Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение   
высшего профессионального образования   
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

**Методическое пособие**

по теме**: «IDS Suricata. Развертывание системы. Возможности. Реагирование на инциденты ИБ. Как работать с зашифрованным трафиком»**

Екатеринбург 2020

**Оглавление**

[Введение 3](#_Toc42703918)

[1. Основы IDS Suricata: 4](#_Toc42703919)

[2. Установка Suricata 7](#_Toc42703924)

[3. Лабораторный практикум 8](#_Toc42703925)

[Заключение 11](#_Toc42703926)

[Литература 12](#_Toc42703927)

# Введение

В 2009 году несколько частных компаний и US Department of Homeland Security создали организацию Open Information Security Foundation (OISF), основной из задач которой было финансирование и разработка многопоточной альтернативы Snort, получившей название Suricata. Работа над новой IDS/IPS шла гораздо быстрее, чем со Snort 3.0, и бета-версия появилась уже в декабре 2009-го, а первый официальный релиз — летом 2010-го. Разработка ведется с активным привлечением сообщества, поэтому темп очень высок. Код распространяется под лицензией GPLv2, но финансовые партнеры имеют доступ к не GPL-версии движка, которую они могут брать для своих продуктов. В данной работе будет рассмотрены некоторые функции ПО Suricata.

1. **Основы IDS Suricata:**

Система обнаружения вторжений (СОВ) — программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа в компьютерную систему или сеть либо несанкционированного управления ими в основном через Интернет. Соответствующий английский термин — Intrusion Detection System (IDS). Системы обнаружения вторжений обеспечивают дополнительный уровень защиты компьютерных систем.

Системы обнаружения вторжений используются для обнаружения некоторых типов вредоносной активности, которое может нарушить безопасность компьютерной системы. К такой активности относятся сетевые атаки против уязвимых сервисов, атаки, направленные на повышение привилегий, неавторизованный доступ к важным файлам, а также действия вредоносного программного обеспечения (компьютерных вирусов, троянов и червей).

**Обычно архитектура СОВ включает:**

* сенсорную подсистему, предназначенную для сбора событий, связанных с безопасностью защищаемой системы
* подсистему анализа, предназначенную для выявления атак и подозрительных действий на основе данных сенсоров
* хранилище, обеспечивающее накопление первичных событий и результатов анализа консоль управления, позволяющая конфигурировать СОВ, наблюдать за состоянием защищаемой системы и СОВ, просматривать выявленные подсистемой анализа инциденты
* существует несколько способов классифицировать СОВ в зависимости от типа и расположения сенсоров, а также методов, используемых подсистемой анализа для выявления подозрительной активности. Во многих простых СОВ все компоненты реализованы в виде одного модуля или устройства.

**Виды систем обнаружения вторжений**

В сетевой СОВ, сенсоры расположены на важных для наблюдения точках сети, часто в демилитаризованной зоне, или на границе сети. Сенсор перехватывает весь сетевой трафик и анализирует содержимое каждого пакета на наличие вредоносных компонентов. Протокольные СОВ используются для отслеживания трафика, нарушающего правила определенных протоколов либо синтаксис языка (например, SQL). В хостовых СОВ сенсор обычно является программным агентом, который ведет наблюдение за активностью хоста, на который установлен. Также существуют гибридные версии перечисленных видов СОВ.

Сетевая СОВ (Network-based IDS, NIDS) отслеживает вторжения, проверяя сетевой трафик и ведет наблюдение за несколькими хостами. Сетевая система обнаружения вторжений получает доступ к сетевому трафику, подключаясь к хабу или свитчу, настроенному на зеркалирование портов, либо сетевое TAP устройство. Примером сетевой СОВ является Snort.

