**Содержание**

[1 Задача разработки формализованной методики построения правил для СОВ. 5](#_Toc357842356)

[1.1 Обзор и классификация современных СОВ 5](#_Toc357842357)

[1.1.1 Сигнатурные методы обнаружения вторжений 8](#_Toc357842358)

[1.1.2 Методы обнаружения аномалий 10](#_Toc357842359)

[1.2 Анализ методов повышения технической эффективности СОВ 11](#_Toc357842360)

[1.2.1 Методы технологии мобильных агентов 12](#_Toc357842361)

[1.2.2 Методы построения иммунных систем 15](#_Toc357842362)

[1.2.3 Применение генетических алгоритмов 16](#_Toc357842363)

[1.2.4 Применение нейронных сетей 18](#_Toc357842364)

[1.2.5 Интегральный подход к обнаружению атак 22](#_Toc357842365)

[1.2.6 Комплексная методика решения на основе термов 23](#_Toc357842366)

[1.3 Разработка требований к проектируемой СОВ 24](#_Toc357842367)

[1.4 Постановка задач на разработку формализованной методики построения правил СОВ 26](#_Toc357842368)

[2 Разработка системных решений при проектировании формализованной методики построения правил для СОВ 27](#_Toc357842369)

[2.1 Разработка структурной модели формализованной методики построения правил для СОВ 27](#_Toc357842370)

[2.1.1 Построение модели сети 27](#_Toc357842371)

[2.1.2 Анализ уязвимостей и источников угроз. 29](#_Toc357842372)

[2.1.3 Анализ атак 32](#_Toc357842373)

[2.1.4 Разработка структурной модели построения правил для СОВ 35](#_Toc357842374)

[2.2 Разработка математической модели формализованной методики   
построения правил для СОВ 37](#_Toc357842375)

[2.3 Выбор системы обнаружения вторжений. 41](#_Toc357842376)

[2.3.1 Коммерческие решения 42](#_Toc357842377)

[2.3.1 Некоммерческие решения 45](#_Toc357842378)

[2.4 Выбор анализатора уязвимостей 47](#_Toc357842379)

[3 Разработка программного обеспечения для проверки работоспособности разрабатываемой методики 49](#_Toc357842380)

[3.1 Разработка алгоритмической модели формализованной методики  
 построения правил для СОВ 49](#_Toc357842381)

[3.2 Разработка алгоритма работы программного обеспечения 51](#_Toc357842382)

[3.3 Тестирование разрабатываемой методики 53](#_Toc357842383)

[3.3.1 Определение подхода к расчету эффективности от использования данной методики. 54](#_Toc357842384)

[3.3.2 Описание генератора атак 57](#_Toc357842385)

[3.3.3 Подготовка и тестирование 59](#_Toc357842386)

[3.3.4 Результаты эксперимента 61](#_Toc357842387)

[3.4 Инструкция системного администратора. 62](#_Toc357842388)

[4 Безопасность труда 65](#_Toc357842389)

[4.1 Анализ и обеспечение безопасных условий труда 65](#_Toc357842390)

[4.2 Расчет тяжести и напряженности труда 70](#_Toc357842391)

[4.3 Возможные чрезвычайные ситуации 77](#_Toc357842392)

[5 Экономическое обоснование выпускной квалификационной работы 83](#_Toc357842393)

[5.1 Организационно - экономическая часть 83](#_Toc357842394)

[5.2 Расчет трудоёмкости разработки программного продукта 83](#_Toc357842395)

[5.2.1 Сложность алгоритма 84](#_Toc357842396)

[5.2.2 Сложность организации контроля входной и выходной информации 85](#_Toc357842397)

[5.3 Расчёт себестоимости программного продукта 90](#_Toc357842398)

[5.4 Расчёт экономического эффекта 92](#_Toc357842399)

[Заключение 94](#_Toc357842400)

[Список использованной литературы 95](#_Toc357842401)

[Приложение А](#_Toc357842402) [Конфигурационный файл Snort 98](#_Toc357842403)

[Приложение Б](#_Toc357842404) [Конфигурационный файл программы 110](#_Toc357842405)

# 1 Задача разработки формализованной методики построения правил для системы обнаружения вторжений (СОВ).

По предварительному прогнозу корпорации Symantec на 2013 [1] год выявлена тенденция к росту количества и агрессивности атак. Мнения экспертов сходится в том, что теперь целью атак будет не только заработок или шпионаж, а так же демонстрация силы. Развитие информационных технологий привело к тому, что сегодня конфликты между различным организациями (даже между странами) все чаще переходят в киберпространство. Сегодня термин информационная война в сфере информационной безопасности уже не является чем - то необычным, однако не стоит забывать, что сама концепция информационной войны постоянно развивается. Таким образом, защита информации имеет огромное значение практически для всех сфер общества. Обеспечение должного уровня защищенности информации в киберпространстве сегодня задача государственной важности. Именно поэтому развитие методов защиты информации с целью повышения уровня защищенности информационных ресурсов является сегодня крайне актуальным.

# 1.1 Обзор и классификация современных СОВ

Один из методов защиты киберпространства – это использование систем обнаружения вторжений (СОВ). Система обнаружения вторжений (СОВ) [2] — 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.

Данный метод получил первые предпосылки к развитию четверть века назад, однако официально стандарты только начали формироваться [3].

Системы обнаружения вторжений используются для обнаружения некоторых типов вредоносной активности, которая 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.

Что бы детально рассмотреть не решенные проблемы СОВ был проведен обзор существующих СОВ [4,5,6,7], в соответствии с которым была составлена признаковая классификация СОВ, приведенная на рисунке 1.1.



Рисунок 1.1 – Классификация СОВ в соответствии с признаками

Классификация, представленная на рисунке 1.1, охватывает весь спектр классификационных признаков. Даже такая обширная классификация не может охватить все известные на данный момент решения в области СОВ, однако отталкиваясь от нее, можно рассмотреть все основные направления разработок в этой области.

По принципу действия выделяют два основных класса СОВ: 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.

По расположению сенсоров различают следующие виды СОВ:

* сетевые СОВ (Network-based IDS, NIDS) отслеживают вторжения, проверяя сетевой трафик и ведут наблюдение за несколькими хостами. Сетевые системы обнаружения вторжений получают доступ к сетевому трафику, подключаясь к хабу или свитчу, настроенному на зеркалирование портов, либо сетевое TAP устройство. Примером сетевой СОВ является Snort.
* основанные на протоколе СОВ (Protocol-based IDS, PIDS) представляяют собой системы (либо агентов), которые отслеживают и анализируют коммуникационные протоколы со связанными системами или пользователями. Для веб-сервера подобные СОВ обычно ведут наблюдение за HTTP и HTTPS протоколами. При использовании HTTPS СОВ должны располагаться на таком интерфейсе, чтобы просматривать HTTPS пакеты еще до их шифрования и отправки в сеть.
* основанные на прикладных протоколах СОВ (Application Protocol-based IDS, APIDS) — это системы (или агенты), которые ведут наблюдение и анализ данных, передаваемых с использованием специфичных для определенных приложений протоколов. Например, на веб-сервере с SQL базой данных СОВ будут отслеживать содержимое SQL команд, передаваемых на сервер.
* узловые СОВ (Host-based IDS, HIDS) — системы (или агенты), расположенные на хосте, отслеживающие вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников. Примером является OSSEC.
* гибридные СОВ совмещают 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.

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.

Отдельное внимание в данной классификации стоит уделить методам обнаружения. На сегодняшний день все известные СОВ делятся по данному признаку на 2 большие группы. Именно данный признак описывает саму суть работы СОВ.

# 1.1.1 Сигнатурные методы обнаружения вторжений

Первая группа – это сигнатурные методы обнаружения вторжений. Основная идея работы данной группы методов основана на идеи сравнения потока данных с некоторой базой сигнатур, или шаблонов атаки. Вообще, существует различное трактовка данного термина. Мы же будем понимать данный термин следующим образом. Сигнатура [7] – это множество условий, при удовлетворении которых наступает событие, определяемое как атака или вторжение (обнаружение сигнатур обычно связывают только с совпадением с образцом).

Наиболее распространены следующие методы:

- совпадение с шаблоном;

- совпадение с шаблоном состояний;

- сигнатурный анализ на протоколах (анализ на основе шаблона используемого протокола [7]);

- эвристический подход.

Совпадение с шаблоном – обнаружение в этом случае базирует­ся на поиске фиксированной последовательности байтов в рас­сматриваемом элементе данных (например, в единичном пакете). Как правило, шаблон сопоставляется только в том случае, если подозреваемый пакет ассоциирован с определенной службой (пред­назначен определенному порту).

Достоинства:

- простота задания правил обнаружения;

- прямая связь с используемой данной атакой уязвимостью;

- высокая надежность;

- применим для всех протоколов.

Недостатки:

- может привести к ложному срабатыванию, если шаблон не уникален;

- может потребоваться множество шаблонов для одной уязвимости;

- ограниченность подхода, связанная с анализом только одного пакета;

- существуют варианты «обхода» данного метода.

Совпадение с шаблоном состояния — по одному пакету устанав­ливается состояние потока данных. Появление другого пакета (или пакетов), который соответствует данным состояния, считается атакой.

Достоинства:

- большая эффективность по сравнению с совпадением с шаблоном;

- прямая связь с используемой данной атакой уязвимостью;

- высокая надежность;

- применим к множеству протоколов;

- затрудняет «обход» системы анализа.

Недостатки:

- может привести к ложному срабатыванию, если шаблон не уникален;

- модификация атаки может привести к пропуску обнаружения

- может потребоваться множество шаблонов для одной уязвимости.

Анализ на основе шаблона используемого протокола — для фор­мирования состояния используется декодирование различных эле­ментов протокола. При декодировании протокола СОВ применяет правила, определенные RFC для нарушений. В некоторых случаях эти нарушения могут находиться в определенных полях протоко­ла, что требует более детального анализа.

Достоинства:

- если протокол хорошо определен, то минимизируются пропуски вторжений;

- прямая связь с используемой данной атакой уязвимостью;

- метод может обнаруживать варианты атак;

- надежен, если определены правила протокола.

Недостатки:

- большое число пропусков, если RFC позволяет различные интерпретации;

- сложность формирования сигнатур.

Эвристический подход — использует логические правила, полу­ченные эвристически. Например, для обнаружения сканирования портов можно использовать порог затронутых портов целевой си­стемы. Кроме того, сигнатура может быть ограничена заданием только определенных типов пакета.

Достоинства:

- сигнатуры могут отражать сложные взаимосвязи;

- возможность обнаружения атак, не обнаруживаемых другими методами.

Недостатки:

- разработка эвристик трудоемка и требует высокой квалификации;

- эвристические алгоритмы требуют приспособления к соответствующему трафику.

В общем, данную группу методов характеризуют следующие достоинства:

- высокая достоверность для известных атак;

- сравнительно невысокие требования к потребляемым ресурсам.

К сожалению, огромным недостатком данного метода является то, что он не адаптивен. Эффективность работы данного метода напрямую зависит от:

- эффективности пополнения сигнатурной базы;

- полноты сигнатурной базы;

- наличия интеллектуальных алгоритмов сведения действий атакующих к некоторым базовым шагам.

# 1.1.2 Методы обнаружения аномалий

Вторая группа методов – это методы обнаружения аномалий. К ним относятся:

- статические значения: решение принимается по данным путем их статической предобработки;

- параметрический: строится шаблонный профиль нормального поведения;

- непараметрические: профиль строится из наблюдений в момент обучения системы;

- сигнатуры «правильного» поведения системы;

- другие меры, например, нейронные сети и генетические алгоритмы, позволяющие классифицировать некоторые наборы видимых сенсору признаков, т.н. методы Data Mining.

К сожалению, применение этой группы методов сопряжено появлением ошибок первого и второго рода, особенно в период обучения, и некоторое время после, до тех пор, пока профиль не будет корректно учитывать всех легитимных действий, совершаемых пользователями.

Таким образом, на деле специалисты сталкиваются с 2 довольно нетривиальными задачами:

- построение профиля объекта: трудно формализуемая и затратная по времени задача;

- определение граничных значений характеристик объекта: в целом, это не столько даже задача, сколько проблема, так как всегда может возникнуть перекос в ту или иную сторону.

Основные достоинства подобных СОВ:

- способны обнаруживать новые типы атак;

- не зависят от сигнатурной базы данных;

- генерируют информацию, которую можно использовать в системах обнаружения злоумышленного поведения.

Недостатки:

- системы требуют длительного и качественного обучения;

- системы генерируют много ошибок второго рода;

- слишком медленны и требуют большого количества вычислительных ресурсов.

Вывод из всего этого следующий. На данный момент развития СОВ основными проблемами в этой сфере являются вопросы создания адаптивных сигнатурных СОВ с одной стороны, и вопросы повышения эффективности (снижения вероятности ошибок 1 и 2 рода) для СОВ, выявляющих аномалии, с другой стороны. Кроме того нерешенными остаются вопросы работы СОВ. Ведь СОВ является частью сети, и может быть подвержена атакам, так же, как и любой другой элемент защищаемой сети. Рост же объема трафика в сети заставил специалистов столкнутся еще с одной проблемой. Из года в год количество атак и их сложность растет, вместе с ростом объема трафика. Проблема заключается еще и в том, что чем сложнее СОВ, тем больше вычислительных ресурсов она требует, и, следовательно, СОВ может запаздывать с анализом трафика от реального времени. Кроме того, для работы с СОВ уровень специалиста должен быть крайне высок. И даже не смотря на уровень подготовленности, специалист может допустить ошибку при настройке СОВ.

Однако и развитие СОВ не стоит на месте. Так, например, одним из приоритетных направлений в области создания СОВ является создание гибридных методов обнаружения атак.

# 1.2 Анализ методов повышения технической эффективности СОВ

Один из основателей концепции Data Mining – Г. Пиатецкий-Шапиро – определил данный термин, как процесс обнаружения в необработанных данных:

- ранее неизвестных;

- нетривиальных;

- практически полезных;

- доступных интерпретаций знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Data Mining является мультидисциплинарной областью, возник­шей и развивающейся на базе достижений прикладной статисти­ки, распознавания образов, методов искусственного интеллекта, теории баз данных и др. Отсюда обилие методов и алгоритмов, ре­ализующих в различных действующих системах Data Mining. Многие из таких систем интегрируют в себе сразу несколько подходов.

Рассмотрим основные классы систем Data Mining.

Наиболее широким подклассом предметно-ориентированных аналитических систем, получившим распространение в области исследования финансовых рынков, является «технический ана­лиз», который представляет собой совокупность множества мето­дов прогнозирования динамики цен и выбора оптимальной струк­туры инвестиционного портфеля, основанных на различных эм­пирических моделях динамики рынка.

Последние версии статистических программных пакетов вклю­чают в себя наряду с традиционными статистическими методами и элементы Data Mining.

Нейронные сети — большой класс систем, архитектура кото­рых имеет аналогию (довольно слабую) с построением нервной ткани из нейронов.

Системы рассуждений на основе аналогичных случаев (CBR, case based reasoning) для прогноза на будущее или выбора правильно­го решения ищут в прошлом близкие аналоги ситуации и выбира­ют тот ответ, который был для них правильным.

Деревья решений (decision trees) создают иерархическую струк­туру классифицирующих правил типа if-then, имеющую вид дерева.

Рассмотрим эволюционное программирование. Гипотезы о виде зависимости целевой переменной от других переменных форму­лируются в виде программ на некотором внутреннем языке про­граммирования. Процесс построения программ строится как эво­люция в мире программ. Когда система находит программу, более или менее удовлетворительно выражающую искомую зависимость, она начинает вносить в нее небольшие изменения и отбирает сре­ди построенных таким образом программ-потомков те, которые повышают точность. В результате проведения подобной эволюции система получает несколько генетических линий программ, кото­рые конкурируют между собой в точности выражения искомой зависимости.

Другое направление эволюционного программирования связа­но с поиском зависимости целевых переменных от остальных в форме функций определенного вида. Например, в методе группо­вого учета аргументов зависимость ищется в форме полиномов.

Хотя генетические алгоритмы используются в основном как мощное средство решения разнообразных комбинаторных задач и задач оптимизации, они в последнее время включаются в стан­дартный инструментарий методов Data Mining.

Алгоритмы ограниченного перебора, предложенные в середине 60-х годов XX в. М.М.Бонгардом, вычисляют частоты комбина­ций простых логических событий в подгруппах данных. На осно­вании анализа вычисленных частот делается заключение о полез­ности той или иной комбинации для установления ассоциации в данных, классификации, прогнозирования и т.д.

Указанные методы в последнее время начинают применяться и для решения задач обнаружения атак и вторжений.

# 1.2.1 Методы технологии мобильных агентов

Для получения полной картины о состоянии безопасности в сетях организации некоторые СОВ используют распределенный сбор данных (и определенных функций предварительной обра­ботки) от контролируемых хостов, но функции анализа и при­нятия решения возлагаются при этом на единственный меха­низм.

Такой подход к организации распределенного обнаружения име­ет свои недостатки:

- анализатор (устройство принятия решения) — единственная точка отказа. Если нарушитель выведет ее из строя или обойдет, то система обнаружения не выполнит своих функций;

- ограниченная расширяемость, так как обработка данных в единственной точке ограничивает число контролируемых элемен­тов сети организации;

- сложность с конфигурированием и реконфигурированием СОВ, особенно при добавлении новых функций в СОВ;

- возможность обмана системы, поскольку она отличается от защищаемой системы.

Поэтому в последние годы разрабатываются варианты распре­деленного сбора данных (сенсоров) и распределенного анализа данных (принятия решения). Одной из технологий, которая мо­жет быть использована для построения распределенных СОВ, яв­ляется технология агентов.

В научной и периодической литературе используются термины: агенты, автономные агенты, программные агенты, мобильные агенты и т.д. Под автономным агентом (или просто агентом) по­нимается программный агент, который осуществляет определен­ные функции контроля хоста. Под автономностью агента подразумевается то, что он является независимо исполняемым объектом (его выполнение контролируется только ОС, а не другими объек­тами). Агенты могут получать данные от других объектов системы или агентов, кроме того, агенты могут получать определенные команды.

Поскольку агенты являются независимо исполняемыми объек­тами, они могут быть добавлены или удалены из системы без уве­домления других компонент, т.е. без необходимости рестарта СОВ. Агент может быть частью группы агентов, которые выполняют разные функции, и использовать данные, получаемые другими агентами.

Поэтому использование технологии агентов для сбора и ана­лиза данных имеет следующие достоинства:

- каждый агент можно запрограммировать на получение дан­ных из своего источника (записи аудита, данные опроса системы, перехват сетевых пакетов и др.). Это позволяет совместно исполь­зовать параметры, характерные как для хостовых, так и для сете­вых СОВ;

- возможность независимого старта и остановки работы раз­личных агентов позволяет оперативно в ходе работы СОВ изме­нять содержание (тип) агентов и их количество;

- агенты могут быть программами, написанными на различных языках программирования (тех, которые наиболее приспособле­ны для решаемой агентом задачи).

Использование автономных агентов для обнаружения вторже­ний послужило толчком к разработке различных прототипов СОВ, использующих технологию агентов. В качестве примера рассмот­рим архитектуру системы AAFID (Autonomous Agent For Intrusion Detection, рисунок 1.2 [7]), разработанную в университете Пурду.

Основными компонентами системы являются: агенты, анали­заторы и мониторы.

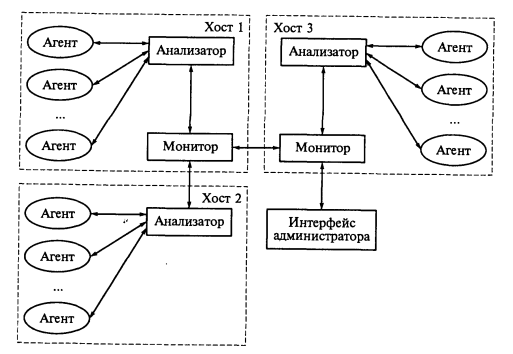


Рисунок 1.2 – Схема архитектуры AAFID для нескольких хостов

Агент — независимо исполняемый объект, который контроли­рует определенные аспекты хоста и докладывает об обнаружен­ных аномалиях или «интересном» поведении соответствующему анализатору.

Каждый хост защищаемой системы может содержать произ­вольное число агентов, контролирующих возникающие на хосте события. Все агенты хоста представляют данные одному анализа­тору. Агент не может генерировать сигнал тревоги.

Анализатор — внешний интерфейс связи каждого хоста. Он контролирует все операции агентов на своем хосте и может оста­навливать их работу, запускать агентов и выбирать им команды. Кроме того, анализатор выполняет функции редукции объема данных, предоставляемых агентами. Результаты своей работы ана­лизатор выдает монитору (одному или более). Можно выделить две основные функции анализатора: управление (запуск и оста­новка исполнения агента на хосте, сохранение записей о деятель­ности агента, ответ на полученные команды от монитора) и об­работка (получение генерируемых агентами данных, обработка и анализ полученных данных, предоставление данных монитору).

Монитор — высокоуровневый компонент архитектуры AAFID. Он также выполняет функции управления (в режиме управления мониторы могут получать команды от других мониторов и выда­вать команды мониторам и анализаторам) и обработки (монитор обрабатывает данные, полученные от анализаторов, поэтому мо­жет обнаруживать вторжения, которые не были обнаружены ана­лизаторами).

Кроме того, мониторы имеют возможность взаимодействовать с интерфейсом администратора, предоставляя администратору возможность доступа ко всем элементам системы.

