh malen'kikh, chast' pervaya: extended BPF, 11 August 2020. https://habr.com/ru/post/514736/

22. Valdemar Švábenský, Jan Vykopal, Pavel Seda, Pavel Čeleda, Dataset of shell commands used by participants of hands-on cybersecurity training, Data in Brief, Volume 38, 2021, 107398, ISSN 2352–3409, https://doi.org/10.1016/j.dib.2021.107398.

БАРИНОВ Андрей Евгеньевич, старший преподаватель кафедры защиты информации, специалист по защите информации, и.о. директора НОЦ «Информационная безопасность», ФГАОУ ВО «Южно-Уральский государственный университет (национальный исследовательский университет)». Россия, 454080, г. Челябинск, пр. Ленина, д. 76. E-mail: barinovae@susu.ru

МОРОЗОВ Игорь Александрович, младший научный сотрудник НОЦ «Информационная безопасность», ФГАОУ ВО «Южно-Уральский государственный университет (национальный исследовательский университет)». Россия, 454080, г. Челябинск, пр. Ленина, д. 76. E-mail: morozovia@susu.ru

МЕЛЬНИКОВ Константин Вадимович, студент кафедры защиты информации, ФГАОУ ВО «Южно-Уральский государственный университет (национальный исследовательский университет)». Россия, 454080, г. Челябинск, пр. Ленина, 76. E-mail: dexmoning@gmail.com

РУДЕНКО Евгений Александрович, студент кафедры защиты информации, ФГАОУ ВО «Южно-Уральский государственный университет (национальный исследовательский университет)». Россия, 454080, г. Челябинск, пр. Ленина, 76. E-mail: s1lentsorrow@yandex.ru

BARINOV Andrey Evgenyevich, Senior Lecturer of the Information Security Department, Information Security Specialist, Acting director of the Research and Educational Center "Information Security", South Ural State University (national research university). Russia, 454080, Chelyabinsk, Lenin Ave., 76. E-mail: barinovae@susu.ru

MOROZOV Igor Aleksandrovich, Junior Researcher of the Research and Educational Center "Information Security", South Ural State University (national research university). Russia, 454080, Chelyabinsk, Lenin Ave., 76. E-mail: morozovia@susu.ru.

MELNIKOV Konstantin Vadimovich, student of the Department of Information Security, South Ural State University (national research university). Russia, 454080, Chelyabinsk, Lenin Ave., 76. E-mail: dexmoning@gmail.com

RUDENKO Evgeniy Alexandrovich, student of the Department of Information Security, South Ural State University (national research university). Russia, 454080, Chelyabinsk, Lenin Ave., 76. E-mail: s1lentsorrow@yandex.ru