Основанное на протоколе СОВ (Protocol-based IDS, PIDS) представляет собой систему (либо агента), которая отслеживает и анализирует коммуникационные протоколы со связанными системами или пользователями. Для веб-сервера подобная СОВ обычно ведет наблюдение за HTTP и HTTPS протоколами. При использовании HTTPS СОВ должна располагаться на таком интерфейсе, чтобы просматривать HTTPS пакеты еще до их шифрования и отправки в сеть.

Основанная на прикладных протоколах СОВ (Application Protocol-based IDS, APIDS) — это система (или агент), которая ведет наблюдение и анализ данных, передаваемых с исполь- зованием специфичных для определенных приложений протоколов. Например, на веб- сервере с SQL базой данных СОВ будет отслеживать содержимое SQL команд, передавае- мых на сервер.

Узловая СОВ (Host-based IDS, HIDS) — система (или агент), расположенная на хосте, отслеживающая вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников. Примером является OSSEC.

Гибридная СОВ совмещает два и более подходов к разработке СОВ. Данные от агентов на хостах комбинируются с сетевой информацией для создания наиболее полного представления о безопасности сети. В качестве примера гибридной СОВ можно привести Prelude.

**Пассивные и активные системы обнаружения вторжений**

В пассивной СОВ при обнаружении нарушения безопасности, информация о нарушении записывается в лог приложения, а также сигналы опасности отправляются на консоль и/или администратору системы по определенному каналу связи. В активной системе, также известной как Система Предотвращения Вторжений (IPS — Intrusion Prevention system (англ.)), СОВ ведет ответные действия на нарушение, сбрасывая соединение или перенастраивая межсетевой экран для блокирования трафика от злоумышленника. Ответные действия могут проводиться автоматически либо по команде оператора.

Хотя и СОВ, и межсетевой экран относятся к средствам обеспечения информационной безопасности, межсетевой экран отличается тем, что ограничивает поступление на хост или подсеть определенных видов трафика для предотвращения вторжений и не отслеживает вторжения, происходящие внутри сети. СОВ, напротив, пропускает трафик, анализируя его и сигнализируя при обнаружении подозрительной активности. Обнаружение нарушения безопасности проводится обычно с использованием эвристических правил и анализа сигнатур известных компьютерных атак.

**Свободно распространяемые СОВ**

Snort NIDS Bro NIDS

Prelude Hybrid IDS OSSEC HIDS

Samhain HIDS Suricata

Open Source Tripwire

**Коммерческие СОВ**

(fr) CATNET

Check Point IPS Blade

Check Point IPS

McAfee IPS

IBM ISS Proventia IPS

[**Suricata**](http://suricata-ids.org/) — open source IPS/IDS система. Основана разработчиками, которые трудились над IPS версией Snort. Бета-версия этой IDS/IPS была представлена на суд общественности в январе 2010-го после трех лет разработок. Одна из главных целей проекта — создание и обкатка совершенно новых технологий обнаружения атак. За Suricata стоит объединение Open Information Security Foundation (OISF), которое пользуется поддержкой серьезных партнеров, включая ребят из US Department of Homeland Security. Изначально поддерживается IPv6, для перехвата трафика используются стандартные интерфейсы: LibPcap, NFQueue, IPFRing, IPFW. Движок автоматически определяет и парсит протоколы (IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB, SMTP и SCTP), поэтому в правилах необязательно привязываться к номеру порта (как это делает Snort), достаточно лишь задать действие для нужного протокола. Вывод результатов унифицирован, поэтому можно использовать стандартные утилиты для их анализа. СОВ [Suricata](http://suricata-ids.org/) является многопоточной, благодаря чему обеспечивается повышенная скорость и эффективность анализа сетевого трафика, так же имеет развитые средства инспектирования различных видов трафика. В конфигурациях Suricata допустимо задействование базы сигнатур, развиваемой проектом Snort, а также наборов правил Emerging Threats и Emerging Threats Pro. Исходные тексты проекта распространяются под лицензией GPLv2.