Таким образом, архитектура AAFID позволяет осуществлять сбор данных от множества источников, распределенных по за­щищаемой системе, и дает возможность комбинировать досто­инства характеристик обнаружения хостовых и сетевых СОВ. Мо­дульность архитектуры позволяет расширять систему, легко ее конфигурировать и модифицировать, что позволяет использо­вать следующие возможности (не реализованные в самом проек­те AAFID):

- обучение агентов с использованием различных методов ма­шинного обучения;

- эволюцию агентов во времени с использованием генетиче­ского программирования;

- сохранение состояния контролируемых сеансов между сеанса­ми, что позволит обнаруживать атаки, протяженные во времени;

- мобильность агентов, т.е. возможность их перемещения с хо­ста на хост при комбинировании AAFID с другими агентскими архитектурами.

Одним из существенных недостатков рассмотренных распреде­ленных СОВ является их уязвимость к атакам против самой систе­мы обнаружения. Компоненты распределенной СОВ находятся в статическом положении и связаны друг с другом иерархической структурой, что позволяет атакующему вывести СОВ из строя, воздействуя на один из высших уровней иерархии. Одним из вари­антов противодействия таким атакам является подход на основе мобильных агентов. При данном подходе неподвижными остают­ся только листья иерархического дерева, а внутренние узлы дере­ва являются мобильными агентами, так как они не имеют непос­редственного взаимодействия с источниками данных хоста или трафика сети. Поскольку внутренние узлы получают данные от других компонентов, обрабатывают их и передают результаты вышележащим узлам, эта обработка может осуществляться в лю­бом месте сети.

# 1.2.2 Методы построения иммунных систем

При построении иммунных систем мы проводим аналогию с иммунной системой организма различать посторонние фрагменты в организме человека. Способность базируется на анализе коротких протеиновых фрагментах. Последовательности системных вызовов так же могут позволить отличить валидную активность от невалидной, как и иммунная система организма отличает воздействия по принципу «свой - чужой». При том, так же следует обращать внимание на время между вызовами.

Использование такой системы проводится в 2 фазы:

1) строится профиль нормального поведения, который не учитывает действия пользователя, а зависит лишь от процессов системы.

2) ведется контроль системы на аномальность по профилю, при этом объектом контроля являются последовательности системных вызовов. Контроль аномалий ведется по 3 значениям: удачное использование, неудачное использование уязвимости, ошибочное условие. Длина записи последовательности вызовов = 10.

Однако если атака не использует привилегированные процессы, то она не может быть обнаружена данным методом.

# 1.2.3 Применение генетических алгоритмов

Генетические алгоритмы (ГА) [7] – класс алгоритмов, называемых эволюционными.

Суть ГА состоит в следующем. Фиксируется начальная популя­ция — множество наборов решений задачи, которые достаточно далеки от точного решения. Для каждого члена популяции вычис­ляется значение функции «согласия» с решением. Один из вариантов – функция средней ценности популяции (формула 1.1).

|  |  |
| --- | --- |
|  | (1.1) |

где – значение функции согласия от данного индивида популяции.

ГА осуществляет переход от нынешней популяции к следующей таким образом, чтобы значение увеличилось. При этом количество строк новой популяции зависит от – коэффициента новизны популяции. Если , то в новой популяции присутствуют строки из старой, иначе, при , все строки новой популяции – это потомки.

Для изменения значения функции средней ценности к индивидам применяют операторы селекции, скрещивания и мутации.

Селекция – попарно выбираются члены текущей популяции с максимальным значением целевой функции (то-есть с вероятностью, определяемой степенью близости). Т.е. оператор селекции строится таким образом, что бы родителем мог стать каждый член текущей популяции с вероятностью > 0. Иногда допускается случай, когда один родитель может выступать в качестве родителя 1 и родителя 2 для индивида новой популяции.

Оператор скрещивания – оператор, результатом работы которого является новое решение, полученное методом взаимного включения коэффициентов двух решений, выбранных в качестве родителя, например с вероятность 0,5.

Оператор мутации – новое решение подвергается случайной модификации. Модификация производится заменой одного случайного параметра на противоположный с определенной вероятностью, например в 0,1%.

Функция ценности строки может быть сопоставлена целевой функции элемента определяющего строку. При том, функция ценности не зависит от алфавита строки, и поэтому м.б. применима ко всем строкам. Наша задача найти строки с наибольшей ценностью.

Для задачи обнаружения вторжений применение ГА сводится к нахождению гипотез для тех событий, вектор которых может представлять собой вторжение. Схема анализа на основе ГА включает 2 стадии:

- кодирование решений (популяции) проблемы строкой бит.

- нахождение значений функции пригодности для каждого члена популяции с целью проверки решений на удовлетворение некоторому эволюционному критерию.

В системе GASSATA генетические алгоритмы применяются к проблеме классификации системы событий с использованием множества векторов гипотез {h} (один вектор на поток интересу­ющих систему событий) размерностью n (n – число потенциаль­но известных атак). По определению h = 1, если оно представляет атаку, и h = 0, если атака отсутствует.

Функция пригодности содержит риск того, что частная атака присутствует в системе. Этот риск умножается на значение векто­ра гипотезы, и полученное произведение корректируется в соот­ветствии с квадратичной функцией штрафов для выделения не­реальных гипотез. Данный шаг улучшает различимость атак. Цель процесса состоит в оптимизации результатов анализа так, чтобы вероятность обнаружения реальной атаки приближалась к еди­нице, а вероятность ошибки в обнаружении атаки стремилась к нулю.

В проведенных экспериментах средняя вероятность точного об­наружения реальной атаки составила 0,996, а средняя вероятность ложных срабатываний — 0,0044. Время, требуемое для настройки фильтров (функции пригодности), было незначительным. Множе­ство из 200 атак потребовало для системы 10 мин 25 с для оценки записей аудита, генерируемых средним пользователем за 30 мин интенсивного использования системы.

Далее приводится пример применения ГА для построения правил обнаружения атаки. Изначально случайным образом строятся правила для обнаружения атак, при этом не факт, что данные правила атаку обнаружат. Затем начинают применять различные атаки. Те правила, которые обнаруживают атаку награждаются, а те которые нет, наказываются. Таким образом полезность одних правил растет, а других падает. Для применения ГА правила преобразовываются в язык строк. Таким образом, мы смещаем наилучшие правила к правилам аномального поведения. Применяя операторы ГА и функцию оценивания мы получаем наилучшую популяцию для поиска аномалий. Подобный подход позволяет определить множество локальных максимумов вместо глобального, что позволить создать популяцию, полезную на всем пространстве атак для данной системы. Для нахождения множества локальных максимумов можно применить технологию ниш, которая заключается в группировании и разделении членов популяции. При группировании используются наиболее сходные члены популяции для замедления приближения к локальным максимумам членов следующей популяции. При разделении уменьшают сходные элементы в нише для достижения локального максимума, который менее популятивен. Для оценки сходства могут использоваться различные метрики, например, расстояние Хэмминга между битами хромосом.

Однако особенностью применения оператора мутации является возможность выхода за разрешенные пределы, что требует постоянный контроль. Кроме того нерешенными остаются вопросы определение числа хромосом начальной популяции, вероятностей скрещивания и мутации, а так же допустимое число поколений. Все это не дает использовать ГА в качестве метода СОВ в настоящий момент, однако данный метод является перспективным.

# 1.2.4 Применение нейронных сетей

Нейронные сети (НС) — это составляющая часть искусствен­ного интеллекта, в которой для обработки сигналов использу­ются явления, аналогичные происходящим в нейронах живых существ. Важнейшая особенность сети, свидетельствующая о ее широких возможностях и огромном потенциале, состоит в па­раллельной обработке информации всеми звеньями сети. При гро­мадном количестве межнейронных связей это позволяет значи­тельно ускорить процесс обработки информации.

Другое не менее важное свойство — способность к обучению и обобщению накопленных знаний. Натренированная на ограничен­ном множестве данных сеть способна обобщать полученную ин­формацию и показывать хорошие результаты на данных, не ис­пользовавшихся в процессе обучения.

Существуют различные способы объединения нейронов между собой и организации их взаимодействия, что привело к созданию сетей различных типов. У каждого типа сети свой способ подбора весов межнейронных связей (т.е. метод обучения).

Искусственные нейронные сети представляют собой попытку использования процессов нервной ткани для выработки новых технологических решений. Нейрон является основным элементом нервной системы (НС). У нейрона есть тело с множеством органелл, называемое сомой. Из сомы выходят множество отрезков, играющих ключевую роль в его взаимодействии с другими нервными клетками. Выделяют два типа отрезков: многочисленные тонкие часто ветвящиеся – дендриты и более толстые, расщепляющиеся на конце, аксоны.

Входной сигнал поступает через синапсы, а выходной сигнал отводится аксоном к другим нейронам через свои многочисленные разветвления на конце, называемые колатералами. Колатералы контактируют с сомой и дендритами других нейронов, образуя очередные синапсы. Передача сигналов внутри нейрона считается сложным процессом.

В модели МакКаллока - Питтса (предложенной в 1943г.; далее на рисунке 1.3 представлена схема модели) нейрон считается бинарным элементом.

В данной модели входные сигналы xj (j = 1,2,…,n) суммируются с учетом весов ωij для каждого сигнала и затем сравниваются с пороговым значением ωi0 (u – аргумент функции активации, функция 1.2). Результат сравнения является аргументом функции активации, при помощи который мы определяем выходной сигнал (yi – выходной сигнал, функция 1.3).

|  |  |
| --- | --- |
| , | (1.2) |

|  |  |
| --- | --- |
| , | (1.3) |

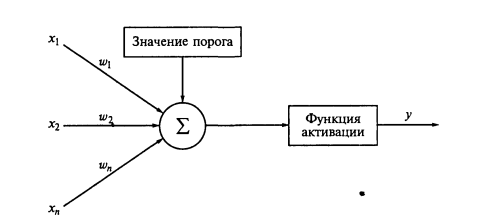


Рисунок 1.3 – Схема модели МакКаллока-Питса

Весовой коэффициент – это вес семантических связей. Положительное значение веса соответствует возбуждающим синапсам, отрицательное значение веса – тормозящим синапсам, нуль значение же соответствует отсутствию связи между i-ым и j-ым нейронами. Модель МакКаллока – Питтса – это дискретная модель, в которой состояние i-ого нейрона в момент времени t+1 рассчитывается по значениям его входных сигналов в момент времени t.

Через несколько лет Хебб предложил теорию обучения неронов. Он предположил, что в результате функции активации вес межнейронных связей увеличивается. Таким образом вес коэффициента (формула 1.4) в момент времени (t+1) зависит от результатов функции активации i-ого и j-ого нейронов в момент времени t (ωij(t+1)).

|  |  |
| --- | --- |
| , | (1.4) |

где t – моменты времени (номер цикла),

η – коэффициент обучения.

В 1962 году Розенблатт в своей книге представил теорию моделирования динамических нейронных систем для моделирования мозговой деятельности, основанную на перцептронной модели нервной клетки, таким образом развив представление нейрона МакКаллока – Питтса, в которой функция активации принимала бинарное значение.

Эта теория представляет модель одноуровневой нейронной сети (НС) из одиночных перцептронов.

Идея применения НС в СОВ заключается в предположении, что нынешняя активность пользователя может быть предсказана на основе параметров и характеристик предыдущего поведения. В настоящее время исследования применения НС активно ведутся в области сетевых СОВ. Это обуславливается следующими достоинствами применения НС:

- способность к быстрой обработке данных вследствие параллельности самой структуры НС;

- способность к обучению и самоорганизации.

Сетевые СОВ на основе НС могут анализировать данные в режиме реального времени, что является одним из основных требований СОВ.

В зависимости от типов применяемых НС в сетевых СОВ направления исследований можно классифицировать следующим образом:

- многоуровневые НС прямого распространения(Multi Layers Feed-Forward, MLFF) — первые работы с такими сетями фокусировались на обнаружении аномалий поведения пользователя (например, рассмотрение перечня исполняемых пользователем команд);

- рекуррентные и адаптивные НС — в таких системах произ­водится анализ реакции защищаемой системы или сети на ре­зультат работы нейронной сети, что позволяет оценивать корре­ляцию текущих выходов нейронной сети с предыдущими входами и состояниями. Примером таких систем является ELMAN (Cere­bellar Model Articulation Controller);

- нейронные системы без учителя — большинство таких си­стем использует самоорганизующиеся шаблоны (Self-Organizing Maps, SOM) для обучения характеристикам нормальной систем­ной активности и идентификации статистических отклонений от нормальных характеристик.

Один из примеров применения НС в СОВ – это СОВ с BP-классификатором (Back Propogation, рисунок 1.4 [7]).



Рисунок 1.4 - модель СОВ с BP-классификатором.

Основными компонентами такой СОВ является монитор пакетов, модуль извлечения свойств, BP-классификатор и генератор тревоги. Монитор пакетов – это простая сетевая карта, работающая в режиме приема каждого пакета. В системе Windows для реализации может быть применена библиотека winpcap. Модуль извлечения свойств – это своего рода фильтр, который формирует вектор параметров, представляющих собой важные для системы характеристики пакетов. Векторы описывают активность в системе, в том числе и вредоносную. Далее приведена структура вектора:

Attack (type) = (P-id, H-len, C-sum, S-port, D-port, Icmp-type, Icmp-code, Flag, P-len, P-data)

Модель BP-классификатора является наиболее распространенной моделью применения НС в задачах распознавания образов. Для получения BP - классификатора необходимо решить следующие задачи:

- определить число скрытых уровней: согласно теории 3-уровневая мо­дель ВР - классификатора может представить соответствие размер­ности n к размерности m, поэтому для большинства приложений одного скрытого слоя достаточно;

- определить размерность входного слоя: определяется числом выбранных параметров для анализа (двоичным представлением параметров), размерность выхода — числом решений, которое будет прини­мать классификатор;

- определить размерность скрытого слоя: определить трудно — реальное число входов скрытого уровня определяется в результате экспе­риментов;

- выбрать функцию передачи: определяется экспериментально;

- инициализировать весы и пороги: производится случайными числами в заданном диапазоне (ωij ϵ [0; 1], ui ϵ [-0,5; 0,5]).

Процесс формирования СОВ включает в себя три основные стадии: обучение на массивах тренировочных знаний, тестирование правильности работы на других массивах данных и обнаружение атак в ходе эксплуатации. На рисунке 1.5 приведена схема обучения классификатора.

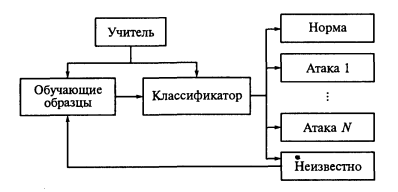


Рисунок 1.5 – Схема обучения классификатора.

Если полученный результат – неизвестен, то это может свидетельствовать о новой атаке. Тогда, после анализа, вектора такой атаки могут быть вставлены в НС. Подобная СОВ является адаптивной.

Если к такой СОВ добавить модуль оценивания, то такая СОВ может быть самообучаемой. Применение НС в СОВ дает достаточно высокие результаты распознавания атак.

Проблемы применения НС похожи на проблемы применения генетических алгоритмов при решении задач СОВ. Они эффективны, но решающую функцию приходится получать экспериментально. Как и генетические алгоритмы, применение методов НС для решения задач распознавания пока не достаточно хорошо проработано.

# 1.2.5 Интегральный подход к обнаружению атак

В статье Аграновского [6] предлагается использовать интегральный подход к определению вторжений, что позволит связать последовательность шагов атаки в единую картину сценария, таким образом, создав механизм связи между теорией и практикой реализации атаки. То есть, этот механизм поможет определить взаимосвязь рисков и угроз с уязвимостями и реализуемыми в данный момент злонамеренными действиями. Определения картины сценария атаки позволит выявить, на что атака направлена, и спрогнозировать возможное развитие данной атаки. Такое прогнозирование строится посредством графа переходов и статистике наиболее частого развития атак, которые направлены на конкретную цель. Определив фазу атаки можно, используя граф перехода, спрогнозировать ее дальнейшее развитие с целью ее устранения. Использование такого подхода позволит так же определять атаки, распределенные во времени и пространстве. Автор предлагает разбить выявление атаки на несколько уровней:

- на нижнем уровне определяются примитивы (происходит выявление фазы по сигнатурам);

- средний уровень агрегирует данные нижнего уровня, и выявляет сценарий атаки;

- верхний данный агрегирует данные предыдущих двух уровней и системы и выявляет атаку, простую и распределенную, их источник, а так же может произвести прогноз.

Кроме того предлагается неоднородные правила выполняются раздельно, но, вместе с тем, и схожие операции, используемые в правилах, выполняются раздельно. Автор предлагает свести схожие операции в единый список шаблонов, а правило разбить на части. Совпадение потока с сигнатурами нескольких шаблонов в таком бы случае значило исполнение одного из правил. А последовательность совпадений определяла бы тип атаки. Это позволило бы сократить количество исполняемых операций. Есть и минус, когда шаблоны связаны друг с другом, то определить атаку, таким образом, не представляется возможным. Выход, вынесение таких правил в отдельный шаблон целиком.

Идея с применением интегрального подхода, на мой взгляд, имеет недостаток. В случае распределенной атаки злоумышленник, во-первых, может запутать СОА, изменив стандартную схему атаки, а во-вторых, в случае распределенной атакой, даже зная следующий шаг злоумышленника, система и специалист, работающий с ней, вряд ли смогут подгадать время следующей фазы и источник атаки.

Преимущества у такого подхода тоже есть. Самое главное – это возможность прогнозирования атаки. Спрогнозировав атаку, мы можем усилить защиту предполагаемого объекта атаки и установить за ним усиленный контроль. Подобные меры помогут снизить вероятность успешной атаки со стороны злоумышленника.

Идея же сведения однородных операций кажется очень удачной, и на мой взгляд не имеет недостатков, кроме упомянутых автором.

# 1.2.6 Комплексная методика решения на основе термов

Повышение эффективности предлагается провести следующим образом [30]. Известно, что любой протокол имеет пользовательские и служебные данные. Служебные данные протокола могут иметь одинаковые параметры. Тогда для неизвестной системы (неизвестных протоколов) мы формируем набор грамматики данного протокола для данной системы. Подразумевается, что мы исследуем контент служебных данных протокола на предмет выявления последовательностей данных, свойственных при обработке в данной системе. Таким образом, мы формируем профиль нормального поведения. При этом не обязательно знать структуру протокола, так как мы оперируем статистическими данными, что позволяет сделать данную методику применимой к любому протоколу. Частота встречаемости термов и слов – это и есть те значения, которые заносятся в профиль нормы. Сами же термы и слова – это, выявленные в ходе исследования протокола, служебные данные. Причем для анализа их совершенно не обязательно знать, какой эти данные функционал имеют, так как выполняется анализ не функционала протокола, а анализ встречаемости последовательности, которая выполняет ту или иную функцию в условиях конкретной системы. Отклонения от значений встречаемости термов, или появление новых термов в таком случае, и есть аномальное поведение, или, иными словами, изменение функционала, которое не был предусмотрено политикой безопасности и не является безопасным.

Данная методика довольна интересна. Разработав мультиплатформенную реализацию такого подхода, возможно было бы ее универсальное применение к любой системе. Кроме того, автор доказывает экспериментальным путем ее эффективность.

Все же и такую систему можно обмануть, используя ликвидные термы профиля для организации атаки. Кроме того, вероятность ошибок второго рода в реальных системах довести до значений, используемых автором, нельзя, не увеличив при этом вероятность возникновения ошибок первого рода. Так что, мы в любом случае сталкиваемся с проблемой выбора такого диапазона значений для термов, при котором СОВ для данной сети будет работать наиболее эффективно. Кроме того, установка нового ПО может вызвать изменения в потоке трафика, в том числе и термов, что неминуемо будет подразумевать ошибки 2 рода. То есть существуют в сети такие пользователи, для которых создание профиля – задача трудновыполнимая, так как последовательности данных таких пользователей могут изменяться от случая к случаю (хотя и для них все же можно создать профиль, но он будет малоэффективен). Таким образом, СОА с методикой, предлагаемой автором, все же сохраняет при себе все проблемы, свойственные СОА, выявляющие аномалии.

Однако данный метод можно считать довольно большим достижением, так как его использование позволяет увеличить эффективность за счет снижения вероятности ошибок первого и второго рода в целом для системы.

# 1.3 Разработка требований к проектируемой СОВ

Требования, предъявляемые к СОВ, разделены на 2 группы: общие и технико-эксплуатационные требования.

Общие требования, предъявляемые к СОВ, это универсальные требования, которые предъявляются к любому объекту проектирования, независимо от решаемой задачи. Однако набор общих требований может различаться в зависимости от решаемой задачи. Так к проектируемой СОВ на основе формализованной методики построения правил для СОВ предъявляются следующие общие требования:

- надежность – свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения, технического обслуживания, хранения и транспортирования;

- достоверность – несомненная верность (правильность) полученных от системы или собранных ею сведений (данных) для воспринимающего их человека;

- адаптируемость – приспособление к изменяющимся условиям системы (для получения необходимых выходных данных на основе изменения входных);

- модифицируемость – возможность изменения как аппаратных, так и программных средств;

- модульность системы – принцип построения технических систем, согласно которому функционально связанные части группируются в законченные узлы (модули);

- возможность дальнейшего развития и наращивания программных и аппаратных средств подсистемы биометрической идентификации состояния пользователя на основе данных о частоте сердечных сокращений на этапе доступа подразумевает соответственно повышение уровня системы и увеличение количества решаемых разнообразных задач;

- экономическая целесообразность – это есть экономическая эффективность, т.е. соотношение полезного результата и затрат на его получение.

Требования, предъявляемые к СОВ, изображены на рисунке 1.6.