**Преимущества Suricata:**

– использование для вывода результатов проверки унифицированного формата Unified2, также используемого проектом Snort, что позволяет использовать стандартные инструменты для анализа, такие как barnyard2. Поддержка вывода в формате PCAP;

– поддержка автоматического определения протоколов (IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB и т.п.), позволяющая оперировать в правилах только типом протокола, без привязки к номеру порта (например, блокировать HTTP трафик на нестандартном порту). Наличие декодировщиков для протоколов HTTP, SSL, TLS, SMB, SMB2, DCERPC, SMTP, FTP и SSH;

– многопоточность и высокая производительность, способность обрабатывать на обычном оборудовании потоки до 10 гигабит/cек, поддержка IPv6;

– высокопроизводительный механизм сопоставления по маске с большими наборами IP адресов. Поддержка выделение контента по маске и регулярным выражениям. Выделение файлов из трафика, в том числе их идентификация по имени, типу или контрольной сумме MD5;

– возможность использования переменных в правилах: можно сохранить информацию из потока и позднее использовать ее в других правилах;

– использование формата YAML в файлах конфигурации, что позволяет сохранить наглядность при легкости машинной обработки;

полная поддержка IPv6;

– встроенный движок для автоматической дефрагментации и пересборки пакетов, позволяющий обеспечить корректную обработку потоков, независимо от порядка поступления пакетов;

– поддержка протоколов туннелирования: Teredo, IP-IP, IP6-IP4, IP4-IP6, GRE;

– поддержка декодирования пакетов: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, Ethernet, PPP, PPPoE, Raw, SLL, VLAN;

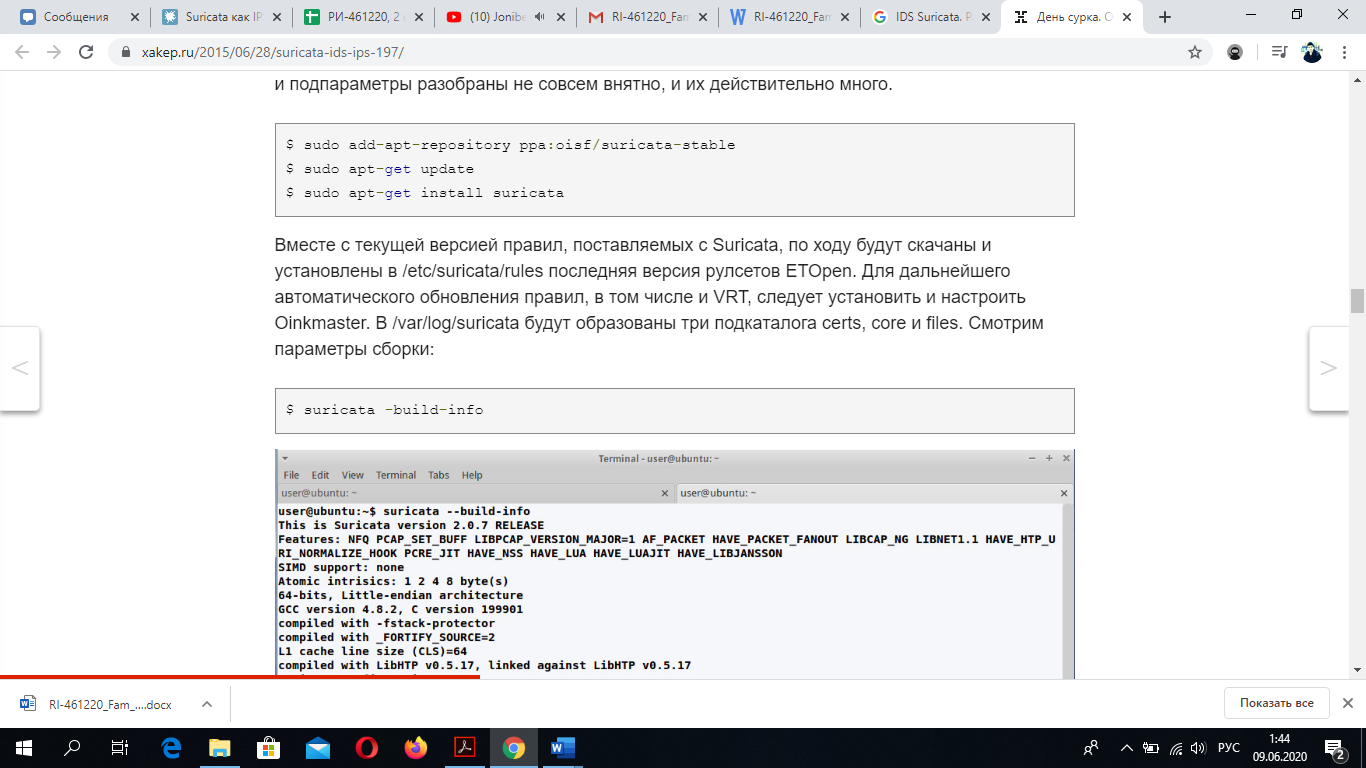
– режим ведения лога ключей и сертификатов, фигурирующих в рамках соединений TLS/SSL;

– возможность написания скриптов на языке Lua для обеспечения расширенного анализа и реализации дополнительных возможностей, необходимых для определения видов трафика, для которых недостаточно стандартных правил.

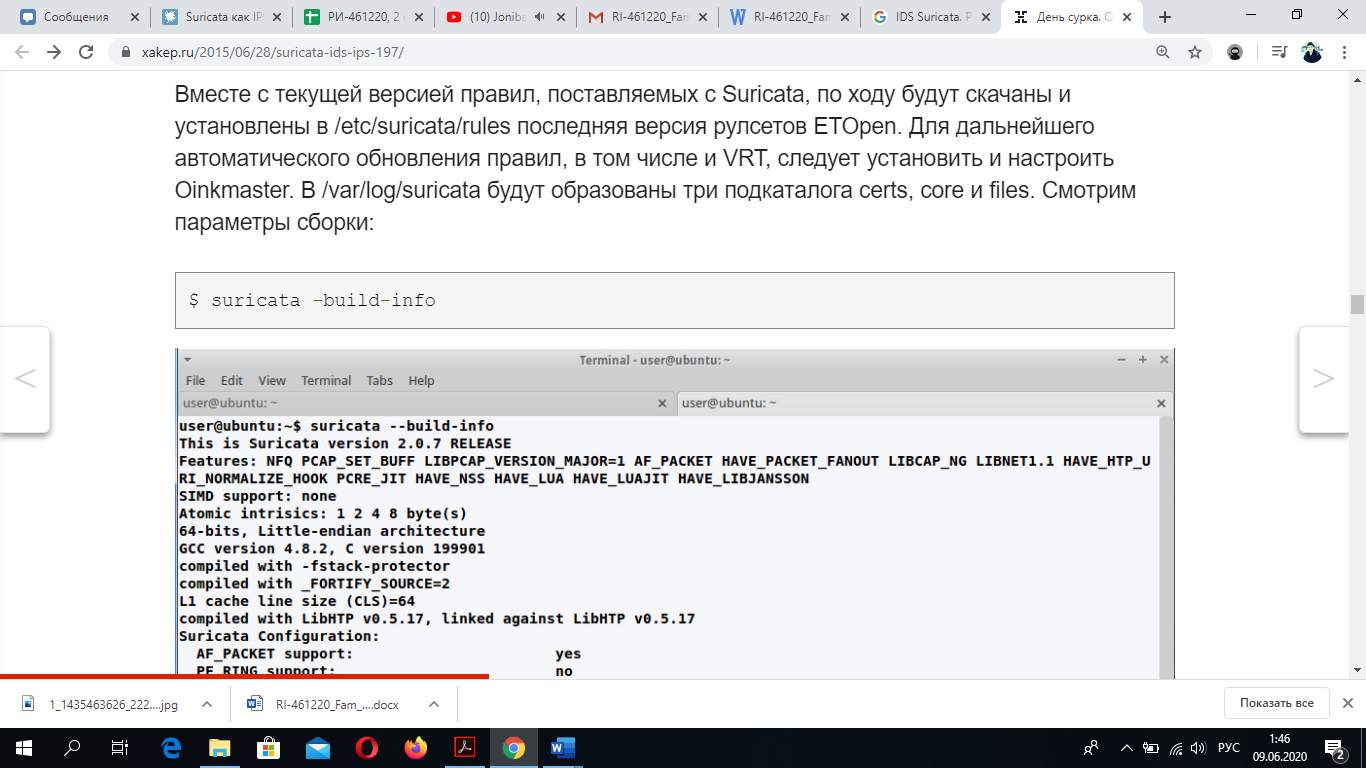
1. **Установка Suricata**

Поддерживается установка Suricata на Linux, BSD, OS X и Win. Проект предлагает исходные тексты и несколько репозиториев Ubuntu (suricata-stable, suricata-beta и дневной срез suricata-daily), также пакеты Suricata добавлены в Debian Backports. Хотя лучшее решение для простого развертывания и поддержки Suricata, вероятно, именно Ubuntu.

Сборка и установка в различных дистрибутивах мало отличаются.



Вместе с текущей версией правил, поставляемых с Suricata, по ходу будут скачаны и установлены в /etc/suricata/rules последняя версия рулсетов ETOpen. Для дальнейшего автоматического обновления правил, в том числе и VRT, следует установить и настроить Oinkmaster. В /var/log/suricata будут образованы три подкаталога certs, core и files. Смотрим параметры сборки:



Изображение выглядит как снимок экрана

Автоматически созданное описание

1. **Лабораторный практикум**
2. Ознакомьтесь с теоретическим материалом.
3. Устанавливайте необходимые программы следующей командой:

*yum install wget gcc libpcap-devel pcre-devel libyaml-devel file-devel zlib-devel jansson-devel nss- devel libcap-ng-devel libnet-devel tar make libnetfilter\_queue-devel lua-devel*

1. В процессе конфигурирования и сборки скаченного и разархивированного пакета Suricata указывайте следующие параметры:

*./configure –prefix=/usr –sysconfdir=/etc – localstatedir=/var–enable-lua && make && make install-full*

, параметр *–enable-lua* включает режим детектирования Lua-скриптов для мониторинга угроз различной сложности.