Рисунок 1.6 - Требования к СОВ

Специальные (или технико-эксплуатационные) требования – это требования, предъявляемые к объекту проектирования, исходя из решаемой задачи. В частности для обнаружения атак проектируемая СОВ должна соответствовать следующими требованиями:

- поддержка используемых сетевых протоколов: при использовании сетевых СОВ работа ведется с данными, структура которых регламентируется соответствующими сетевыми протоколами;

- определение критических точек сети: при выборе и установке СОВ необходимо определить наиболее уязвимые сервисы и наиболее значимые узлы в сети. Если это сетевая СОВ, то наиболее значимая точка в сети, это место соединения глобальной сети и локальной. Там же расположен Firewall. В этом случае наиболее уязвимые сервисы - это сетевые сервисы, расположенные на границе глобальной и локальной сети, в демилитаризованной зоне;

- достаточность вычислительных ресурсов – это требования, предъявляемые к оборудованию, на которое устанавливается СОВ для проведения анализа информационных потоков;

- кроссплатформенность – это требование, которое стали предъявлять ко многим программным продуктам (ПО) относительно недавно. Кроссплатформенность – это свойство ПО, означающее возможность установки ПО на различные операционные системы, иными словами возможность применения на различных программно-аппаратных платформах.

# 1.4 Постановка задач на разработку формализованной методики построения правил для СОВ

Темой выпускной квалификационной работы является «Разработка формализованной методики построения правил для системы обнаружения вторжений (СОВ)».

Целью данной работы является повышение технической и экономической эффективности работы СОВ.

Предполагается создание такой методики работы СОВ, при которой обнаружение вторжений определялось бы сравнением с сигнатурами, а для неизвестных атак применялись бы методы обнаружения аномалий, с целью выявления атаки и записи ее в виде сигнатуры в базу данных сигнатур.

Для достижения цели необходимо решить следующие задачи:

* разработать классификацию современных СОВ и провести аналитический обзор методов повышения технической эффективности работы СОВ;
* разработать требования к проектируемой СОВ;
* разработать структурную модель построения правил для СОВ;
* разработать математическую модель построения правил для СОВ;
* выбрать и обосновать необходимое для реализации формализованной методики построения правил для СОВ аппаратное и программное обеспечение;
* разработать алгоритмическую модель построения правил для СОВ;
* разработать алгоритм программного обеспечения, реализующего формализованную методику построения правил для СОВ;
* протестировать разрабатываемую формализованную методику построения правил для СОВ;
* определить меры по обеспечению безопасности жизнедеятельности персонала;
* оценить экономический эффект от результатов разработки.

# 2 Разработка системных решений при проектировании формализованной методики построения правил для СОВ

# 2.1 Разработка структурной модели формализованной методики построения правил для СОВ

При построении системы защиты сети необходимо, прежде всего, знать то, от чего информацию защищают. Если это СОВ, то необходимо провести анализ уязвимостей сети и выявить угрозы и их источники. Зная уязвимости конкретной сети и понимая, на что может быть направлена атака, можно более точно понять, какие атаки на сеть будут производиться, скорее всего. Разрабатывая метод, позволяющий повысить эффективность работы СОВ, необходимо учитывать все те факторы, которые имеют значение и при выборе и установки СОВ.

# 2.1.1 Построение модели сети

Для анализа уязвимостей сети (как и для всех последующих действий) в аспекте решения задачи установки СОВ (или создания метода работы СОВ, что является задачей отчасти похожей), необходимо, прежде всего, решить, о каком типе СОВ идет речь и какие точки сети, на которых возможна установка СОВ, будут критическими [7]. На рисунке 2.1 приведена типовая схема подключения сетевой СОВ.



Рисунок 2.1 – Типовая схема подключения СОВ к сети

Представленная схема подключения является структурной схемой. В реальности же сама СОВ расположена в демилитаризованной зоне. На структурной схеме отчетливо видно, какие потоки данных для сетевой СОВ (IDS – это аббревиатура СОВ) имеют значение. Это не единственный вариант подключения СОВ, но в аспекте проводимой работы данный тип подключения представляет большой интерес, так как точка соединения локальной (LAN) и глобальной сети является одним из наиболее важных рубежей любой сети предприятия. Через данную точку проходит все множество сетевых потоков, направленных из сети, и в сеть. Кроме того, при определенной настройке сети можно заставить все внутри сетевые потоки проходить через данную точку соединения. Данная точка сети в независимости от размера предприятия нуждается в защите. Как правило, данный рубеж обороны сети представляется в виде Firewall (F). Расположение СОВ в данной точке, таким образом, является полностью обоснованным. Располагая СОВ в демилитаризованной зоне, администратор (ADMIN) может вести контроль не только самой сети, но так же и контроль Firewall, так как именно на Firewall может быть направлена первая из множества атак.

Таким образом, на входе сетевой СОВ есть множество X, которое описывается всеми множествами трафика, поступающими на вход IDS в соответствии с выбранным расположением.

X = {Xin, X’in, Xdrop-in, Xout, X’out, Xdrop-out}, (2.1)

где Xin – подмножество входящего трафика, принимаемого фильтрами IDS до файрвола;

X’in – подмножество входящего трафика, поступающего на фильтры IDS после прохождения через фильтры файрфола;

Xdrop-in – подмножество входящего трафика, отсеянного фильтрами Firewall;

Xout – подмножество исходящего трафика, принимаемого фильтрами IDS до файрвола;

X’out – подмножество исходящего трафика, поступающего на фильтры IDS после прохождения через фильтры файрфола;

Xdrop-out – подмножество исходящего трафика, отсеянного фильтрами Firewall.

Понятно, какая информация может быть проанализирована сетевой СОВ, расположенной представленным образом. Для лучшего понимания того, как сетевая СОВ может быть установлена так, что бы контролировать данные потоки, рассмотрим физическую топологию сети. В дальнейшем, при проведении каких либо исследовательских работ предлагается использовать именно такой вариант физической топологии сети (а точнее, физическую топологию подключения сетевой СОВ). На рисунке 2.2 представлен вариант подключения СОВ в структуре конкретной сети.

На представленной модели количество серверов и персональных компьютеров (ПК) может быть не точным (не обязательно именно 5 ПК и конкретный сервер, например WEB). В данном случае физическая топология изображена для наглядности представления того, как в целом будет организована сеть, и каким образом к сети будет подключена СОВ.



Рисунок 2.2 – физическая топология модели сети.

Для сравнения структурной схемы и физической топологии проведем описание физической топологии. Здесь на вход маршрутизатора WAN приходит поток информации из глобальной сети интернет. Firewall на рисунке изображен в виде стены. Две стены не значит, что в сети есть 2 Firewall. Это значит что правила у брандмауэра (Firewall) настроены таким образом, что для внешней сети и для внутренней они различны и есть возможность совместного использования ресурсов демилитаризованной зоны (DMZ).

Таким образом, весь поток информации, поступающий на брандмауэр, который может быть как аппаратным, так и программным (но чаще всего это интегрированное программно-аппаратное решение), проходит через DMZ, где и установлена сетевая СОВ, контролирующая все сетевые потоки. Помимо этого в DMZ располагаются те службы, доступ к которым можно, или необходимо иметь как изнутри сети, так и снаружи.

# 2.1.2 Анализ уязвимостей и источников угроз.

Для представленной модели сети (как и для любой модели сети, в принципе) основными источниками угроз являются следующие [8]:

- интернет – глобальная информационная сеть является основным источником распространения любого рода вредоносных программ. Как правило, вирусы и другие вредоносные программы размещаются на популярных веб-сайтах интернета, "маскируются" под полезное и бесплатное программное обеспечение. Множество скриптов, запускаемых автоматически при открытии веб-сайтов, могут также содержать в себе вредоносные программы.

- электронная почтовая корреспонденция – почтовые сообщения, поступающие в почтовый ящик пользователя или на почтовый сервер (если он есть), хранящиеся в почтовых базах, могут содержать в себе вирусы. Вредоносные программы могут находиться как во вложении письма, так и в его теле. Как правило, электронные письма содержат вирусы и почтовые черви. При открытии письма, при сохранении на диск вложенного в письмо файла вы можете заразить данные на вашем компьютере. Также почтовая корреспонденция может стать источником еще двух угроз: спама и фишинга. Если спам влечет за собой в основном потерю времени, то целью фишинг-писем является конфиденциальная информация (например, номера кредитных карт).

- уязвимости в программном обеспечении – так называемые "дыры" в программном обеспечении являются основным источником хакерских атак. Уязвимости позволяют получить хакеру удаленный доступ к локальной сети, а, следовательно, и к доступным ресурсам локальной сети и другим источникам информации.

- съемные носители информации – для передачи информации по-прежнему широко используются съемные диски, дискеты, карты расширения памяти (флеш). Запуская какой-либо файл, расположенный на съемном носителе, пользователь можете поразить данные на своем рабочем ПК вирусом, что может привести к распространению вредоносного ПО внутри сети.

Обобщая все вышесказанное можно сделать следующее заключение, что основным источником угроз для сети является сам пользователь. Контроль учетных записей и ограничение доступа к ресурсам сети для пользователей сети, а так же ограничение полномочий пользователя позволяют, от части, решить данную задачу. Здесь же можно упомянуть, что и саму СОВ можно настроить таким образом, что бы контролировать действия самих пользователей сети.

Другим источником угроз является спам корреспонденция, но как правило, достаточно хорошо настроенный брандмауэр со спам фильтрами позволяет отсеять большую часть подобной корреспонденции. Здесь так же можно применить СОВ.

Но больше всего СОВ раскрывает себя тогда, когда источником угроз выступает уязвимость ПО. Именно уязвимости ПО, как и было сказано, являются источниками атак на сеть.

Отталкиваясь от типовой модели сети можно провести анализ уязвимостей этой сети. При этом для модели вводятся определенные уточнения. Необходимо уточнить, какие именно службы будут в сети использоваться, и какие клиент-серверные приложения выполняться. Необходимо перечислить используемые сетевые протоколы, что бы знать, по каким портам наиболее вероятно будет проводиться атака. Но для начала, необходимо определить само понятие уязвимости и привести классификацию наиболее распространенных типов уязвимостей в целом.

В компьютерной безопасности, термин уязвимость (англ. vulnerability) [9] используется для обозначения недостатка в системе, используя который, можно нарушить её целостность и вызвать неправильную работу. Уязвимость может быть результатом ошибок программирования, недостатков, допущенных при проектировании системы, ненадежных паролей, вирусов и других вредоносных программ, скриптовых, а также SQL-инъекций. Некоторые уязвимости известны только теоретически, другие же активно используются и имеют известные эксплойты.

Обычно уязвимость позволяет атакующему «обмануть» приложение — заставить его совершить действие, на которое у того не должно быть прав. Это делается путем внедрения каким-либо образом в программу данных или кода в такие места, что программа воспримет их как «свои».

Некоторые уязвимости появляются из-за недостаточной проверки данных, вводимых пользователем, и позволяют вставить в интерпретируемый код произвольные команды (SQL-инъекция, XSS, SiXSS).

Другие уязвимости появляются из-за более сложных проблем, таких как запись данных в буфер без проверки его границ (переполнение буфера).

Метод информирования об уязвимостях является одним из пунктов спора в сообществе компьютерной безопасности. Некоторые специалисты отстаивают немедленное полное раскрытие информации об уязвимостях, как только они найдены. Другие советуют сообщать об уязвимостях только тем пользователям, которые подвергаются наибольшему риску, а полную информацию публиковать лишь после задержки или не публиковать совсем. Такие задержки могут позволить тем, кто был извещён, исправить ошибку при помощи разработки и применения патчей, но также могут и увеличивать риск для тех, кто не посвящён в детали.

Существуют инструментальные средства, которые могут помочь в обнаружении уязвимостей в системе. Хотя эти инструменты могут обеспечить аудитору хороший обзор возможных уязвимостей, существующих в системе, они не могут заменить участие человека в их оценке.

Для обеспечения защищённости и целостности системы необходимо постоянно следить за ней: устанавливать обновления, и использовать инструменты, которые помогают противодействовать возможным атакам. Уязвимости обнаруживались во всех основных операционных системах, включая Microsoft Windows, Mac OS, различные варианты UNIX (в том числе GNU/Linux) и OpenVMS. Так как новые уязвимости находят непрерывно, единственный путь уменьшить вероятность их использования против системы — постоянная бдительность. Таким образом, применение СОВ – это ключевой аспект контроля уязвимостей в сети.

Распространённые типы уязвимостей включают в себя:

- нарушения безопасности доступа к памяти, такие как: переполнения буфера, висящие указатели;

- ошибки проверки вводимых данных, такие как: ошибки форматирующей строки, неверная поддержка интерпретации метасимволов командной оболочки, SQL-инъекция, инъекция кода, E-mail инъекция, обход каталогов, межсайтовый скриптинг в веб-приложениях, межсайтовый скриптинг при наличии SQL-инъекции;

- состояния гонки, такие как: ошибки времени-проверки-ко-времени-использования, гонки символьных ссылок;

- ошибки путаницы привилегий, такие как: подделка межсайтовых запросов в веб-приложениях;

- эскалация привилегий, такие как: Shatter attack.

На представленной модели сети из всех серверов и сервисов представлена только служба WEB-сервера. По факту же, даже в представленной модели, используется намного больше сетевых служб. В таблице 2.1 приведен неполный перечень используемых в модели протоколов.

Таблица 2.1 – Перечень протоколов, используемых в представленной модели

|  |  |  |
| --- | --- | --- |
| Порт назначения | Аббревиатура | Расшифровка |
| 1 | 2 | 3 |
| 23 | telnet | teletype network |
| 53 | dns | domain name service |
| 67 | dhcp v.4 – client | dynamic host configuration protocol |
| 68 | dhcp v.4 – server |
| 80 | http | hypertext transfer protocol |
| 443 | https | http secure |

Данный перечень описывает модель, изображенную наглядно, в достаточно полной мере, однако в реальности может применяться куда гораздо большее количество служб. Любой протокол может содержать уязвимости, и таким образом, быть подвержен атаке. Конкретно для представленной модели, любой из перечисленных выше протоколов содержит большое количество уязвимостей.

# 2.1.3 Анализ атак

Выше уже было сказано, что наибольшую значимость СОВ имеет при работе с уязвимостями. Уязвимость же ПО – это главный источник атак на сеть. Отсюда и название, система обнаружения вторжений, так как основной смысл ее работы заключается именно в контроле сетевых потоков на предмет использования уязвимостей. Используя уязвимость в ПО, хакер реализует атаку на сеть. Таким образом, атака, это не что иное, как реализация какой либо уязвимости. На одном известном интернет ресурсе в статье хакерская атака этому термину дается следующее определение [10]: атака — действие, целью которого является захват контроля (повышение прав) над удалённой/локальной вычислительной системой, либо её дестабилизация, либо отказ в обслуживании. Данные два определения не противоречат друг другу. По сути, атака – это не столько реализация уязвимостей, сколько реализация конкретной угрозы. Но в аспекте сетей под атакой нужно понимать именно данные два определения. Данное утверждение подтверждается классификациями атак на сеть, которые представляют различные авторы.

Классификация по Питеру Меллу

Согласно классификации, представленной Питером Меллом в статье, посвященной СОВ и атакам на сеть [11], все атаки можно разделить следующим образом:

- удаленное проникновение;

- локальное проникновение;

- удаленный отказ в обслуживании;

- локальный отказ в обслуживании;

- атаки с использованием сетевых сканеров;

- атаки с использованием сканеров уязвимостей;

- атаки с использованием взломщиков паролей;

- атаки с использованием анализаторов протоколов.

Подобная классификация предоставляет возможности подробного описания практических реализаций атак, однако при помощи нее нельзя провести анализ модели угроз безопасности. Целью атак по такой классификации является угроза, хотя анализ при помощи данной классификации провести и не получится. А реализация атаки производится за счет какой либо уязвимости. Данная классификации подтверждает приведенные выше определения.

Классификация от компании Internet Security System [31] по признакам

1) по степени риска (ранжирование опасности атак):

- высокий;

- средний;

- низкий.

2) по типу атаки:

- осуществляемые локально;

- осуществляемые удаленно.

3) по подверженному данной атаке ПО.

4) по характеру действий:

- черные ходы;

- DoS;

- DDoS;

- потенциально незащищенная ОС;

- неавторизованный доступ.

В данной классификации не учитываются такие классификационные признаки, как: по последствиям атаки, по цели атаки. Кроме того в каждом из признаков не приводится достаточно полная классификация.

Классификация, используемая в продукте Nessus [29]

Здесь основным признаком классификации атак является характер уязвимости:

- черные ходы;

- ошибки в CGI скриптах;

- DoS;

- ошибки реализации FTP;

- наличие сервиса Finger и ошибки ПО, реализующих этот сервис;

- ошибки реализации Firewall;

- ошибки, позволяющие пользователю, имеющему терминальный вход на данный сервер, получить права администратора;

- ошибки, позволяющие атакующему удаленно получить права администратора;

- прочие ошибки, не вошедшие в другие категории;

- ошибки реализации NIS-серверов;

- ошибки реализации RPC-серверов;

- уязвимости, позволяющие получить любой файл с сервера;

- ошибки реализации SMTP-серверов;

- неиспользуемые сервисы.

Классифицировать атаки при работе с СОВ таким образом наиболее удобно, так как явно указав на уязвимость, за счет которой реализуется та или иная атака, мы можем определить, на реализацию какой угрозы данная атака направлена, и какова будет последовательность действий злоумышленника. Единственный недостаток этой классификации (как, в принципе, и любой другой классификации атак) в том, что она не может охватить все существующие атаки в полной мере из-за постоянного появления новых атак. Если это данная классификация, то данная проблема связана с постоянным появлением новых технологий, и как следствие, новых уязвимостей.

Отсюда можно сделать следующий вывод: любая классификация атак делается для достижения определенной цели, которая, скорее всего, связана с построением защиты какой-либо системы. То есть, классификации самих атак имеют скорее прикладной характер, нежели теоретический. Кроме того полная классификация атак будучи актуальной сегодня, не может быть таковой завтра. При рассмотрении вопросов защиты мы анализируем риски и, уже отталкиваясь от них, выбираем классификации атак, которые могут быть реализованы на систему. Притом, классификация атак напрямую будет связана с построением правил для системы защиты. Отталкиваясь от классификации атак, в первую очередь, и стоятся правила для СОВ. Но самое важное, что нам дает классификация атак, это представление того, какие угрозы, скорее всего, будут реализованы, и каким будет ущерб от их реализации.

Для представленной модели сети, которую предстоит защищать, актуальны следующие атаки [7,32,33,34] (по типам используемых сетевых служб):

1) атаки на telnet:

- ошибки реализации telnet-серверов;

- перехват пароля, передаваемого протоколом telnet;

- манипуляция переменными окружения;

- модификация файла rhosts.

2) атаки на DNS-сервера:

- ошибки реализации dns-серверов;

- DDoS атаки.

3) атаки на DHCP-сервера:

- атака на отказ в обслуживании;

- поддельный DHCP-сервер;

- атака «Man in the middle».

4) атаки на http: разрешение методов put и delete и т.д.

Данная классификация описывает рассматриваемую модель сети. Не стоит забывать то, что данная модель создана с целью проведения экспериментов по улучшению эффективности работы СОВ в целом. Следовательно, и классификация охватывает лишь те аспекты, которые справедливы для данной модели. В реальности же тот же WEB-сервер, который в модели описывается лишь протоколами http и https, на деле использует куда гораздо большее количество служб, имеющих различного рода уязвимости.

Если же данную классификацию привязывать к основным типам угроз, то ее более чем достаточно для описания всех возможных вариантов атак, реализующих уязвимости именно в представленной модели. Целью атак, представленных в классификации, может стать реализация любой угрозы, будь то отказ в обслуживании или кража пароля. Именно поэтому данной модели сети достаточно для демонстрации работы метода.

Основная причина ограничения модели, как по реализуемым технологиям, так и по типам угроз (если подразумевается модель атаки), в том, что на рассмотрение всех типов атак на всех видах сервиса попросту не хватит времени, так как необходимо вникать в аспекты любой атаки при ее осуществлении на сеть. Отталкиваясь от представленной классификации атак можно создать модель атаки на сеть. В дальнейшем, для ограничения рассматриваемых воздействий на сеть с целью экономия времени в качестве основы для модели атаки предложено использовать лишь DDoS атаки (атаки на отказ в обслуживании). Выбор данного типа угроз обусловлен тем, что на сегодняшний день данный тип атак является наиболее распространенным, а проблема DDoS атак не разрешима.

# 2.1.4 Разработка структурной модели построения правил для СОВ

Создание гибридной СОВ (рисунок 2.3), так же не решит всех задач, которые существуют в данной сфере, но применение подобного подхода к организации СОВ, все же, позволит сделать СОВ более эффективными в вопросах обнаружения вторжений. В первую очередь за счет постоянного увеличения базы правил по обнаружению тех или иных атак.

За основу возьмем сигнатурный метод обнаружения. Таким образом, в СОВ, при использовании данного метода, будет некоторая база сигнатур. Так-как большинство атак на сеть (это справедливо и для DDoS атак), это уже известные атаки, то эффективность использования сигнатурного метода очевидна. Однако при появлении нового воздействия на сеть, новой DDoS-атаки, например, будет использоваться анализ уязвимостей.



Рисунок 2.3 – Структурная модель формализованного метода выбора правил для СОВ

На представленном рисунке изображена структурная модель построения правил для СОВ. Она представляет собой объединение работы двух методов работы СОВ и использования некоторого генератора сигнатур. Данную модель в дальнейшем можно исследовать на представленной модели сети.