1. Формируйте кэш динамических библиотек после установки.

для этого воспользуйтесь утилитой *ldconfig*, которая сформирует необходимы ссылки для корректной работы Suricata.

1. В какой каталог устанавливается Suricata в результате выполнения вышеуказанной команды конфигурирования и сборки ?
2. Укажите конфигурационные файлы и лог-файлы. Какие параметры активны?
3. Попробуйте включить неактивные параметры. Какие действительно нужны для режима IDS ?
4. Сохраните настройки вывода в файле определенного раздела или параметра. Для этого используйте команду include или !include.
5. В конфиге много всего плюс комментарии, просмотрите текущие установки и переменные с помощью параметра --dump-config. Сделайте скриншот.
6. Перед первым запуском проверьте значение переменных, определенных в разделе vars, по названию они должны совпадать со снортовскими.
7. Скомпилируйте программу стандартными командами *make, make install*.
8. Проверьте результат установки. Если не получится то попробуйте варианты автоматической установки. *make install-conf, make install-rules, make install-full.*
9. Еще раз проверьте результат установки (командой *suricata –build-info*). Что в этот раз видите? Объясните результат.
10. Посмотрите конфигурационные параметры командой *suricata –dump-config.*
11. В файле *suricata.yaml* укажите значения переменным и выберите режим работы Suricata. Наиболее важными переменными являются:

* vars.address-groups.HOME\_NET: указывается диапазон защищаемой сети
* vars.port-groups.HTTP\_PORTS: указываются необходимые порты
* default-rule-path: указываются необходимые правила
* host-mode: данный параметр определяет режим работы Suricata
* host-os-policy: определяет политику для ОС
* app-layer: осуществляет проверку по определенным протоколам.