Решающее правило в данном случае работает следующим образом. Через Firewall (аппаратный брандмауэр) в LAN (локальная вычислительная сеть) и из нее идет поток информации, представленный в виде сетевого трафика. На брандмауэре вредоносный сетевой трафик отсеивается. Но если вредоносный трафик все же поступает в сеть (или иным образом влияет на работоспособность сети), то такой трафик подвергается обработке в СОВ. Стоит упомянуть, что и нормальный трафик тоже отслеживается СОВ, так-как заранее неизвестно, вредоносный трафик это или нет. Последовательность данных передаваемых по сети обрабатывается сетевыми протоколами, поэтому вредоносные данные можно в соответствии с правилами обработки описать по определенным признакам. Таким образом, если в последовательности данных в соответствии с сигнатурами присутствуют признаки атаки, то администратору сообщается об атаке на сеть. Если же признаков атаки нет, то решающее правило определяет данный трафик как нормальный, однако при этом так же проверяется реакция сети на данный трафик.

Если сигнатурным анализом мы определили атаку, то нет смысла проверять реакцию сети на атаку. Однако если производится атака, сигнатур для которой пока нет, анализируется реакция сети при помощи сканера уязвимостей (vulnerability Scanner). Если состояние сети по какому либо признаку, соответствующему некоторой атаке, ухудшилось, то на основе данного трафика создается сигнатура для обнаружения вторжения в дальнейшем для данного типа атаки, а сама СОВ сообщает администратору, что на сеть была проведена атака определенного типа. Кроме того, результатом решающего правила становится так же сведения в отчете о создании новой сигнатуры.

# 2.2 Разработка математической модели формализованной методики построения правил для СОВ

Как было сказано раньше, в основе данного метода используется набор сигнатур, сравнение с которыми и будет решающим правилом для СОВ. Отталкиваясь от этого можно определить правило для данного типа СОВ так:

|  |  |
| --- | --- |
| , | (2.2) |

где – очередное правило;

– действие, определяющее i-ое правило;

– множество сигнатур, определяющих i-ое правило.

На рисунке 2.3 мы видим несколько информационных потоков. Это информационные входные и выходные потоки x и x’, которые эквивалентны описываемому в формуле 6 множеству X. Поток, показывающий состояние сети Y. И поток Z предпринимаемых СОВ действий.

То есть у нас есть множество потоков X. От этого множества зависит выходной поток Z. Известно, что для исполнения инструкции необходимо, что бы поток соответствовал некоторой сигнатуре. Таким образом, выходной поток зависит так же от правил работы СОВ в момент времени t.

|  |  |
| --- | --- |
| , | (2.3) |

где – то – есть множество заданных правил проверки потока;

– очередной поток из множества сетевых потоков X.

При этом мы знаем, что правило по формуле 2.2 определено как множество сигнатур и выполняемое при совпадении действие. Но что если правило для данного потока не задано.

Пусть в некоторый момент времени t на вход поступил неизвестный информационный поток X. Для этого потока pi – это неизвестная величина. Следовательно, после проверки потока по правилам, которые настроены в СОВ, Z так же будет неизвестным.

Так как уже предусмотрено, что состояние сети можно проанализировать, то можно ввести некоторую функцию, которую назовем функцией создания правила (или множества правил).

|  |  |
| --- | --- |
| , | (2.4) |

где Y(t) – состояние сети после воздействия на него некоторого потока.

Тогда в момент времени t+1 к множеству правил P в момент времени t, будет добавлено новое множество правил:

|  |  |
| --- | --- |
| ,…,}, | (2.5) |

где m-n – количество правил, добавляемых к СОВ, а очередное правило вычисляется по формуле 8.

Следовательно, в момент времени t+1 множество правил p определяется следующей зависимостью.

|  |  |
| --- | --- |
| , | (2.6) |

При этом в момент времени t+1 система так же будет зависеть от функции определения правила.

|  |  |
| --- | --- |
|  | (2.7) |

Здесь мы уже говорим, что f1 – известная нам функция.

Обобщение опыта построения СОВ было сделано в документе «Единая структура систем обнаружения нарушителя». В данном документе определено множество компонентов, образующих СОВ:

* данные;
* модуль анализа;
* модуль хранения;
* модуль реакции.

Модуль сбора данных (ССД) предназначен для регистрации событий в системе и передачи их остальным модулям СОВ.

Модуль сбора данных является модулем, связывающим СОВ с внешним миром (без него СОВ не имеет информации о событиях в системе, относящихся к безопасности). Модуль анализа обрабатывает информацию от модуля сбора данных.

Большая часть исследований в области обнаружения нарушителя посвящена различным методам построения модуля анализа (МА).

Модуль сбора данных и модуль анализа обеспечивают систему большим количеством информации об активности системы. Данная информация может быть востребована и использована пользователями системы по истечении некоторого периода времени.

Хранением информации, сгенерированной во время работы СОВ, занимается модуль хранения (СХД). СОВ построены так, что в ответ на атаку они выдают сигнал тревоги. Многие системы в настоящее время имеют в своем составе модуль реакции (МПР). Данный модуль может по желанию пользователя СОВ автоматически запускает некоторый процесс, закрывает соединение TCP или модифицирует конфигурацию межсетевого экрана с целью предотвращения дальнейших атак.

На рисунке 2.4 изображена типовая схема СОВ. На ней представлены все компоненты и то, как они взаимодействуют.



Рисунок 2.4 – Типовая схема СОВ

Говоря об архитектуре СОВ, невозможно обойти вниманием вопрос о периодичности работы СОВ. Сканеры уязвимостей, обычно выявляющие уязвимости хоста или сети, выполняются периодически по запросу администратора системы. СОВ могут быть построены на основе периодического выполнения в пакетном режиме. Но современная практика показывает, что предпочтение отдается анализу нарушений в реальном режиме, что, естественно, налагает дополнительные требования на производительность этих средств обеспечения безопасности. Таким образом, представленная на рисунке модель СОВ должна работать в режиме реального времени. Подбор достаточного количества вычислительных ресурсов может обеспечить бесперебойную работу подобной СОВ.

Говоря о СОВ, необходимо отметить язык IDMEF [12], разработанный в качестве предполагаемого стандарта для обмена сообщениями между различными СОВ. В языке IDMEF с использованием языка XML описан формат сообщений, передаваемых между СОВ. Данный язык может решить проблему кроссплатформенности и взаимодействия между СОВ.

Отталкиваясь от типовой структурной схемы СОВ можно разработать структурную схему СОВ, в соответствии с которой будет работать формализованный метод выбора правил СОВ (рисунок 2.5). Эта схема более подробно описывает механизм работы формализованного метода выбора правил СОВ.



Рисунок 2.5 - Структурная схема СОВ с применением формализованного метода выбора правил СОВ.

На схеме блок МФСП – модуль формализованного построения правил (иными словами генератор сигнатур), БДмсп – БД данного модуля, МА – математический анализатор, МСП – модуль создания правила. При неизвестном p поток X попадает в БДсов, а затем в БДмсп, кроме того в БДмсп производится сбор данных о состоянии сети. После эти данные анализируются на математическом анализаторе. В данном случае МА выполняет роль функции . После чего на МСП происходит генерация правила для СОВ. Данное правило поступает в МПР, который заносит новое правило в систему хранения правил, и сообщает о результате выполнения созданного правила администратору сети.

Таким образом на выходе мы получаем по сути модель адаптивной СОВ. Обзор литературы [4,13,14,15,16,17,18,19,20,21,22,23] по данной теме показал, что все СОВ в той или иной степени являются адаптивными.

Адаптивность [24] – способность к адаптации, способность приспособится к изменяющимся условиям. Термин адаптации применим к любым системам, обладающим возможностью к изменению. В одном из определений, относящихся к фильтрации сигнала, под адаптивностью подразумевается автоматическое изменение параметров фильтров в зависимости от изменившихся условий. Термин же адаптивная система подразумевает под собой систему, автоматически изменяющую данные алгоритма своего функционирования и (иногда) свою структуру с целью сохранения или достижения оптимального состояния при изменении внешних условий.

Это значит, что любая современная СОВ может изменять свои правила. Однако, адаптивность, как свойство, не всегда является частью самой СОВ, и далеко не каждая СОВ изначально является адаптивной.

Адаптивность, как один из признаков функционирования СОВ описывается некоторой математикой, и, следовательно, должен находится под постоянным пристальным наблюдением со стороны администратора. В данном случае адаптивность СОВ описывается функцией f2, функцией выбора правил СОВ, а модуль формализованного построения правил – это и есть механизм адаптации представленной модели СОВ.

# 2.3 Выбор системы обнаружения вторжений.

Следующим шагом на пути решения задачи разработки формализованной методики построения правил СОВ является выбор самой СОВ, как основного блока контроля, применяемого при решении задачи, что само по себе является задачей, при том крайне не простой, так как в настоящее время на рынке представлено огромное множество решений в сфере обнаружения вторжений.

Ранее была приведена классификации СОВ по различным параметрам (эти параметры охватывали практически все основные аспекты данной тематики). Но в приведенной классификации не был учтен такой аспект, как особенности распространения (он был не обязателен). А по этому признаку все решения, представленные на рынке, делятся на две огромные категории: коммерческие и некоммерческие СОВ. Для того, что бы определиться с выбором СОВ, которую и будем использовать в дальнейшем, необходимо провести обзор и сравнительный анализ коммерческих и некоммерческих решений.

# 2.3.1 Коммерческие решения

В таблице 2.2 [25] представлены основные коммерческие решения на рынке СОВ. Рассмотрим некоторые из этих решений более подробно.

Таблица 2.2 – Основные коммерческие СОВ

|  |  |  |
| --- | --- | --- |
| Название  продукта | Производитель | Описание |
| 1 | 2 | 3 |
| Enterprise  Security for Communication and Collaboration | Trend Micro | Основной акцент сделан на защите от кибер-преступников службы e-mail, сервисов быстрых сообщений и систем для совместной работы сотрудников, партнеров и клиентов компании. |
| Apani EpiForce® | Apani Networks | Кросс-платформенное ПО для изоляции серверов, шифрования и контроля доступа, позволяющее разделять инфраструктуру на зоны с разным уровнем безопасности. |
| Cisco Catalyst 6500 Series Intrusion Detection System Module | Cisco Systems Inc | Решение для сложных угроз безопасности, таких как сетевые черви, отказ от обслуживания (DoS) и атаки на бизнес-приложения. |
| Digital Vaccine® Security Filter Service (Tipping Point NIPS) | Hewlett-Packard | ПО для фильтрации и анализа новых сетевых угроз (База уязвимостей обновляется еженедельно). |
| Reputation Digital Vaccine® Service | Hewlett-Packard | База уязвимостей этого ПО обновляется еженедельно для анализа новых сетевых угроз и их фильтрации. |
| Corero Top Layer IPS™ | Corero Network Security | Система предотвращения вторжений, которая предоставляет наиболее всестороннюю защиту сети, по сравнению с аналогичными продуктами. |
| fwsnort | CipherDyne | Это IDS/IPS на уровне приложения, которая работает на основе правил SNORT® и создаёт эквивалентные правила в брандмауэре iptables. |
| IDP Series Intrusion Detection and Prevention Appliances | Juniper Networks | В состав этого семейства продуктов входят IDS и IPS, которые предоставляют всестороннюю защиту сети от сетевых червей, троянских коней, шпионских программ, кейлоггеров и т. д. |

Продолжение таблицы 2.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| NetWitness NextGen™ | NetWitness | Система, которая причисляет себя к продуктам следующего поколения - знающая о всех проблемах безопасности и отвечающая на все возникающие проблемы в автоматическом режиме. |
| Secospace NIP Series Network Intrusion Detection System | Huawei Symantec | Интеллектуальна полиция сети (англ. Network Intelligent Police, NIP) - это IDS нового поколения, которая исследует сеть на основании сессий пользователей. |
| Strata Guard® | StillSecure | Высокопроизводительная мультисегментная IDS/IPS, которая пресекает все попытки хакеров взломать сеть, защищает от вредоносного ПО, шпионских модулей и сканирования портов. |
| UTM & Product Services | Clavister | Предоставляет высокий уровень защиты как физических сети, так и виртуальной, а также поддерживает большое количество дополнительных модулей безопасности. |
| Venusense Intrusion Detection and Management System (IDMS) | Beijing Venustech Inc. | Система обнаружения и управления вторжениями (англ. Intrusion Detection and Management System, IDMS), возможности которой шире, чем у обычной IDS. |
| Security Center (Network Management Monitoring Software) | Lan-Secure | Это ПО для безопасности сети, работающее как IDS и IPS в реальном времени, которое защищает от потенциальных угроз безопасности. |

Enterprise Security for Communication and Collaboration от Trend Micro - защищает почту, системы коллективной разработки и службы мгновенных сообщений Microsoft в режиме реального времени, предотвращая угрозы до того, как они нанесут ущерб. Уникальная архитектура "Интернет-ПК" Trend Micro Smart Protection Network обеспечивает защиту в режиме реального времени от разнообразных угроз (например, заражения электронной почты или ущерба веб-репутации), которая, действуя совместно с обычными технологиями безопасности контента, позволяют поддерживать связь без всякого риска.

Достоинства:

- механизмы упреждающей блокировки вредоносного контента;

- проверка почтовых респондентов по сервису репутации электронной почты;

- достаточно высокая скорость обработки трафика;

- гибкая система фильтрации контента;

- эффективные алгоритмы фильтрации спама;

-возможности централизованного контроля распределенной в пространстве системы;

- возможность создания взаимосвязей;

- стандартизация настройки правил;

- постоянно обновляемая сигнатурная база;

- поддержка различных платформ…

CISCO CATALYST 6500 SERIES INTRUSION DETECTION SYSTEM (IDSM-2) service module - это составная часть системы предотвращения вторжений Cisco Intrusion Prevention System. Работая во взаимодействии с другими элементами, этот модуль обеспечит эффективную защиту вашей инфраструктуры обработки данных. Учитывая все более изощренный характер угроз, которые могут подорвать безопасность, эффективные решения противодействия сетевыми вторжениями исключительно важны для поддержания высокого уровня надежности. Продуманная защита гарантирует работу без сбоев и сводит к минимуму негативные последствия вторжений.

Достоинства:

- решение интегрировано с аппаратной платформой, что дает выигрыш в производительности;

- возможность не только контроля, но и так же активного противодействия;

- в режиме IDS не снижает пропускную способность оборудования в независимости от загруженности сети;

- высокая надежность и отказоустойчивость;

- различные алгоритмы перехвата и контрмер;

- стандартизация настройки правил;

- интеграция сигнатур на базе партнерства CISCO и Trend Micro…

TippingPoint NIPS Digital Vaccine от Hewlett-Packard

Достоинства:

- еженедельно разрабатываются новые версии фильтров, исключая при этом атаку нулевого дня;

- гибрид фильтров работающих с сигнатурами и аномалиями сети;

- ПО обновляется за счет цифровых вакцин, которые предоставляются дважды в неделю;

- высокая степень масштабируемости при использовании ПО от HP;

- высокая отказоустойчивость;

- стандартизация настройки правил;

- поддержка различных платформ…

Corero NIPS

Достоинства:

- предоставляют возможности применения огромного множества алгоритмов обнаружения вторжений (пользователь при этом должен быть специалистом высшего уровня);

- прозрачно внедряемые в сеть устройства, невидимые для входящих потоков;

- возможности централизации системы управления для крупных внедрений;

- комплексные услуги по обновлению и эксплуатации базы угроз;

- кроссплатформенность;

- очень высокая скорость обработки трафика;

- реализован трехуровневый подход определения атаки (суть данного подхода чем то похожа на интегральный подход Аграновского);

- высокая отказоустойчивость;

- стандартизация настройки правил;

- поддержка различных платформ…

Анализ коммерческих СОВ показывает, что практически любое коммерческое решение предоставляет высокую степень защиты. Более того, коммерческие решения специализированны для решения определенного круга задач, что в итоге означает, что каждый коммерческий продукт будет защищать одни сервисы лучше чем другие, и, при этом, скорее всего, защита тех сервисов, на которые данный продукт специализирован, будет реализована лучше чем в других решениях конкурирующих компаний. Таким образом, на рынке складывается ситуация, когда конкурирующие продукты в принципе занимают различные ниши в решении задач обнаружения вторжений, и создают тем самым на класс определенных решений своеобразную монополию. Можно конечно использовать иной продукт, но для данного сервиса эффективность его будет ниже. Например, для провайдеров предпочтительней использовать, встроенные в коммутаторы и маршрутизаторы, СОВ CISCO CATALYST, в то время как для почтовых сервисов лучше использовать Enterprise Security for Communication and Collaboration. Таким образом, каждая СОВ занимает свою нишу, и, по сути, не имеет в ней конкурентов. Возможно именно поэтому цена на коммерческие СОВ такая высокая (а не только в виду специфичности самого производства СОВ).

Высокая цена коммерческих решений зачастую бывает непреодолимым барьером к их использованию в сфере мелкого и среднего бизнеса, где вероятность потерь и доходы не так высоки, что бы тратить десятки тысяч долларов на создание системы защиты с применением коммерческих СОВ. Именно поэтому и существуют некоммерческие решения.

# 2.3.1 Некоммерческие решения

Существование некоммерческих решений в первую очередь обусловлено тем фактом, что стоимость проприетарного программного обеспечения (ПО) очень высока. Но значит ли это, что некоммерческое ПО хуже решений частных. Для этого рассмотрим наиболее известная сетевые СОВ в классе некоммерческих решений:

- Snort NIDS;

- Bro NIDS;

- Prelude Hybrid IDS.

Первое, что бросается в глаза это количество решений, которые существуют бесплатно. Сфер деятельности, в которых существуют сети и применяются различные сервисы, куда гораздо больше чем 3. Более того, детальный анализ показал, что эти СОВ по функционалу во многом схожи. Чем это обусловлено?

Все это потому, что любая из этих СОВ не предназначена к использованию из коробки. Системный администратор конечно может использовать такую СОВ без предварительной настройки, но эффективность от нее будет крайне низкая. Здесь и проявляют себя коммерческие СОВ, в которых тонкая настройка хоть и нужна, выполнить ее гораздо легче. Кроме того, есть группа технической поддержки, готовая всегда помочь своему клиенту.

С некоммерческими решениями ситуация другая. Системный администратор сам настраивает СОВ под систему, которую он администрирует. А это требует достаточно высокого уровня знаний. Для наиболее полной настройки СОВ специалист по защите сетей должен знать язык правил используемой СОВ (благо дело существует такое понятие как стандартизация, что существенно облегчает изучение языков написания правил большинства СОВ), уметь программировать на таких языках как C/C++, понимать, как производятся атаки на сеть и быть в курсе наиболее часто используемых уязвимостей. Это далеко не все, что должен уметь специалист по защите сетей, перечислены наиболее важные аспекты. Само понятие некоммерческого ПО подразумевает использование бесплатно распространяемого ПО с открытым исходным кодом. То есть каждый человек, обладающий достаточным багажом знаний, может внести свою лепту в разработку данного продукта. Для каждого из них существует свое сообщество и официальный сайт разработчиков, на котором вы можете предложить свое нововведение, и если оно всем понравится, то в новой версии СОВ это нововведение будет включено. И вот здесь огромный плюс уже за некоммерческими решениями. Дело все в том, что некоммерческие решения используются десятками тысяч человек (так как бесплатные), а соответственно и нововведения появляются достаточно часто. Другое дело, что все эти вещи нужно уметь подключать отдельно самому. Кроме того, такая вовлеченность в процесс развития некоммерческого ПО привела к тому, что та же самая СОВ Snort, наиболее популярная среди некоммерческих продуктов, обладает ничуть не меньшей базой сигнатур, чем коммерческие СОВ.

Все это наводит на мысль о том, что использование некоммерческих решений оправдывает себя даже более чем, но при одном условии – к сотруднику, работающему с данным ПО необходимо предъявлять достаточно высокие требования (он должен хотя бы знать язык написания правил СОВ и уязвимости своей сети).

В итоге получается, что для реализации модели формализованного метода выбора правил для СОВ наиболее закономерным решением будет использование некоммерческих решений в виду следующих причин:

- некоммерческие решения бесплатны;

- развиваются не хуже, чем коммерческие;

- исходный код является открытым;

- возможности тонкой настройки под особенности используемых сервисов;

- большое количество различной документации и дополнительных модулей контроля, написанных энтузиастами.

Что касается непосредственно самой СОВ, то наиболее интересным является использование Snort в виду наиболее динамичного развития среди некоммерческих решений (к СОВ Snort написано наибольшее количество вспомогательной документации).

# 2.4 Выбор анализатора уязвимостей

Помимо СОВ в разрабатываемом решении необходимо использовать анализатор уязвимостей (АУ). Как и в случае с СОВ, рынок АУ весьма разнообразен. И как и в случае с СОВ, существуют некоммерческие решения с открытым исходным кодом ничем не уступающие коммерческим продуктам при должной настройке. Выбор между коммерческими и некоммерческими решениями очевиден, но для начало необходимо разобраться с тем, что такое АУ?

Анализаторы уязвимостей (или сканеры уязвимостей) [26] – это программные или аппаратные средства, служащие для осуществления диагностики и мониторинга сетевых компьютеров, позволяющее сканировать сети, компьютеры и приложения на предмет обнаружения возможных проблем в системе безопасности, оценивать и устранять уязвимости. Сканеры уязвимостей позволяют проверить различные приложения в системе на предмет наличия «дыр», которыми могут воспользоваться злоумышленники. Также могут быть использованы низкоуровневые средства, такие как сканер портов, для выявления и анализа возможных приложений и протоколов, выполняющихся в системе.

Работу сканера уязвимостей можно разбить на 4 шага:

1) сканер обнаруживает активные IP-адреса, открытые порты, запущенную операционную систему и приложения;

2) составляется отчёт о безопасности (необязательный шаг);

3) пытается определить уровень возможного вмешательства в операционную систему или приложения (этот шаг может повлечь сбой);

4) на заключительном этапе сканер может воспользоваться уязвимостью, вызвав сбой операционной системы или приложения.

Сканеры могут быть вредоносными или «дружественными». Последние обычно останавливаются в своих действиях на шаге 2 или 3. Среди сканеров уязвимостей можно выделить:

- сканер портов;

- сканеры, исследующие топологию компьютерной сети;

- сканеры, исследующие уязвимости сетевых сервисов;

- сетевые черви;

- CGI-сканеры ("дружественные" — помогают найти уязвимые скрипты).

Таким образом, сканер уязвимостей – это необходимый элемент для определения изменения состояния системы. Далее представлен список некоторых известных решения:

- Nessus: оценка уязвимостей под UNIX;

- GFI LANguard: коммерческий сканер сетевых уязвимостей под Windows;

- Retina: коммерческий сканер для оценки уязвимостей;

- Core Impact: автоматизированный продукт для тестирования несанкционированных проникновений в систему;

- ISS Internet Scanner: оценка уязвимостей на уровне приложений;

- X-scan: сканер для исследования сетевых уязвимостей…

На самом деле выбор очевиден – Nessus. Данное решение, по мнению экспертов, является одним из лучших сканеров уязвимостей. Есть так же версия и для Windows. Вот тот «небольшой» список преимуществ, которые получает пользователь, используя Nessus:

- много документации, позволяющей изучить все аспекты работы данного сканера;

- великолепная поддержка со стороны сообщества разработчиков, измеряемая выходом новых тестов ежедневно, если не ежечасно;

- высокая отказоустойчивость;

- более 2000 механизмов проверки уязвимостей;

- огромный перечень поддерживаемых платформ, на которых возможно проверить уязвимости, и возможностей проверки (проверка потайных входов, неликвидное использование CGI, проверка Cisco, выявление атак на доступность, ненадлежащее использование устаревших сервисов (например, finger), проверка FTP, общие проверки приложений, Netware, NIS, наличие встроенных сканеров портов, выявление удаленного доступа к файлам, RPC, возможность гибкой настройки, выявление проблем SMTP, SNMP, поддержка непроверенных и «бесполезных» сервисов, поддержка Windows и т.д.);

- клиент-серверная архитектура на основе встроенных модулей, что позволяет сделать АУ платформно-независимым и легко расширяемым;

- наличие встроенного языка описания сценариев атак (NASL);

- возможность взаимодействия с другим ПО (например, использование NMAP) и т.д.

И это далеко не весь перечень преимуществ данного сканера, при том, что данное средство, являясь одним из лучших всеми признанных продуктов, находится в категории свободно распространяющегося ПО.

Таким образом, в результате проделанной работы была разработан структурная и математическая модель формализованной методики построения правил для СОВ. Для реализации разработанной модели в качестве СОВ был выбран Snort, а сканером уязвимостей - Nessus.

# 3 Разработка программного обеспечения для проверки работоспособности разрабатываемой методики

# 3.1 Разработка алгоритмической модели формализованной методики построения правил для СОВ

Отталкиваясь от уже созданной структурной схемы СОВ и описав математику метода можно схематично изобразить алгоритм работы СОВ с описываемым методом (рисунок 3.1).

Алгоритм работы СОВ работает следующим образом. В начальный момент времени (t=1) на вход подается X = {x,x’} – множество входящих и исходящих потоков. Притом, СОВ будет работать со всеми потоками. Если информационные потоки в сети отсутствуют, то наиболее вероятной причиной будет не рабочее состояние сети, что в принципе равносильно состоянию, при котором программа не активна, так как в данный момент необходимость в ней отпадает. Кроме того стартовым параметром так же является все множество правил, которые уже заданы при включении программы. После включения программы начинается бесконечный цикл проверки информационного потока до тех пор пока пользователь не выключит СОВ.

Сразу же после включения СОВ начинает анализировать тот сетевой трафик, который в настоящий момент времени проходит через демилитаризованную зону в месте подключения СОВ. Данный момент времени для СОВ считается начальным, а тот трафик, который анализируется в этот начальный момент времени, это X(t) – множество входящих и исходящих информационных потоков в начальный момент времени.

Трафик, анализируемый СОВ, при каждом новом обороте цикла сверяется с сигнатурами которые есть в СОВ на данный момент времени (в начальный момент времени это те сигнатуры, которые уже заданы заранее до включения СОВ). Если трафик совпадает с сигнатурами, то считается, что на сеть производится атака. Так как атака распознана тем набором правил СОВ, который уже есть, то новых правил для распознавания строить не нужно.

Если трафик не совпадает с сигнатурами, то это может значить, что на сеть атака либо не производится, либо производится, но не определяется. Так как для нее еще нету сигнатур (то есть сценарий какой либо атаки был модифицирован). Тогда производится проверка состояния сети. Если оно не изменилось, то считаем, что атаки на сеть не производится, а значит и действия производить никакие не нужно. В противном случае, при изменении состоянии сети мы по признакам определяем атаку, которая наиболее соответствует подобному состоянию сети при успешном проведении. Затем ключевые данные, определяющие данный тип атаки, принимают значения данного информационного потока, считая эти значения сигнатурами новой атаки подобного типа.

В случае, если на сеть производится атака, то СОВ реагирует на атаку соответствующим образом (извещает администратора сети о возможной атаке на сеть, ведет учет всех потоков, связанных с атакой, или даже разрывает соединение).



Рисунок 3.1 – Алгоритмическая модель формализованного метода выбора правил для СОВ

После того как прошла проверка условий и было определено, какое действие должна совершить СОВ, необходимо обновить базу правил СОВ. В случае, если на сеть была проведена новая атака, то к правилам, которые уже есть в СОВ будет добавлено множество правил, определенных для данной атаки. Кроме того, если в данный момент времени входные данные были изменены, то на следующем шаге будут использованы уже измененный набор данных, будь то множество правил, сигнатуры, или положение, откуда производится съем сетевого трафика.

Проверка возможности изменения входных данных необходима, так как закрытие программы, это тоже параметр, проверяемый при работе (play). В случае если play принимает ложное значение, то завершается выполнение цикла, и программы в целом, так как смена значения play на ложное означает выключение программы пользователем.

# 3.2 Разработка алгоритма работы программного обеспечения

Во второй главе был произведен выбор непосредственно самих средств, при помощи которых будет апробирована разрабатываемая методика. Основная задача на данном этапе, составления алгоритма взаимодействия выбранного программного обеспечения (ПО). Схема алгоритма, представленная на рисунке 3.2 отличается от предыдущей тем, что она показывает реализацию метода на конкретном программном обеспечении (эта схема – уже непосредственно схема ПО).

Алгоритм взаимодействия выбранного ПО, представленный на схеме, работает следующим образом.

Сначала, данные (data), собранные фильтрами СОВ, проходят обработку предпроцессорами (preprocessors). Snort, имея модульную архитектуру, является легко расширяемым продуктом. Именно поэтому существует множество различных препроцессоров, обрабатывающих данные различных протоколов. Препроцессоры постоянно совершенствуются и обновляются, и как следствие, постоянно совершенствуется сам Snort. Так, например, последние версии препроцессоров, таких, как, Frag3, Stream5 [28] являются, по сути, препроцессорами обработки, основанными на движках работы с конкретными протоколами. Такой подход являет собой достаточно новую концепцию препроцессорной обработки данных различными движками в соответствии с протоколом, по которому данные обрабатываются.

Затем, данные (data’), обработанные препроцессором (отсеивается ненужная информация и данные выстаиваются таким образом, что бы их было удобно проверять), сравниваются с сигнатурами неликвидного и ликвидного поведения. Если данные совпали с сигнатурой, то, при необходимости, вызывается сигнал тревоги (alert).

Если данные представлены неизвестной последовательностью, то управление передается дополнительному модулю (add-on\_mod), который из набора сетевых данных производит отбор необходимых для запуска анализатора уязвимостей (АУ) параметров (например, адрес источника и приемника, порты, заголовок TCP…). Затем производит вызов сервера Nessus с заданными параметрами.

Сам Nessus в свою очередь производит необходимую проверку в соответствии с параметрами. Изначально Nessus не обладает возможностями запуска с определенными параметрами. Таким образом, вызов Nessus с определенными параметрами будет соответствовать вызову пользовательских функции сетевого анализатора Nessus.



Рисунок 3.2 – Схема алгоритма работы ПО

В соответствии с документацией [29], АУ Nessus обладает мощным встроенным C-подобным языком NASL. Данный язык позволяет писать пользователю пользовательские функции и модули, не вникая в алгоритмы работы самого АУ. Таким образом, вызывая Nessus, мы лишь запускаем проверку определенной машины или службы, что позволит сократить время проверки до минимума. В результате проверки Nessus определяет уязвимости по заданному адресу, возможные черные ходы.… То есть, Nessus формирует отчет о состоянии системы (именно той части, на которую было произведено воздействие).

Если состояние системы изменилось (например, появился черный ход), то система выдает сигнал тревоги, и передает управление дополнительному модулю Snort вместе с данными об изменении состояния. На основе данных об изменении состояния модуль формирует сигнатуру определенной атаки, наиболее всего подходящую по параметрам.

Однако, если состояние системы не изменилось, мы все равно можем создать сигнатуру. Правда, это уже будет сигнатура ликвидного поведения.

После этого, в независимости от того, была распознана атака, или нет, сведения о любой необычной активности записываются в лог файл (logfile), с которым может ознакомиться системный администратор. Кроме того в лог записывается подробная информация о анализе нового потока и само сформированное правило. Если СОВ не смогла сформулировать правило, то тогда в лог попадет информация о неизвестном потоке и результатах анализа этого потока, в соответствии с которым системный администратор, хорошо проанализировав ситуацию, может сам написать новое правило.

Далее проводится проверка на возможность изменения входных параметров (input\_change). То есть администратор в любой момент может изменить базу правил СОВ. Кроме того здесь же, на проверке ввода проверяется, не была ли программа отключена системным администратором.

Если была вызвана процедура завершения программы, то тогда цикл прекращает выполнение.

Таким образом, видно, что правильное конфигурирование данных двух средств поможет решить некоторые задачи, которые стоят обычно перед системными администраторами. Сама по себе методика изначально именно для того и предназначена, что бы упростить работу системному администратору. Тем не менее, не стоит забывать, что это программа, и человек, посвятивший себя ее изучению, сможет ее обмануть или заставить работать в свою пользу, именно поэтому системный администратор хоть и может воспользоваться данным ПО, все же обязан проверять действия, совершаемые программным обеспечением, реализующим описываемою методику.

# 3.3 Тестирование разрабатываемой методики

Теперь, когда формализованная методика построения правил для СОВ написана, и есть ее программная модель, необходимо провести тестирование. Но для начала необходимо определится с тем, как мы будем считать эффективность, каким средством мы будем пользоваться для генерации атак, описать процесс подготовки и тестирования и уже после показать получившийся результат.

# 3.3.1 Определение подхода к расчету эффективности от использования данной методики.

В соответствии со статьей Д.В. Ушакова, Н.Г. Милославской [35] существует два подхода к оценке эффективности: количественный и качественный. И это действительно так. Так например, решая проблему выбора СОВ и АУ, мы прибегли к использованию качественных характеристик оценки эффективности работы. Итак, как же произвести расчет эффективности.

Для вычисления эффективности оценивается зависимость вероятности обнаружения и вероятности ложного срабатывания от количества сигнатур.

Для вычисления вероятности ложного срабатывания используется следующая зависимость:

|  |  |
| --- | --- |
| , | (3.1) |

где X – определяется множеством таких параметров, как количество машин, метод обнаружения, настройка системы, объём и тип трафика.

Такую зависимость нельзя решить в общем виде, потому, что неизвестны коэффициенты параметров в данной зависимости и есть неопределенность в постановке самой задачи.

Ряд зарубежных институтов (в том числе и NIST)[36,37] предлагают использовать результаты опытных испытаний.

Так же в некоторых исследованиях предлагается использовать величину, называемую рабочей характеристикой анализатора, которая выражается зависимостью вероятности обнаружения от вероятности ложного срабатывания.

Многие коммерческие организации количественным предпочитают качественные характеристики, для оценки конкурентоспособности своего товара:

- возможности централизованного управления;

- создания собственных сигнатур;

- детального редактирования существующих;

- подробного анализа трафика…

Вывод: количественные характеристики дают достаточно точную оценку эффективности, однако труднореализуемы. Качественные же характеристики не столь точны, но их легко реализовать. Основной недостаток обеих групп методов в том, что точные характеристики можно получить лишь после опытных исследований на объекте.

Однако, можно предположить примерный количественный эффект от использования формализованной методики построения правил для СОВ. Это можно сделать на основании общеизвестных фактов и различных отчетов об исследованиях корпоративных сетей крупными компаниями-производителями защитного ПО. Так например, известным является тот факт, что большинство системных администраторов предпочитают минимизировать количество ложных срабатываний за счет снижения ограничений, используемых при проверке. Это ведет к увеличению вероятности пропуска вторжения, но системные администраторы сознательно идут на такой риск, так как зачастую проблема ложных срабатываний негативно сказывается на финансовом благополучии компании.

Используя числа из отчетов различных организаций попробуем произвести примерный расчет количественного эффекта.

По данным компаний QRator и Trend Micro [38,39] известно, что в среднем корпоративные сервера подвергаются 9-10 атакам в день. По данным той же компании, вредоносный трафик составляет примерно 10 процентов всего трафика. Следовательно, за год, в среднем, система защиты должна срабатывать, в идеале, около 3500 раз, тогда как остальные заявки и обращения (около 32 тысяч на один сервер) система должна пропускать. Как же оно на самом деле.

К сожалению данных на количество ложных срабатываний найти не удалось. Однако известно, что хорошо настроенные сигнатурные системы обнаружения вторжений [5,23,28] дают возможность обнаруживать все известные атаки, и как пишут разработчики Snort [28], при максимальной минимизации ложных срабатываний хорошо сконфигурированный Snort может распознать практически каждой известное вторжение. Таким образом, можно сделать следующее заключение: для сигнатурных СОВ процент ложных срабатываний стремится к 0, а процент обнаруженных атак – к 100 процентам. Однако данное справедливо лишь для известных атак. А что делать с атаками, для которых нет сигнатур. К сожалению Snort не распознает такую атаку. Следовательно процент обнаружения новых атак равен нулю. Осталось лишь узнать, а каков процент новых атак в общей массе всех атак, производящихся на сервера.

Данные, представляемые в отчетах таких компаний, как Kaspersky Lab, Symantec, Corero [40,41,42] показывают, что количество образцов нового вредоносного кода ежегодно растет, но ведь вместе с тем и растет количество сигнатур, обнаруживающих этот код. Приблизительный подсчет показывает, что ежегодное соотношения числа сигнатур к числу новых разновидностей вредоносного кода составляет примерно 10%. То есть из 100 атак 10 будут новыми.

Таким образом, сигнатурная СОВ распознает 90% всех атак на сервер, при практически 0% ложных срабатываний. Однако не стоит забывать, что данное заявление справедливо лишь для хорошо сконфигурированной СОВ, которая учитывает работу всех внутренних сервисов и правила которой настроены в соответствии с уязвимостями, свойственными именно для той сети, в которой СОВ установлена. Например, Snort «из коробки» сконфигурирован так, что, скорее всего, распознает лишь порядка 20% атак на сеть. Для доведения настройки СОВ до оптимальной необходимо исследовать сеть. Такое исследование системным администратором сети составляет порядка 2-3 месяцев [5,28].

Что касается систем обнаружения аномалий, известно лишь, что они, в отличие от сигнатурных СОВ, могут обнаруживать известные и неизвестные атаки с равным успехом, и то, что в общем случае, эффективность систем распознавания аномалий ниже. Дело в том, что разработки данных систем до сих пор ведутся, и пока не достигли таких значительных результатов обнаружения, как сигнатурные методы обнаружения для известных атак.

В нашем случае используется метод сравнения с эталоном. Этот метод не является лучшим по эффективности среди методов обнаружения аномалий. Однако он достаточно прост в реализации. Разные авторы сулят ему различную эффективность, однако из различных источников известно, что, как правило, эффективности классическому сравнению с эталоном (здесь подразумевается система) не дают оценку выше 60%. Относительно ложных срабатываний, эту цифру вообще предположить нельзя.

А теперь снова обратимся к алгоритму, который разработали. Модель СОВ, реализующая методику, представляет собой гибрид. В основе лежит сигнатурное обнаружение. Следовательно эффективность должна составлять порядка 90%, однако для неизвестного потока применяется сравнение с эталоном, которое из 10% оставшихся атак сможет распознать около 60% (то есть еще 6% атак распознает полученный гибрид). Не сложно подсчитать, что общая эффективность должна составлять порядка 96% для обнаружения атак. Цифру для ложного срабатывания, к сожалению, в теории подсчитать не удастся. Наглядно данные можно посмотреть в таблице 3.1.

Таблица 3.1 – Сравнение эффективности сигнатурного, аномального и гибридного методов по среднестатистическим данным за год

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Обнаруживаемые вторжения | Ошибка первого рода (ложные срабатывания) | Ошибка второго рода (пропуск вторжений) | Обнаружено новых атак |
| Сигнатурный | 3120 (≈89%) | 288 (≤1%) | 380 (≈11%) | 0 |
| Аномальной активности | 2100 (≈60%) | - | 1400 (≈40%) | 210 (≈60%) |
| Гибридный | 3325 (≈95%) | - | 170 (≈5%) | 210 (≈60%) |

В таблице для аномального и гибридного методов ошибка первого рода не указанна, так как данных статистики нет. Однако, она вероятнее всего, много больше 1%.

Что касаемо времени настройки, которое мы получим в результате применения формализованной методики построения правил СОВ, подсчитать не составит труда. Известно, что в среднем, на просмотр одной записи о нарушении безопасности и построения правил системный администратор может потратить порядка 2 часов.

Однако в случае автоматического построения правил, системному администратору нужно будет лишь посмотреть верно ли данное правило или нет. На это может потребоваться от 10 до 20 минут. Таким образом, мы получаем выигрыш в скорости в 7-8 раз.

На рисунке 3.3. изображен график роста эффективности работы СОВ во времени.

Рисунок 3.3 – Сравнение скорости роста эффективности Snort с применением ручной настройки и с использованием описываемого метода

В ходе написания дипломной работы на персональном компьютере был установлен и постепенно настраивался Snort в соответствии с рекомендациями разработчиков. До конца все настройки были реализованы лишь за 2 месяца. Предположительно применение методики ускорит данный процесс в 7-8 раз.

# 3.3.2 Описание генератора атак

Для проведения эксперимента необходимо произвести выбор генератора атак. Достаточно интересным представляется генератор атак LOIC [27] - (сокр. от Low Orbit Ion Cannon (англ. Низкоорбитальная ионная пушка), отсылка к играм серии Command & Conquer) — программа с открытым исходным кодом для осуществления сетевых DoS-атак, написанная на языке программирования C#. Первоначально разработана Praetox Technologies, но позже была распространена как общественное достояние. У данной программы есть форк.

GAS (сокр. от Ground-based Antisatellite System) — форк LOIC, изначально содержавший код из разных версий, но на данный момент отличие кода от других версий составляет более 50 %. Поддерживает около 9 видов атак. Работает эффективнее оригинальной версии, имеет консольную версию, что позволяет атаковать с сервера. Разрабатывается достаточно активно. Так же можно запустить программу на Linux, имея установленный Mono. Базируется на клиентском профиле .NET 4.0. На данный момент вид GAS не сильно отличается от LOIC, но со слов автора планируется новый интерфейс и поддержка «Hive Mind».

И так, в качестве генератора атак будем использовать GAS (рисунок 3.4)

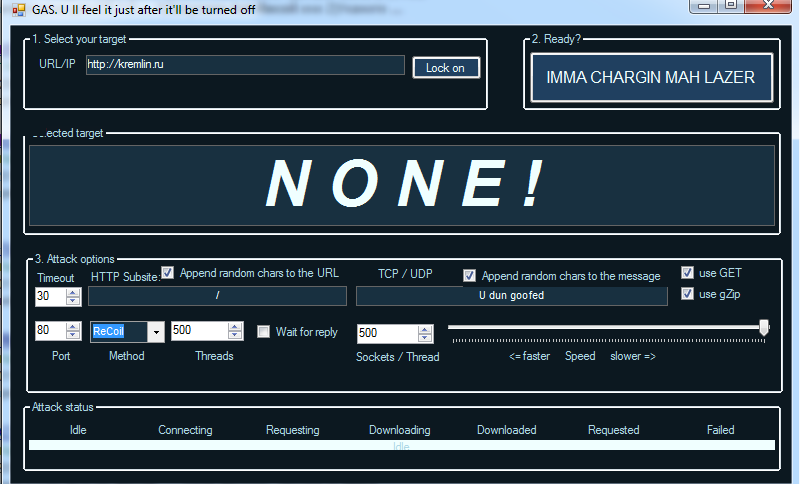


Рисунок 3.4 – Интерфейс генератора DDoS-атак GAS

На рисунке отображен метод атаки ReCoil. Для атаки на ПК это не самый эффективный способ. Намного более эффективными в данном случае будут атаки типа Smurf и Fraggle [43].

Атака Smurf или ICMP-флуд — одна из самых опасных видов DoS-атак, так как у компьютера — жертвы после такой атаки произойдет отказ в обслуживании практически со 100 % гарантией. Злоумышленник использует широковещательную рассылку для проверки работающих узлов в системе отправляя ping-запрос. Очевидно, атакующий в одиночку не сможет вывести из строя компьютер-жертву, поэтому требуется еще один участник-это усиливающая сеть. В ней по широковещательному адресу злоумышленник отправляет поддельный ICMP пакет. Затем адрес атакующего меняется на адрес жертвы. Все узлы пришлют ей ответ на ping-запрос. Поэтому ICMP-пакет, отправленный злоумышленником через усиливающую сеть, содержащую 500 узлов, будет усилен в 500 раз. Поэтому для такой атаки обычно выбирается большая сеть, чтобы у компьютер-жертвы не было никаких шансов.