1. Попробуйте настроить переменные *threading* и *detect- thread-ratio* чтобы оптимизировать режим работы Suricata в зависимости от аппаратных ресурсов, например, количество ядер*.*

Переменные *stream* и *defrag* отвечают за оптимальную работу с сетевыми данными. Переменная *runmode* определяет режим поточности и очередности обрабатываемых задач.

1. Узнайте возможности различных режимов реализаци с помощью Команды *suricata --list-runmodes*
2. Примените режимы реализации auto и workers. Объясните результаты.
3. В зависимости от аппаратных возможностей можно подобрать оптимальный вариант. В данном случае выберите режим autofp.
4. Рассмотрите режим работы IDS. В данном режиме, если система не выполняет функцию роутера, переменной *host-mode* укажите значение «*sniffer only»,* так как основная задача данного режима – детектирование угроз.
5. Узнайте MTU своей сети командой *ip link.*
6. Установите оптимальное значение default-packet-size, указав MTU своей сети.
7. С помощью --list-app-layer-protos получите список настроенных для проверки протоколов уровня приложения.
8. Проверьте подключенные правила. Для этого перейдите в католог /etc/suricata/ и смотрите файл rules.
9. Какая политика для ОС Linux? Можно ли ее поменять ?
10. Определите ошибки в конфигурационном файле (рекомендуется выполнить команду: *suricata –c /etc/suricata/suricata.yaml –i eth0 –init-errors-fatal )*
11. Запустите Suricata в данном режиме с помощью командой:

*suricata –D –c /etc/suricata/suricata.yaml –i eth0*

1. Зайдите на <https://www.google.ru/> и проверьте в терминале, как работает правило.
2. Теперь нам понадобиться еще одна виртуальная машина, на ней должен быть установлен nmap.
3. Со второй виртуальной машины используйте ping.
4. Педейдите в первую виртуальную машину и посмотрите, как реагирует Suricata.
5. Перейдите в католог var/log/suricata.
6. Найдите файл с информацией о трафике. Сколько пакетов проходили?
7. Оцените производительности CPU при тестировании Suricata в режимах IDS.
8. Счетчик CPU-utilization показывает среднее значение загрузки процессора в процентном эквиваленте, фиксируя время простоя процессора, нагрузку в пользовательском и системном режимах и т.д. CPU-load average фиксирует среднее значение загрузки системы за определенный период времени.

С помощью данных счетчиков определите нагрузку тестируемой системы на ресурсы виртуальной машины. Сделайте скриншоты графики загрузки.

1. Посмотрите файл */var/log/suricata/fast.log*. за что отвечает этот файл ?
2. Закройте и запустите виртуальную машину еще раз.
3. Повторите тестирование для большого потока данных ( рекомендуется запустить не менее часа )
4. Во время попробуйте скачать какой-нибудь файл.
5. Обратите внимание на графику загрузки. Что вы видите? Что изменилось?
6. Объясните результаты.

**Заключение**

Suricata — гибкий инструмент по обработке пакетов, который позволяет менять маршруты в зависимости от содержания пакета, детектировать атаки и предотвращать попадание «плохих» пакетов в систему (например DROP'ать или подменять пакеты, пока они не дошли до WEB сервера). Выполнение лабораторной позволит будущим защитникам информации иметь использовать основные функции этого инструмента.

# Литература

1. Suricata-ids [электронный ресурс] // URL: https://suricata-ids.org
2. Abdinov O., Bondyakov A., Khalilova Sh., Orujova N. XXIV International Symposium NEC 2013, Conception GRID Infrastructure in Azerbaijan, p.9-12.
3. Opennebula [электронный ресурс] // URL: https://opennebula.org