Атака Fraggle (от англ. Fraggle attaсk) есть полный аналог атаки Smurf, только вместо ICMP пакетов используются пакеты UDP, поэтому ее еще называют UDP-флуд. Принцип действия этой атаки простой: на седьмой порт жертвы отправляются echo-команды по широковещательному запросу. Затем подменяется ip-адрес злоумышленника на ip-адрес жертвы, которая вскоре получает множество ответных сообщений. ИХ количество зависит от числа узлов в сети. Эта атака приводит к насыщению полосы пропускания и полному отказу в обслуживании жертвы. Если все же служба echo отключена, то будут сгенерированы ICMP-сообщения, что так же приведет к насыщению полосы.

Кроме того, некоторые типы атак можно регулировать. Например, в случае с TCP-флудом, можно настроить номер порта.

Кроме того можно регулировать интенсивность атаки. Чем более интенсивная атака, тем больше задействует ваш процессор ресурсов. Максимальная скорость подразумевает, что ваш процессор будет генерировать пакеты с наибольшей скоростью.

# 3.3.3 Подготовка и тестирование

Последним шагом перед апробацией метода является подготовка ПО. Нам необходимо установить Snort и Nessus [28,29]. Модули взаимодействия данных двух проектов нужно писать на языках C и NASL [28,44,45].

Так как оба проекта является открытыми, то можно спокойно самому собрать эти проекты под управлением своей операционной системы (ОС). И вот здесь встает один из главных вопросов. А какую же ОС лучше использовать.

Дело в том, что Linux является свободно распространяемой ОС. Поэтому существует огромное число проектов, связанных с данной ОС. В повседневной жизни обычные пользователи привыкли к использованию ОС Windows. Но все мы забываем одну важную вещь. Windows является проприетарным программным обеспечением. Это значит, что исходный код Windows закрыт.

В отличие от Window – Linux имеет открытый исходный код, что дает возможность разработчикам сконфигурировать настройки самой ОС так, как им удобно. Именно поэтому многие профессиональные разработчики в повседневной работе обычно используют Linux.

Конечно много ПО пишется и под Windows. Но это в основном прикладное офисное или домашнее ПО. В то время как под Linux пишут с целью создания крупных проектов. Большинство серверов стоят под управлением Unix-подобных систем. Следовательно, в основном все серверное ПО изначально подразумевается под Linux. Кроме того, много научной литературы, связанной с информационными технологиями в примерах и описании используют именно Linux. Таким образом необходимо при написании ПО и установке использовать данную ОС (кроме того, вероятность того, что разрабатываемый продукт будет в дальнейшем использоваться под управлением Linux наибольшая из возможных).

Для написания ПО в Linux используется компилятор gcc. Кроме компилятора нам необходим некоторый инструментарий.

Для написания ПО необходим текстовый редактор. В мире Unix-систем есть два лагеря: любители Vi и любители Emacs. В настоящее время написано много клонов редактора Vi (Vim, Elvis, GVim) и много клонов (а точнее - расширений) редактора Emacs. И для Vi и для Emacs существуют как консольные так и графические версии. Оба этих редактора обладают огромными возможностями. Мы будем использовать Emacs, так как у него есть графический интерфейс, связанный с графическим интерфейсом Linux (здесь стоит обратить внимание на то, что в отличие от Windows, Linux не подразумевает графического интерфейса, кроме того, из консоли в Linux работать намного удобнее).

Так же необходимо уточнить командную оболочку. Bash, Csh, Ksh - это все командные оболочки или, как их еще называют, командные интерпретаторы. Командная оболочка - это не просто исполнитель команд, но и полноценный интерпретатор собственного языка программирования. Чтобы узнать, какая оболочка запущена под терминальной программой, просто наберите команду ps. В нашем случае используется командная оболочка bash.

Snort устанавливается довольно просто. В качестве предварительного условия требуется установить пакет libpcap.

Его можно загрузить по адресу [http://www.tcpdump.org](http://www.tcpdump.org/). При установке библиотеки Libcap возможно появится необходимость установки некоторых библиотек, от которых зависит работа Libcap. Все зависимости данной библиотеки, порядок установки и адрес для обращения в случае возникновения проблем указаны в файле Instal.После загрузки этих библиотек необходимо загрузите Snort с web-сайта разработчиков.

Когда файл окажется в вашей машине, распакуйте его и выполните команды компиляции:

./configure

make

make install

Snort возможно запустить в нескольких режимах работы, однако нас интересует режим обнаружения вторжений, при котором Snort протоколирует подозрительные или требующие дополнительного внимания пакеты. Для перевода Snort в режим обнаружения вторжений необходимо в кончоле прописать следующую инструкцию:

<snort -de -l /var/log/snort -h 192.168.1.0/24 -c ../конфигурационный\_файл>

Данная инструкция предписывает использовать указанный конфигурационный файл для управления протоколированием пакетов. Конфигурационный файл определяет все настройки Snort, он очень важен (приложение A). Snort поставляется с подразумеваемым конфигурационным файлом, но перед запуском в него целесообразно внести некоторые изменения, отражающие специфику вашей среды.

Установка Nessus для систем Linux

Имеется два необходимых условия, которые должны быть выполнены перед установкой Nessus, и два других, которые желательно выполнить заранее, чтобы в полной мере воспользоваться дополнительными возможностями.

Необходимым условием является установка двух программных продуктов - Gimp Tool Kit (GTK) и libpcap. Можно загрузить GTK по адресу <ftp://ftp.gimp.org/pub/gtk/v1.2>

libpcap уже был установлен при установке Snort.

Двумя необязательными, но желательными программами являются OpenSSL и Nmap. Nessus может использовать Nmap в качестве сканера портов, а OpenSSL - для безопасных коммуникаций между сервером и клиентом.

Установка GTK так же как и Libcap может показаться сложной, из-за громадного числе зависимостей, необходимых её для работы. Так же как и с библиотекой libcap, важную информацию можно найти в файле Install.

После этого необходимо загрузить Nessus с сайта производителя. После распаковки бинарного файла запускаем Nessus командой Nessus start. Данная команда запускает Nessus Server. После этого необходимо при помощи браузера подсоединиться к порту 8834.

После этого начнется SSL-сессия. По окончанию который вы должны уже создать пользователя (с административными полномочиями) и подтвердить регистрацию. Для этого нужно зарегистрироваться на сайте производителя Nessus [29]. Тоже самое и для Snort. Что бы получить доступ к базе правил, необходимо зарегистрироваться на сайте разработчиков [28].

# 3.3.4 Результаты эксперимента

После установки всех пакетов и написания ПО можно запускать данный комплект ПО для тестирования.

Данная система не была протестирована полностью, но на ограниченно выборке показала свою эффективность, хоть и не полностью.

На рисунке 3.5 отображен скриншот загруженности процессора до использования комплекта данного ПО и во время использования.

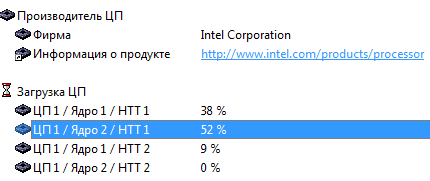
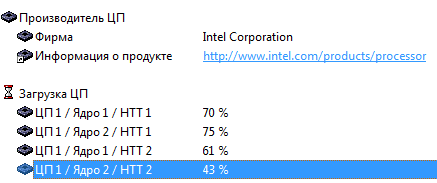


Рисунок 3.5 – Загруженность ЦП до использования комплекса ПО и во время использования

Как видно из рисунка, загруженность ЦП изменилась, в связи с тем, что производился отброс некоторого количества запросов при использовании СОВ.

На 10 атак, которые не были заложены в правила СОВ, 4 атаки на отказ в обслуживании были обнаружены. Следовательно, порядка 40% новых атак система может обнаруживать по факту. В таблице 3.2 приведены результаты тестирования данного комплекта ПО.

Таблица 3.2 – Сравнение результатов работы комплекса используемого ПО и СОВ Snort

|  |  |  |  |
| --- | --- | --- | --- |
| СОВ | Обнаруживаемые вторжения | Обнаружено новых атак | Ошибка второго рода (пропуск вторжений) |
| Snort | 10 (≈50%) | 0 | 10 (≈50%) |
| Комплекс разработанного ПО | 14 (≈70%) | 4 (≈40%) | 6 (≈30%) |

Так же стоит упомянуть, что данное ПО разрабатывалось лишь с целью проверки методики и в виду нехватки времени, не было должным образом протестировано и не может распознать все виды атак, которые возможны на сеть. Поэтому применение в реальных условиях не желательно.

Важно сказать так же, что общее число атак на ПК, выступавшего в роли сервера, было равно 20, чего не совсем достаточно для хорошей статистики. Но даже для 20 атак при использовании формализованной методики построения правил СОВ было выявлено на 4 атаки больше, что свидетельствует о некотором успехе данного метода.

# 3.4 Рекомендации по работе с программным обеспечением системному администратору

Процесс установки был ранее уже описан. Важно отметить некоторые особенности работы комплекса данного ПО.

Во первых, что касается сканера уязвимостей Nessus [5,29], стоит придерживаться следующих правил:

1) нельзя сканировать сеть, не находящуюся под вашим контролем, не имея разрешение на сканирование, т.к. некоторые из видов активности, инициированной Nessus, могут юридически рассматриваться хакерским взломом (особенно при включенной опции атак на доступность).

2) необходимо перед сканирование делать резервную копию (категорически не рекомендуется запускать сканирование в момент создания резервной копии данный, потому, что это не только может привести к порче данных резервной копии, но и существенно замедлит оба процесса);

3) необходимо спланировать время проведения сканирования для получения требуемых результатов с минимальным влиянием на других служащих и скоординировать эти действия с персоналом (сообщить заранее о том, что сеть будет подвержена сканированию, что приведет к сильной загруженности сети);

4) рекомендуется выполнять сканирование постоянно включенных серверов в нерабочее время и избегать наложения с другими операциями по администрированию системы и повседневной деятельностью;

5) не рекомендуется проводить сканирование сети так часто, что затруднительно будет интерпретировать данные в отчетах (из-за их количества и скорости появления новых отчетов);

6) оптимальный интервал между двумя полными сканированиями от 1 недели до 1 месяца;

7) для проверки уязвимости информационной системы извне (из Интернета) следует разместить сервер Nessus вне установленного внешнего межсетевого экрана (при этом необходимо включить поддержку SSL для криптографической защиты коммуникаций между клиентом и сервером);

8) при сканировании внутренней сети сервер необходимо разместить позади межсетевого экрана;

9) установка Nessus на ПК-блокноте может облегчить выполнение сканирования как изнутри, так и извне вашей сети, не требуя множества машин;

10) необходимо учитывать, что выполнение некоторых проверок, по сути, являются атакой на сеть, что приведет к конфликту с СОВ и неработоспособности используемого ПО (возможно, даже к краху операционной системы);

Настроить запуск тестов можно в соответствующем конфигурационном файле (приложение Б). Там же можно дописывать новые процедуры проверки.

Решая, где разместить сетевую систему обнаружения вторжений [5,23,28], следует принять во внимание, что необходимо защитить и как можно максимизировать эффективность и взаимную поддержку средств сетевой безопасности. Имеется несколько вариантов размещения сетевой системы обнаружения вторжений, у каждого из которых есть свои достоинства и недостатки:

1) в ЛВС позади межсетевого экрана – это наиболее распространенная конфигурация, которая предлагает наилучшую защиту как от внешних, так и от внутренних угроз. Прослушивая локальную среду передачи, можно выявлять внутреннюю активность пользователей, такую как взаимодействие между рабочими станциями или ненадлежащее применение программ. Это также обеспечивает дополнительную поддержку межсетевого экрана, позволяя обнаружить подозрительный трафик, каким-то образом сумевший проникнуть во внутреннюю сеть через фильтры экрана. В действительности, систему обнаружения вторжений можно применять для тестирования межсетевого экрана, чтобы увидеть, какой трафик он пропускает. При этом стоит учесть то, что будет генерироваться много сигналов тревоги на основе потоков данных Windows, так что необходимо будет проделать большой объем работы по настройке в этой области. Далее, если локальная вычислительная сеть (ЛВС) является коммутируемой, то понадобится возможность отражения всех портов в порт монитора, чтобы система обнаружения вторжений могла прослушивать весь трафик ЛВС;

2) в демилитаризованной зоне - можно поместить сенсор Snort в демилитаризованной зоне, чтобы отслеживать активность по отношению к общедоступным серверам. Так как эти серверы наиболее открыты в организации и обычно представляют собой ценные ресурсы, то весьма разумно наблюдать за ними с помощью системы обнаружения вторжений. Проблема, которая возникает при подобной конфигурации, состоит в сортировке всех сигналов. Хотя все они могут быть оправданными сигналами тревоги, в наше время общий уровень атакующего трафика в Интернет таков, что любой общедоступный IP-адрес по несколько раз в день подвергается случайным атакам. Реагирование и попытки отследить эти сигналы будут излишними и не продуктивными. Для эффективного обнаружения именно целевой атаки можно сократить число сигнатур до небольшой величины, чтобы срабатывания происходили, только если компьютер действительно был скомпрометирован. Примером могут служить специальные правила для приложений, выполняющихся на этом компьютере, такие как правила для MySQL или web-iis, или правила, связанные с административным входом в систему. Можно исключить большинство сигналов зондирующего характера и не реагировать на такую деятельность, как сканирование портов и т.д.

3) между поставщиком Интернет-услуг и межсетевым экраном – в этом случае будет фильтроваться весь входящий и исходящий трафик ЛВС и демилитаризованной зоны. Положительная сторона этого подхода состоит в том, что будет перехватываться все, что направлено против общедоступных серверов и внутренней ЛВС, отрицательная - в том, что нет возможности увидеть внутренний трафик, а общий объем сигналов может быть весьма большим из-за высокого уровня фонового атакующего трафика. Как и в предыдущем примере, можно ограничить набор сигналов, оставив включенными только те, которые действительно будут отражать проблему для данного сетевого сегмента. Следует учитывать также, что сенсор, размещенный в канале между поставщиком Интернет-услуг и межсетевым экраном, может стать точкой отказа для сетевого трафика. Возможное решение состоит в установке небольшого концентратора между двумя каналами и в подключении системы обнаружения вторжений к нему.

Это 3 наиболее разумных варианта размещения систем обнаружения вторжений. Разумеется, ничто не мешает использовать их все, если при этом компания располагает достаточными мощностями для вычисления временем для управления.

Важно соблюдать приведенные рекомендации при работе со средствами защиты сети, потому, что не соблюдение этих рекомендаций может привести к отказу в работе сети, операционной системы или программного обеспечения, что так же, как и в случае реализации атаки, может принести значительный ущерб организации и потерю средств от использования систем защиты

# 4 Безопасность труда

Задача, решаемая в дипломной работе, на стадии своего проектирования и использования предполагает применение компьютера, состоящего из монитора и системного блока и питающегося от переменного напряжения. Возможно использование стенда сетевого оборудования, а также осветительных приборов помещения, питающихся от переменного напряжение.

Вышеперечисленные технические средства могут отрицательно воздействовать на здоровье человека. Также здоровье человека зависит от таких факторов, как микроклимат помещения, его освещенность, шум и запыленность. Именно поэтому важно произвести расчет условий труда. Выполнение данной задачи напрямую связано с такой научной категории, как безопасность жизнедеятельности человека.

Безопасность жизнедеятельности (БЖД) - это комплекс мероприятий, направлен­ных на обеспечение безопасности человека в среде обитания, сохранение его здо­ровья, раз­работку методов и средств защиты путем снижения влияния вредных и опас­ных фак­торов до допустимых значений, выработку мер по ограничению ущерба в лик­видации по­следствий чрезвычайных ситуаций мирного и военного времени.

Объектом, рассматриваемым в работе, является учебная компьютерная аудитория №14137. Расположена на первом этаже 14 учебного корпуса ГОУ «ОГУ» по адресу: г. Оренбург, Шарлыкское шоссе, д. 5.

# 4.1 Анализ и обеспечение безопасных условий труда

С точки зрения поражения электрическим током данное помещение с ЭВМ относится к классу помещений без повышенной опасности. Температура окружающей среды варьируется от 20 до 25° С, значения относительной влажность воздуха так же варьируются в диапазоне от 40 до 70%.

В санитарных нормах СН-245-71 установлены величины параметров микроклимата, создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения. (см. табл. 5.1).

Таблица 4.1 – Параметры микроклимата для помещений, где установлены компьютеры

|  |  |  |
| --- | --- | --- |
| Период года | Параметр микроклимата | Величина |
| Холодный | Температура воздуха в помещении Относительная влажность  Скорость движения воздуха | 22…24°С  40…60%  до 0,1м/с |
| Теплый | Температура воздуха в помещении Относительная влажность  Скорость движения воздуха | 23…25°С  40…60%  0,1…0,2м/с |

Из таблицы видно, что параметры микроклимата в рассматриваемом помещении практически соответствуют нормам.

В рассматриваемом помещении непроводящие полы, отсутствует токопроводящая пыль, отсутствует электрически активная среда, отсутствует возможность одновременного прикосновения к металлическим частям прибора и заземляющему устройству (ПУЭ 1.1.13). Для защиты от поражения электрическим током все токоведущие части должны быть защищены от случайных прикосновений кожухами (ПУЭ 1.1.32), корпус системного блока ЭВМ должен быть заземлен. Заземление выполняется изолированным медным проводом сечением 1.5 мм2 (ПУЭ 1.7.78), который присоединяется к общей шине заземления с общим сечением 48 мм2 при помощи сварки. Питание устройства должно осуществляться от силового щита через автоматический предохранитель, срабатывающий при коротком замыкании нагрузки.

Помещение оснащено аптечкой первой помощи и углекислотным огнетушителем ОУ-3. Основными причинами поражения электрическим током являются: пробой изоляции, короткое замыкание, несоблюдение правил техники безопасности. Для предотвращения чрезвычайных ситуаций применены следующие технические способы и средства защиты:

* регулярные инструктажи сотрудников и студентов;
* зануление;
* защитное отключение.

Эксплуатация устройства должна производиться персоналом, имеющим квалификацию по ТБ III (согласно ПТЭ). Работа по устранению неисправностей и наладка должна производиться персоналом с квалификационной группой по ТБ не ниже III (согласно ПТЭ) и только после снятия напряжения питания с устройства. Снятие защитного кожуха с компьютера и какие-либо другие конструктивные изменения компьютерной системы пользователю строго запрещены. Это должны делать только квалифицированные специалисты. Допуск к наладке, ремонту компьютерной системы имеет инженер-электронщик не ниже 11-го разряда.

Согласно требованиям СанПиН (Санитарные правила и нормы СанПиН 2.2.2.542-96 "Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы", утвержденные постановлением Госкомсанэпиднадзора РФ от 14 июля 1996 г. N 14) площадь на одно рабочее место в помещениях для эксплуатации ВДТ и ПЭВМ для взрослых пользователей должна составлять не менее 6 м2., а объем не менее 20 м3 . В исследуемом помещении находится четыре рабочих места, три из которых оснащены компьютерами. Таким образом, площадь на одно рабочее место составляет 6,06 м2, а объем 19,4 м3.

Для защиты от избыточной яр­кости окон применены шторы [46]. Для внутренней отделки интерьера помещений с ВДТ и ПЭВМ должны использоваться материалы со следующими значениями коэффициентов отражения: для потолка – 70% - 80%, для стен – 50% - 60%, для пола – 30% - 50%. Цветовой фон соответствует расположению окон (ориентированы на запад). Цвет потолка белый, цвет стен – желтый, цвет пола – светло-коричневый, что не раздражает глаза и соответствует санитарным нормам.

Поверхность пола рассматриваемого помещения покрыта линолеумом, что удобно для влажной уборки и не позволяет пыли скапливаться.

Нормирование естественного и искусственного освещения осуществляется в зависимости от характера выполняемых работ. По характеру зрительных работ эксплуатация компьютерной системы относится к 4-му разряду (освещённость 400 лк). Естественное освещение должно осуществляться через светопроемы, ориенти­рованные преимущественно на север и севе­ро-восток и обеспечивать коэффициент есте­ственной освещенности не ниже 1,5%. Для помещения, в котором находится рабочее место, коэффициент естественной освещенности равен 1,9%.

Входящие в состав компьютерной системы персональный компьютер и монитор могут являться источниками электромагнитных излучений, что особенно актуально для старых моделей системных блоков и ЭЛТ-мониторов.

В рассматриваемом помещении два из трех системных блоков являются устаревшими и не являются безопасными источниками электромагнитных излучений. Каждое рабочее место оборудовано мониторов на основе электро-лучевой трубки, ни один из мониторов не имеет защитной сетки, два из трех монитора расположены к стене ближе, чем один метр, не отвечая тем самым требованиям СанПиН 2.2.2 542-96 «Гигиенические требования к видео-дисплейным терминалам, персональным электронно-вычислительным машинам и организации работы».

Шум ухудшает условия труда оказывая вредное действие на организм человека. Ра­бо­тающие в условиях длительного шумового воздействия испытывают раздражитель­ность, головные боли, головокружение, снижение памяти, повышенную утомляе­мость, понижение аппетита, боли в ушах и т. д. Такие нарушения в работе ряда орга­нов и сис­тем организма человека могут вызвать негативные изменения в эмоциональ­ном состоя­нии человека вплоть до стрессовых. Под воздействием шума снижается концен­трация внимания, нарушаются физиологические функции, по­является уста­лость в связи с повы­шенными энергетическими затратами и нервно-психическим на­пряжением, ухуд­шается речевая коммутация. Все это снижает работоспособность че­ловека и его производитель­ность, качество и безопасность труда. Длительное воздей­ствие интенсивного шума [выше 80 дБ(А)] на слух человека приво­дит к его частичной или полной потере.

Административное здание, в котором находится рассматриваемое помещение, расположено далеко от проезжей части, поэтому шума от автотранспорта нет. Источником шума в кабинете является лишь работа охлаждающих устройств системных блоков, работа лазерного принтера (во время печати документов) и разговор самих работников. Таким образом, уровень шума находится в пределах 40 дБ [48,49], следовательно, данный фактор не оказывает отрицательного воздействия на здоровье человека.

Модульный размер рабочей поверхности стола для ПЭВМ, составляет: ширина 800-1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм. Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм. Конструкция рабочего стула должна обеспечивать ширину и глубину поверхности сиденья не менее 400 мм, поверхность сиденья с закругленным передним краем, регулировку высоты поверхности сиденья, стационарные или съемные подлокотники. Данные нормативные требование выполнено и соответствует СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».

Необходимо предусматривать возможность различного размещения документов: сбоку от видеотерминала, между монитором и клавиатурой и т.п. Кроме того, в случаях, когда видеотерминал имеет низкое качество изображения, например заметны мелькания, расстояние от глаз до экрана делают больше (около 700мм), чем расстояние от глаза до документа (300-450мм). Вообще при высоком качестве изображения на видеотерминале расстояние от глаз пользователя до экрана, документа и клавиатуры может быть равным.

Положение экрана определяется:

* расстоянием считывания (0,6…0,7м);
* углом считывания, направлением взгляда на 20° ниже горизонтали к центру экрана, причем экран перпендикулярен этому направлению.

Должна также предусматриваться возможность регулирования экрана:

* по высоте +3 см;
* по наклону от -10° до +20° относительно вертикали;
* в левом и правом направлениях.

Требования к рабочей позе пользователя видеотерминала следующие:

* голова не должна быть наклонена более чем на 20°;
* плечи должны быть расслаблены;
* локти - под углом 80°…100°;
* предплечья и кисти рук - в горизонтальном положении.

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет 60…80 см, то высота знака должна быть не менее 3мм, оптимальное соотношение ширины и высоты знака составляет 3:4, а расстояние между знаками – 15…20% их высоты. Соотношение яркости фона экрана и символов - от 1:2 до 1:15.

Во время пользования компьютером медики советуют устанавливать монитор на расстоянии 50-60 см от глаз. Специалисты также считают, что верхняя часть видеодисплея должна быть на уровне глаз или чуть ниже. Когда человек смотрит прямо перед собой, его глаза открываются шире, чем когда он смотрит вниз. За счет этого площадь обзора значительно увеличивается, вызывая обезвоживание глаз. К тому же если экран установлен высоко, а глаза широко открыты, нарушается функция моргания. Это значит, что глаза не закрываются полностью, не омываются слезной жидкостью, не получают достаточного увлажнения, что приводит к их быстрой утомляемости.

Скорость движения воздуха в рассматриваемом помещении составляет 0,1 м/с. Это достигается с помощью приточно-вытяжной вентиляции. Данный параметр удовлетворяет требованию нормативов, значение которых не должны превышать 0,2 - 0,5 м/с в зимнее время, и 0,2 - 1,0 м/с – в летнее.

Основные параметры рабочего места и соответствие этих параметров нормативным документам приведены в таблице 4.2.

Таким образом, помещение своими размерами, отделкой, размерами и положением окон удовлетворяет требованиям нормативных документов. Мебель удобна в использовании и удовлетворяет эргономическим показателям. Расположение экранов монитора, а так же выбираемые режимы работы так же соответствуют нормам. Большинство параметров (например влажность воздуха или запыленность) удовлетворяет требованиям нормативных документов по безопасности труда.

Температура иногда превышает оптимальную, и это является недостатком, для устранения которого рекомендуется установить кондиционер в помещении.

Появление ЖК-дисплеев, а так же плазменных дисплеев предоставляет возможности для снижения влияния монитора на зрительные рецепторы человека. Кроме того значения ЭМИ от таких дисплеев на много ниже в сравнении с мониторами с ЭЛТ. Таким образом рациональным шагом по улучшению условий труда является замена ЭЛТ мониторов на жидкокристаллические или плазменные панели.

Для поддержания условий труда на безопасном уровне необходимо:

* правильно организовывать рабочее место;
* проводить обучение работающих безопасным методам труда;
* знать и уметь пользоваться средствами индивидуальной защиты;
* знать безопасные методы производства работ;
* соблюдать правила электробезопасности;
* соблюдать правила электропожаробезопасности;
* уметь пользоваться инструментами пожаротушения;
* установить кондиционер.

Таблица 4.2 – Соответствие параметров рабочего помещения требованиям нормативов

|  |  |  |
| --- | --- | --- |
| Параметр | Фактический показатель | Требования по нормативам |
| Площадь рабочего места, м2 | 6,06 | 6 |
| Объем пространства рабочего помещения, м3 | 19,4 | 20 |
| Шум, дБ | 40 | 80 |
| Освещенность, лк | 400 | 300-500 |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Температура, °С:  - в теплое время года  - в холодное время года | 27  19 | 20-25  18-22 | |
| Влажность воздуха, % | 60 | 50-75% | |
| Запыленность, мг/м3 | 2,96 | 6 | |
| Наличие средств индивидуальной защиты | - | + | |
| Наличие средств пожаротушения | + | + | |
| Наличие аптечки | + | - | |
| Эргономические показатели | + | + | |

# 4.2 Расчет тяжести и напряженности труда

Одним из наиболее важных факторов, определяющих качество трудового процесса, является соблюдение правильного режима труда и отдыха. Не правильно же выбранный режим труда и отдыха при работе с ПК приводит к значительному напряжение зритель­ного аппа­рата с появлением жалоб на неудовлетворенность работой, головные боли, раздражи­тельность, нарушение сна, усталость и болезненные ощущения в глазах, в по­яснице, в области шеи и руках.

Согласно лабораторному практикуму «Оценка тяжести и напряженности трудового процесса при аттестации рабочих мест по условиям труда» [51] тяжесть трудового процесса (труда) оценивают независимо от индиви­дуальных особенностей человека по следующим показателям:

- физическая динамическая нагрузка;

- масса поднимаемого и перемещаемого груза вручную;

- стереотипные рабочие движения;

- статическая нагрузка;

- рабочая поза;

- наклоны корпуса;

- перемещение в пространстве.

В таблице 4.3 приведены результаты оценки условий труда по показателям тяжести трудового процесса.

Согласно Р 2.2.755-99 «Гигиенические критерии оценки и классификация условий труда по показателям вредности и опасности факторов производственной среды, тяжести и напряженности трудового процесса.» оценка тяжести физического труда проводится на основе учета используемых в таблице показателей. Так как физическая нагрузка, связанная с перемещением тяжестей, отсутствует, то по большинству показателей, которые связанны с перемещением грузов, данный вид деятельности относится к оптимальному.

Таблица 4.3 - Результаты оценки условий труда по показателям тяжести трудового процесса

| Показатель тяжести трудового процесса | Нормативное  значение | | Фактическое  значение | Класс  условий |
| --- | --- | --- | --- | --- |
|  | Опти-мальное | Допустимое |  | труда |
| 1 | 2 | 3 | 4 | 5 |
| 1. Физическая динамическая нагрузка ( единицы внешней механической работы за смену, кг⋅м) | | | | |
| 1.1. При региональной нагрузке (с участием преимущественно мышц рук и плечевого пояса) при перемещении груза на расстояние до 1м  - для мужчин | до 2500 | до 5000 |  |  |
| - для женщин | до 1500 | до 3000 | отсутствует | 1 |
| 1.2.При общей нагрузке (с участием мышц рук, корпуса, ног): |  |  |  |  |
| 1.2.1. При перемещении груза на расстояние от 1 до 5 м |  |  |  |  |
| - для мужчин | до 12500 | до  25000 |  |  |
| - для женщин | до 7500 | до 15000 | отсутствует | 1 |
| 1.2.2.При перемещении груза на расстояние  более 5 м |  |  |  |  |
| - для мужчин | до 24000 | до  46000 |  |  |
| - для женщин | до 14000 | до  28000 | отсутствует | 1 |
| 2. Масса поднимаемого и перемещаемого груза вручную, кг | | | | |
| 2.1.Подъём и перемещение (разовое) тяжестей при чередовании с другой работой (до 2р. в час) |  |  |  |  |
| - для мужчин | до 15 | до 30 |  |  |
| - для женщин | до 5 | до 10 | отсутствует | 1 |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 2.2. Подъём и перемещение тяжестей постоянно в теч. рабочей смены |  |  |  |  |
| - для мужчин | до 5 | до 15 |  |  |
| - для женщин | до 3 | до 7 | отсутствует | 1 |
| 2.3. Суммарная масса грузов, перемещаемых в течение каждого часа смены: |  |  |  |  |
| 2.3.1. С рабочей поверхности |  |  |  |  |
| - для мужчин | до 250 | до 870 |  |  |
| - для женщин | до 100 | до 350 | отсутствует | 1 |
| 2.3.2. С пола |  |  |  |  |
| - для мужчин | до 100 | до 435 |  |  |
| - для женщин | до 50 | до 175 | отсутствует | 1 |
| 3. Стереотипные рабочие движения (количество за смену) | | | | |
| 3.1. При локальной нагрузке (с участием мышц кистей и пальцев рук) | до 20000 | до 40000 | до 28000 | 2 |
| 3.2. При региональной нагрузке (при работе с преимущественным участием мышц рук и плечевого пояса) | до 10000 | до 20000 |  |  |
| 4. Статическая нагрузка  Величина статической нагрузки за смену при удержании груза, приложении усилий, кг⋅с | | | | |
| 4.1. Одной рукой: |  |  |  |  |
| - для мужчин | до 18000 | до 36000 |  |  |
| - для женщин | до 11000 | до 22000 | отсутствует | 1 |
| 4.2. Двумя руками: |  |  |  |  |
| - для мужчин | до 36000 | до 70000 |  |  |
| - для женщин | до 22000 | до 42000 | отсутствует | 1 |
| 4.3. С участием мышц корпуса и ног: |  |  |  |  |
| - для мужчин | до 43000 | до 100000 |  |  |
| - для женщин | до 26000 | до 60000 | отсутствует | 1 |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 5. Рабочая поза | | | | |
| 5. Рабочая поза | Свободная удобная поза, нахождение стоя до 40% | Периодическое, до 25% времени смены, нахождение в неудобной позе, нахождение стоя до 60% | Нахождение в фиксированной позе более 50% времени смены | 3.2 |
| 6. Наклоны корпуса | | | | |
| 6. Наклоны корпуса (кол-во за смену) | до 50 | 51-100 | до 5 | 1 |
| 7.Перемещения в пространстве, обусловленные технологическим процессом, км | | | | |
| 7.1. По горизонтали | до 4 | до 8 | 3-4 км | 1 |
| 7.2. По вертикали | до 2 | до 4 | отс | 1 |
| Общая оценка тяжести трудового процесса | | | | 3.2 |

Из-за крайне низкой двигательной активности вследствие особенностей работы, перемещения в пространстве ограничены, по этому показателю условия труда так же оптимальны. Однако в связи с тем, что труд специалиста по системному анализу, либо специалиста по защите информационной безопасности предполагает нахождение в фиксированной позе, в виду технологического процесса, более 50% рабочего времени, класс труда относится к категории вредный труд степень тяжести (класс 3.2).

Окончательная оценка тяжести труда устанавливается по показателю, отнесенному к наибольшей степени тяжести. При наличии двух и более показателей класса 3.1 и 3.2 общая оценка устанавливается на одну степень выше 3.2 и 3.3 классы соответственно.

Оценка напряженности труда профессиональной группы работников основана на анализе трудовой деятельности и ее структуры, которые изучаются путем хронометражных наблюдений в течение всего рабочего дня. Анализ основан на учете всего комплекса производственных факторов (стимулов, раздражителей), создающих предпосылки для возникновения неблагоприятных нервно-эмоциональных состояний (перенапряжения). Все факторы (показатели) трудового процесса имеют качественную или количественную выраженность и сгруппированы по видам нагрузок: интеллектуальные, сенсорные, эмоциональные, монотонные, режимные нагрузки. Комплексная оценка напряженности трудового процесса проводится по совокупности всех показателей. При наличии 7 и более показателей класса 3.1 и 3.2 общая оценка устанавливается на одну степень выше 3.2 и 3.3 классы соответственно.

Таблица 4.4 – Результаты оценки напряженности труда

| Показатели напряженности трудового процесса | Класс условий  труда | | | | |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3.1 | 3.2 | 3.3 |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1. Нагрузка интеллектуального характера | | | | | |
| **1.1. Содержание работы** |  |  |  | **+** |  |
| 1.2 Восприятие сигналов (информации) и их оценка |  |  |  | + |  |
| 1.3. Распределение функций по степени сложности задания |  |  | + |  |  |
| 1.4. Характер выполняемой работы |  |  |  | + |  |
| 2. Сенсорные нагрузки | | | | | |
| 2.1. Длительность сосредоточенного наблюдения (в % от времени смены) |  |  |  | + |  |
| 2.2. Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы | + |  |  |  |  |
| 2.3. Число производственных объектов одновременного наблюдения | + |  |  |  |  |
| 2.4. Размер объекта различения при длительности сосредоточенного внимания (в % от времени смены) |  |  | + |  |  |
| 2.5. Работа с оптическими приборами (микроскоп, лупа (в % от времени смены)) | + |  |  |  |  |
| 2.6. Наблюдение за экраном видеотерминала (часов в смену) |  |  |  | + |  |
| 2.7. Нагрузка на слуховой анализатор | + |  |  |  |  |
| 2.8. Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемых в неделю) | + |  |  |  |  |
| 3. Эмоциональные нагрузки | | | | | |
| 3.1. Степень ответственности за результат собственной деятельности. Значимость ошибки. |  |  |  | + |  |
| 3.2. Степень риска для собственной жизни | + |  |  |  |  |
| 3.3. Степень ответственности за безопасность других лиц | + |  |  |  |  |
| 3.4. Количество конфликтных ситуаций, обусловленных профессиональной деятельностью, за смену |  |  |  | + |  |

Продолжение таблицы 4.4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | | 6 | |
| 4. Монотонность нагрузок | | | | | | | |
| 4.1. Число элементов (приёмов), необходимых для реализации простого задания или многократно повторяющихся операций. | | + |  |  |  | |  |
| 4.2. Продолжительность (в сек.) выполнения простых производственных заданий или повторяющихся операций. | | + |  |  |  | |  |
| 4.3. Время активных действий (в % к продолжительности смены) | | + |  |  |  | |  |
| 4.4. Монотонность производственной обстановки(время пассивного наблюдения за ходом тех. процесса в % от времени смены) | |  |  | + |  | |  |
| 5. Режим работы | | | | | | | |
| 5.1. Фактическая продолжительность рабочего дня | |  |  | + |  | |  |
| 5.2. Сменность работы | |  | + |  |  | |  |
| 5.3. Наличие регламентированных перерывов и их продолжительность (без обеденного перерыва) | |  |  | + |  | |  |
| Количество показателей в каждом классе | | 10 | 1 | 5 | 7 | |  |
| Общая оценка напряжённости труда | | 3.3 | | | | | |

Таким образом напряженность труда в соответствии с расчетами относится к классу 3.3 (вредный труд 3 степени тяжести).

Кроме того существует расчет тяжести труда относительно рабочей среды. Оценка тяжести труда градуированна баллами и рассчитывается в соответствии с методикой по формуле:

|  |  |
| --- | --- |
| , | (4.1) |

где *x*max – наивысшая из полученных частных балльных оценок хi;

*N –* общее чис­ло факторов;

*xj –* балльная оценка по i-му из учитываемых факторов (частная бал­льная оценка в соответствии с критериями для бальной оценки факторов рабочей среды);

*п –* число учитываемых факторов без учета одного фактора.

Таким образом необходимо сопоставить условия, в которых работает сотрудник с условиями из таблицы критериев для бальной оценки факторов рабочей среды, приведенной в методических указаниях [52].

В результе по каждому из критериев баллы следующие:

X1 (температура воздуха) – 3 балла;

X2 (токсичное вещество) – 1 балл;

X3 (вибрация) – 2 балла;

X4 (промышленный шум) – 1 балл;

X5 (ультразвук) – 1 балл;

X6 (интенсивность теплового излучения) – 2 балла;

X7 (освещенность рабочего места) – 2 балла;

X8 (динамическая нагрузка) – 1 балл;

X9 (статическая нагрузка) – 1 балл;

X10 (рабочее место) – 5 баллов;

X11 (сменность) – 4 балла;

X12 (напрерывная работа) – 3 балла;

X13 (длительность наблюдения) – 5 баллов;

X14 (число важных объектов) – 4 балла;

X15 (темп кисти) – 5 баллов;

X16 (темп рук) – 4 балла;

X17 (число сигналов в час) – 4 балла;

X18 (монотонность приемов в операции) – 1 балл;

X19 (длительность повторяющихся операций) – 4 балла;

X20 (режим труда и отдыха) – 3 балла;

X21 (нервно эмоциональная нагрузка) – 5 баллов.

Подставив значения в расчетную формулу 5.1, необходимо произвести расчет тяжести труда. В результате получилось следующее значение T = 4,5 балла. Таким образом, в соответствии с интегральной бальной оценкой труда, данный вид деятельности соответствует 3 категории тяжести. Следовательно, данный вид деятельности относится к работам, при которых вследствие не вполне благоприятных условий труда у работающего формируются реакции, характерные для пограничного состояния организма.

В таблице 5.4 представлены сведения о регламентированных перерывах, которые необходимо делать при работе на компьютере, в зависимости от продолжительности рабочей смены, видов и категорий трудовой деятельности с ВДТ (видеодисплейный терминал) и ПЭВМ (в соответствии с СанПиН 2.2.2 542-96 «Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работ»).

Таблица 5.4 – Время регламентированных перерывов при работе на компьютере

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Категория работы с ВДТ или ПЭВМ | Уровень нагрузки за рабочую смену при видах работы с ВДТ | | | Суммарное время регламентированных перерывов, мин | |
| Группа А, количество знаков | Группа Б, количество знаков | Группа В, часов | При 8-часовой смене | При 12-часовой смене |
| I | до 20000 | до 15000 | до 2,0 | 30 | 70 |
| II | до 40000 | до 30000 | до 4,0 | 50 | 90 |
| III | до 60000 | до 40000 | до 6,0 | 70 | 120 |

Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

В соответствии со СанПиН 2.2.2 546-96 все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

* группа А: работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;
* группа Б: работа по вводу информации;
* группа В: творческая работа в режиме диалога с ЭВМ.

Категория труда в данном случае в соответствии с интегральным бальным показателем относится к 3 (класс 3.3 по напряженности и 3.1 по тяжести), а группа труда В (творческая работа в режиме диалога с ЭВМ). Официально рабочая смена длится 8 часов, следовательно суммарное время регламентированных перерывов должно составлять 70 минут.

Для уменьшения дискомфорта от сидячей работы за персональным компьютером рекомендуется применить следующие меры:

- выбор оптимального разрешения экрана монитора в соответсвии с индивидуальными особенностями и диагональю

- возможно, эффективным решение будет использование монитора с диагональю 22 дюймов и выше и разрешением 1920x1080 пикселей, что дает картинку высокой четкости (Full HD). При этом, за счет большой диагонали экрана возможно отдаление глаз оператора на расстояние метра и более при сохранении наиболее удобного шрифта для восприятия и увеличения обзора рабочей среды;

- наличие комфортной обстановки (например, установка аквариума) может создать благоприятный фон для напряженной работы за ПЭВМ;

- необходимо так же обеспечивать индивидуальное освещение для рабочего места данного специалиста;

- важным аспектом является наличие активных физических пауз (физ. минутки);

- необходимо обеспечить такие условия труда, при которых сотрудники могли бы отдыхать в отдельном помещении (желательно наличие зеленой зоны, либо небольшого спортивного помещения, например, с установленным столом для настольного тенниса).

# 4.3 Возможные чрезвычайные ситуации

Возможная чрезвычайная ситуация, которая может произойти в непосредственной близости от ГОУ «ОГУ» – это выброс аварийного химически опасного вещества (АХОВ) в результате аварии железнодорожной цистерны.

При оценке [47] определяются глубины зон заражения и другие величины. Масштабы заражения АХОВ в зависимости от их физических свойств и агрегатного состояния рассчитываются по первичному и вторичному облаку АХОВ: для сжиженных газов – отдельно по первичному и вторичному облаку; для сжатых газов – только по первичному облаку; для ядовитых жидкостей, кипящих выше температуры окружающей среды – только по вторичному облаку.

Предположим, что произошла авария с выбросом хлора. Хлор — токсичный удушливый газ, при попадании в лёгкие вызывает ожог лёгочной ткани, удушье. Раздражающее действие на дыхательные пути оказывает при концентрации в воздухе около 0,006 мг/л (то есть в два раза выше порога восприятия запаха хлора). Предельно допустимые концентрации хлора в атмосферном воздухе следующие: среднесуточная — 0,03 мг/м³; максимально разовая — 0,1 мг/м³; в рабочих помещениях промышленного предприятия — 1 мг/м³. При работе с хлором следует пользоваться защитной спецодеждой, противогазом, перчатками. На короткое время защитить органы дыхания от попадания в них хлора можно тряпичной повязкой, смоченной раствором сульфита натрия Na2SO3 или тиосульфата натрия Na2S2O3. Хлор был одним из первых химических отравляющих веществ, использованных Германией в Первую мировую войну, впервые применен в 1915 году во время битвы при Ипре.

В результате аварии произошел выброс 10 тонн хлора в атмосферу на железной дороге, расположенной в 0,5 км от ГОУ «ОГУ». Скорость ветра 3 м/с. Проведем расчет размеров и площади зоны заражения, а так же времени подхода зараженного облака к ГОУ «ОГУ».

Определим глубину зоны заражения из таблицы «Глубина распространения облаков зараженного воздуха с поражающими концентрациями АХОВ на открытой местности». Она составляет при ирзотермии 7 км. Так как скорость ветра 3 м/с, определяем поправочный коэффициент по таблице влияния скорости ветра на глубину распространения зараженного воздуха с поражающей концентрацией.

С учетом поправочного коэффициента для скорости ветра 3 м/с при изотермии глубина заражения составляет 3,85 километра

Определим ширину зоны химического заражения. Ширина (Ш) зоны химического заражения зависит от степени вертикальной устойчивости воздуха и определяется для изотермии как:

Ш = 0.15\*Г. (4.2)

Таким образом:

Ш = 0.15\*3.85 = 0.58 км.

Площадь зоны заражения определяется по формуле:

S = 0.5\*Г\*Ш. (4.3)

Таким образом:

S = 0.5\*0.58\*3.85 = 1.12 км.

По таблице для изотермии и скорости ветра 3 м/с определим среднюю скорость переноса облака зараженного хлором, в данном случае она составляет 18 км/ч или 5м/с.

Время подхода облака t к заданному объекту зависит от скорости переноса облака воздушными потоками и определяется по формуле:

(4.4)



Подставив известные значения в формулу, находим:



Время поражающего действия АХОВ в очагах химического поражения определяется временем испарения АХОВ с поверхности его выброса (разлива).



По таблице времени испарения некоторых АХОВ определяем время поражающего действия хлора (время испарения) при скорости ветра 1м/с. Время испарения хлора равно 1,3 ч.

Время испарения используется для определения ориентировочного времени поражающего действия АХОВ в очаге химического поражения.

Т.к. скорость ветра больше 1 м/с, введем поправочный коэффициент. Для скорости 3 м/с он будет равен 0.55.

Время поражающего действия хлора, составит:



= 1,3\*0,55=0,72 часа = 43.2 мин.

Потери рабочих, служащих и населения в очаге химического поражения зависят от токсичности, количества АХОВ и времени пребывания людей в очагах поражения, степени защищенности и своевременности использования индивидуальных средств защиты (противогазов). Характер поражения людей, находящихся в зоне химического поражения может быть различным. Он определяется главным образом токсичностью АХОВ и полученной токсидозой.

Предположим, в момент аварии в ГОУ «ОГУ» находилось 1200 человек учащихся и служащих, обеспеченных противогазами на 60%. Определяем потери P среди людей, находящихся в ГОУ «ОГУ» по таблице «Возможные потери рабочих, служащих и населения от АХОВ в очагах поражения»:

P = 1200\*0.22 = 264 чел.

Ориентировочная структура потерь людей в очаге поражения составляет

* со смертельным исходом 35% - 264\*0,35 = 93 чел.
* средней и тяжелой степени 40% - 510\*0,4 = 106 чел.
* легкой степени 25% - 510\*0,25 = 65 чел.

Эвакуация сотрудников в случае возникновения чрезвычайной ситуации производится в соответствии с планом эвакуации.

Следующей возможной чрезвычайной ситуацией является пожар. Пожарная профилактика основывается на исключении условий, необходимых для горения, и использовании прин­ципов обеспечения безопасности. При обеспечении пожар­ной безопасности решаются четыре задачи: предотвраще­ние пожаров и загораний, локализация возникших пожа­ров, защита людей и материальных ценностей, тушение пожаров. Пожарная безопасность обеспечивается предотвращением пожаров и пожарной защитой. Предотвраще­ние пожара достигается исключением образования горючей среды и источников зажигания, а также поддержанием параметров среды в пределах, исключающих горение. Предотвращение образования источников зажигания достигается следующими мероприятиями: соответствую­щим исполнением, применением и режимом эксплуатации машин и механизмов; устройством молниезащиты зданий и сооружений; ликвидацией условий для самовозгорания; регламентацией допустимой температуры и энергии искро­вого разряда; курение и использование открытого огня только в специальных местах; обучение рабочих и служащих мерам пожарной безопасности.

Выбор типа и расчет необходимого количества огнетушителей рекомендуется производить в зависимости от их огнетушащей способности, предельной площади, класса пожара горючих веществ и материалов в защищаемом помещении или объекте.

Для рассчитываемого помещения класс пожара – (Е). Класс (Е) — это пожары, связанные с горением электроустановок. Категория помещения определяется как «Общественные здания». Для тушения очагов пожаров в нашем помещении площадью около 20 м2 используется 1 порошковый огнетушитель емкостью 5 литров, что соответствует «Правилам пожарной безопасности». На рабочих местах пожар может случиться только по вине самих работников, не исполняющих меры противопожарной защиты. Возгорание может произойти вследствие оставления без присмотра вычислительных машин и различной, питающейся от электросети техники.

При эвакуации сотрудников ГОУ «ОГУ», происходит движение людей из помещения аудитории 14137 по коридору первого этажа к выходу через дверной проем, длина пути приближенно 38м;

Далее происходит выход людей из здания и рассеивание.

Основными параметрами, характеризующими процесс эвакуации людей из зданий и сооружений, являются: плотность движения D, скорость движения v людского потока, пропускная способность пути (выходов) Q и интенсивность движения q. Кроме того, эвакуационные пути, как горизонтальные, так и наклонные, характеризуются длиной L, и шириной b движения.

Расчетное время эвакуации людей T, определяется как сумма времени движения людского потока по отдельным участкам пути:

, (4.5)

где t1, t2,…, tn – время движения людского потока на каждом этапе эвакуации.

Время движения людского потока по любому участку пути вычисляется по формуле:

, (4.6)

где *vi* – скорость людского потока на i-том участке, определяемая по таблице в зависимости от плотности Di.

Плотность людского потока Di, состоящего из N людей находится по формуле:

, (4.7)

где Ni – среднее число людей на i-том участке;

Ai – площадь пути эвакуационного участка, м2.

N1 = 90 человек

Площадь i-того эвакуационного участка рассчитывается по формуле:

, (4.8)

где b – ширина коридора (b = 2 м).

Тогда плотность людского потока на участке эвакуации равна:

D =  = 1,18 чел./м;

Процесс движения людей в силу угрожающей им опасности инстинктивно начинается одновременно в одном направлении – в сторону выходов. Это приводит к тому, что проходы быстро заполняются людьми определенной плотности потоков. Образование высоких плотностей наиболее вероятно и наиболее опасно в случае вынужденной эвакуации, так как с увеличением плотности потока скорость движения уменьшается.

В соответствии с методическим пособием по определению уровня обеспечения пожаробезопасности при плотности людского потока 0.9 и выше, скорость движения по горизонтальному пути равна 15 м/мин.

Отсюда время движения сотрудников ГОУ «ОГУ» при эвакуации:



Итак, время эвакуации сотрудников из университета больше времени подхода облака АХОВ на 52 секунды. Таким образом, для защиты сотрудников необходимо применять индивидуальные средства защиты, такие как респираторы или противогазы, во время эвакуации из здания университета.

Таким образом, безопасность условий труда на рассматриваемом объекте соответствует нормам безопасности жизнедеятельности. Для улучшений условий труда так же представлены конкретизированные рекомендации. Рассчитана тяжесть и напряженность условий труда, а так же необходимое время перерыва сотрудника в производственном процессе. Кроме того, рассмотрены наиболее вероятные чрезвычайные ситуации в данном районе: рассмотрен случай выброса аварийного химически опасного вещества (АХОВ) (в данном случае рассматривается выброс 10т. хлора) и возможности эвакуации при выбросе, а так же при пожаре.

# 5 Экономическое обоснование выпускной квалификационной работы

# 5.1 Организационно - экономическая часть

Одной из целей дипломного проектирования является получение положительного экономического эффекта от улучшения методов защиты сети от вторжений. В настоящее время средства защиты информации сформировали целый рынок, а в условиях рыночной экономики любое решение будет конкурировать с другими подобными решениями. Именно поэтому определение экономического эффекта от разработки и внедрения новых программных продуктов в сфере защиты информации имеет огромную важность.

Любое улучшение средств защиты информации ведет к снижению риска. Именно в снижении возможности потерь за счет уменьшения количества угроз при использовании более качественного подхода к защите и заключается экономический эффект. Чем больше возможностей по защите информации предоставляет нововведении, тем больше будет экономическая выгода от использования разработанного метода в реальных условиях.

В данном разделе дипломного проекта произведен расчёт экономического эффекта [50] от разработки и внедрения программного продукта для выбора правил системы обнаружения вторжений.

# 5.2 Расчет трудоёмкости разработки программного продукта

Расчёт затрат времени на разработку программного обеспечения охватывает работы выполняемые специалистами на следующих стадиях:

- техническое задание (ТЗ);

- эскизный проект (ЭП);

- технический проект (ТП);

- рабочий проект (РП);

- внедрение.

Данные о трудоемкости каждой из стадий по плану и хронометражу представлены в таблице 5.1.

Таблица 5.1 - Данные о трудоемкости на стадиях разработки программного обеспечения по плану и хронометражу

|  |  |  |
| --- | --- | --- |
| Стадия | Трудоемкость, дней | |
|  | По плану | По хронометражу |
| Техническое задание | 10 | 8 |
| Эскизный проект | 20 | 27 |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Технический проект | 30 | 53 |
| Рабочий проект | 20 | 38 |
| Внедрение | 5 | 4 |

При расчёте фактических затрат времени на программирование необходимо учесть влияние следующих факторов:

- количество разновидностей форм входной информации;

- количество разновидностей форм выходной информации;

- степень новизны комплекса задач;

- сложность алгоритма;

- виды используемой информации;

- сложность контроля входной и выходной информации;

- объем входной информации;

- использование типовых проектных решений.

Предусматривается четыре степени новизны разрабатываемых задач, которые представлены в таблице 5.2.

Таблица 5.2 - Степень новизны разрабатываемых задач

|  |  |
| --- | --- |
| Обозначение | Степень новизны |
| А | Разработка комплекса задач, предусматривающая применение новых методов разработки, проведения научно-исследовательских работ |
| Б | Разработка решений задач и систем, не имеющих аналогов |
| В | Разработка решений задач и систем, имеющих аналогичное решение |
| Г | Привязка типовых проектных решений |

# 5.2.1 Сложность алгоритма

Сложность алгоритма представлена тремя группами в таблице 4.3.

Таблица 5.3 - Группы сложности алгоритмов

|  |  |
| --- | --- |
| Группа | Виды алгоритмов |
| 1 | Алгоритмы оптимизации и моделирования систем и объектов |
| 2 | Алгоритмы учёта и отчётности, статистики и поиска |
| 3 | Алгоритмы, реализующие стандартные методы решения |

Трудоёмкость разработки проекта зависит от вида используемой информации и режима работы. Виды информации и режимы работы представлены втаблице 5.4.

Таблица 5.4 - Виды используемой информации и режима работы

|  |  |
| --- | --- |
| Обозначение | Виды информации и режима работы |
| ПИ | Переменная информация |
| НСИ | Нормативно-справочная информация |
| БД | Базы данных |
| РВ | Режим работы в реальном времени |
| ТОУ | Режим телекоммуникационной обработки и управления |

# 5.2.2 Сложность организации контроля входной и выходной информации

Сложность организации контроля входной и выходной информации представлена в таблице 5.5 четырьмя группами.

Таблица 5.5 - Группы сложностей организации контроля входной и выходной информации

|  |  |
| --- | --- |
| Группа | Группа сложности |
| 11 | Входные данные и документы разнообразного формата и структур (контроль осуществляется перекрёстно) |
| 12 | Входные данные и документы однообразной формы и содер­жания (осуществляется формальный контроль) |
| 21 | Печать документов сложной многоуровневой структуры, разнообразной формы и содержания |
| 22 | Печать документов однообразной формы и содержания, вывод массивов данных на машинные носители |

Далее в справочных таблицах 5.6-5.11 представлены затраты времени при выполнении различных видов работ на стадиях процесса разработки программного продукта.

Таблица 5.6 - Затраты времени при выполнении работ на стадии ТЗ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Комплекс задач, подсистем | Степень новизны  (в днях) | | | |
| А | Б | В | Г |
| 1 | 2 | 3 | 4 | 5 |

Продолжение таблицы 5.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Перспективное планирование, управление проектируемым капитальным строительством, технико-экономическое планирование, оперативное управление и управление ценообразованием | 79 | 57 | 37 | 34 |
| Управление материально-техническим снабжением, сбытом продукции; управление комплектацией, экспортными и импортными поставками | 105 | 76 | 42 | 30 |
| Бухгалтерский учет и управление финансовой деятельностью | 103 | 72 | 30 | 35 |
| Управление организацией труда, кадрами и зарплатой | 63 | 46 | 30 | 19 |
| Управление качеством продукции и технологическими процессами в производстве | 64 | 47 | 31 | 22 |
| Управление транспортными перевозками, техобслуживанием, вспомогательными службами и энергоснабжение | 91 | 66 | 43 | 26 |
| Управление научно-технической информаци­ей. Совершенствование документооборота и контроль исполнения документа. Управление охраной природы и окружающей среды | 50 | 36 | 24 | 15 |
| Учет пенсий, пособий и страховых операций | 79 | 55 | 36 | 26 |
| Статистические задачи | 129 | 111 | 61 | 38 |
| Задачи расчётного характера | 92 | 69 | 47 | 29 |

Таблица 5.7 - Затраты времени при выполнении работ на стадии ЭП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Комплекс задач, подсистем | Степень новизны  (в днях) | | | |
| А | Б | В | Г |
| 1 | 2 | 3 | 4 | 5 |
| Перспективное планирование, управление проектируемым капитальным строительством, технико-экономическое планирование, опера­тивное управление и управление ценообразо­ванием | 175 | 117 | 77 | 53 |
| Управление материально-техническим снаб­жением, сбытом продукции; управление ком­плектацией, экспортными и импортными по­ставками | 115 | 79 | 53 | 35 |
| Бухгалтерский учет и управление финансовой деятельностью | 166 | 112 | 67 | 57 |

Продолжение таблицы 5.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Управление организацией труда, кадрами и зарплатой | 151 | 101 | 67 | 44 |
| Управление качеством продукции и техноло­гическими процессами в производстве | 157 | 99 | 67 | 44 |
| Управление транспортными перевозками, техобслуживанием, вспомогательными служ­бами и энергоснабжение | 170 | 100 | 70 | 45 |
| Управление научно-технической информаци­ей. Совершенствование документооборота и контроль исполнения документа. Управление охраной природы и окружающей среды | 151 | 101 | 67 | 46 |
| Учет пенсий, пособий и страховых операций | 103 | 70 | 45 | 36 |
| Статистические задачи | 103 | 70 | 45 | 49 |
| Задачи расчётного характера | 103 | 70 | 45 | 41 |

Таблица 5.8 - Поправочные коэффициенты, (К1, К2, К3) для определения трудо­ёмкости работ на стадии ТП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Вид используемой информации | Степень новизны | | | |
| А | Б | В | Г |
| 1 | ПИ, К1 | 1,7 | 1,2 | 1 | 0,5 |
| 2 | НСИ, К2 | 1,45 | 1,08 | 0,72 | 0,43 |
| 3 | БД, К3 | 4,37 | 3,12 | 2,08 | 1,25 |

Таблица 5.9 - Поправочные коэффициенты (К1, К2, К3) определения трудоёмкости работ на стадии РП

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вид используемой информации | Группа сложности алгоритма | Степень новизны | | | |
| А | Б | В | Г |
| 1 | 2 | 3 | 4 | 5 | 6 |
| ПИ, К1 | 1 | 2,27 | 1,62 | 1,2 | 0,65 |
| 2 | 2,02 | 1,44 | 1,1 | 0,58 |
| 3 | 1,68 | 1,2 | 1 | 0,48 |
| НСИ, К2 | 1 | 1,36 | 0,97 | 0,65 | 0,4 |
| 2 | 1,21 | 0,86 | 0,58 | 0,34 |
| 3 | 1,01 | 0,72 | 0,48 | 0,29 |
| БД, К3 | 1 | 1,14 | 0,81 | 0,54 | 0,32 |
| 2 | 1,05 | 0,72 | 0,48 | 0,29 |
| 3 | 0,85 | 0,6 | 0,4 | 0,24 |

При использовании информации разных видов для технического и рабо­чего проекта поправочный коэффициент рассчитывается по формуле (5.1):

 (5.1)

где m - количество наборов данных ПИ;

n - количество наборов данных НСИ;

р - количество наборов данных БД.

Таблица 5.10 - Поправочные коэффициенты, учитывающие сложность контро­ля входной и выходной информации на стадиях ТП, РП и внедрения

|  |  |  |
| --- | --- | --- |
| Сложность контроля входной информации | Сложность контроля выходной информации | |
| 21 | 22 |
| 11 | 1,16 | 1,07 |
| 12 | 1,08 | 1 |

Таблица 5.11 - Поправочные коэффициенты трудоёмкости ра­бот

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Стадия разработки | Вид обрабатывае­мой информации | Степень новизны | | | |
| А | Б | В | Г |
| ТП | РВ | 1,67 | 1,45 | 1,26 | 1,1 |
| ТОУ | 1,75 | 1,52 | 1,36 | 1,15 |
| РП | РВ | 1,75 | 1,52 | 1,36 | 1,15 |
| ТОУ | 1,92 | 1,67 | 1,44 | 1,25 |
| В | РВ | 1,6 | 1,39 | 1,21 | 1,05 |
| ТОУ | 1,67 | 1,45 | 1,26 | 1,1 |

Общая трудоёмкость разработки программного продукта рассчитывает­ся по формуле (5.2):

*ТОБЩ = tТЗ + tЭП + tТП + tРП + tВ* (5.2)

где tТЗ - затраты труда на стадии технического задания (в днях);

tЭП - затраты труда на стадии эскизного проекта (в днях);

tТП - затраты труда на стадии технического проекта (в днях);

tРП - затраты труда на стадии рабочего проекта (в днях);

tВ - затраты труда на стадии внедрения (в днях).

Трудоёмкости разработки на этапах: техническое задание tТЗ - определя­ется из таблицы 5.6, эскизный проект tЭП - определяется из таблицы 5.7, ос­тальные трудоёмкости определяются методом хронометража, подставив в фор­мулу (5.2) числовые значения трудоёмкости разработки программного продукта на каждом этапе без учёта поправочных коэффициентов, получим:

ТОБЩ = 69 + 70 + 53 + 38 + 4 = 234 дней

Общая трудоёмкость разработки программного продукта с учётом по­правочных коэффициентов рассчитывается по формуле (5.3):

Т’ОБЩ = t’ТЗ + t’ЭП + t’ТП + t’РП + t’В (5.3)

где t’ТП затраты труда на стадии технического проекта с учётом поправочного коэффициента;

t’РП - затраты труда на стадии рабочего проекта с учётом поправочного коэффициента;

t’В - затраты труда на стадии внедрения с учётом поправки.

Теперь с учётом поправки на использование разных видов информации и на основе справочной таблицы (5.10), (5.11) вычислим затраты труда на ста­дии технического проекта с учётом поправки:

t’ТП = 53 \* 1,07 \* 1,45 = 82,23 дней

С учётом поправки на использование разных видов информации и на основе справочных таблиц 4.10 и 4.11 вычислим затраты труда на стадии рабочего проекта с учётом поправки:

t’РП = 38 \* 1,07 \* 1,52 = 61,80 дней

Для расчета затрат труда на стадии внедрения используются поправоч­ные коэффициенты из справочных таблиц (5.10) и (5.11):

t’В = 4 \* 1,07 \* 1,39 = 5,95 дня

Таким образом, общие затраты труда на разработку программного про­дукта с учётом поправочных коэффициентов составят:

Т’ОБЩ = 69 + 70 + 82,23 + 61,80 + 5,95 = 288,98 дней

Численность исполнителей определяется по формуле 5.4:

, (5.4)

где Ч - численность исполнителей;

Fпл - время по хронометражу (находится по таблице 5.1).

Количество исполнителей в зависимости от значения Ч рассчитывается следующим образом:



где Исп- количество исполнителей.

Согласно формулe 5.4 вычисляем показатель Ч:

Ч=288,98/(8+27+53+38+4)=2,22

следовательно, для реализации проекта необходимо два исполнителя.

# 5.3 Расчёт себестоимости программного продукта

Себестоимость программного продукта рассчитывается по формуле 5.5:

 (5.5)

где Мвс - затраты на вспомогательные материалы, руб.;

Э - затраты на электроэнергию на технологические цели, руб.;

 - основная зарплата разработчика, руб.;

 - дополнительная зарплата разработчика, руб.;

 - отчисления на социальные нужды, руб.;

Н - накладные расходы, руб.

Затраты на вспомогательные материалы, М*вс*, руб., рассчитываются по формуле:

, (5.6)

где: Ц*д* – цена одного компакт - диска, Ц*д* = 20 руб.;

З*ч* – затраты на расходные материалы для принтера, З*ч* = 120 руб.;

З*б* – затраты на бумагу, З*б* = 500 руб.;

С*у* – стоимость услуги без лимитный Internet, С*у* = 600руб.;

N*м* – количество месяцев пользования услугой без лимитный Internet, N*м* = 4мес.

Рассчитываем затраты на вспомогательные материалы по формуле (5.6):

М*вс* = 20 + 120 + 500 + 4 \* 600 = 3040 руб.

Затраты на электроэнергию на технологические цели Э, руб., определяются по формуле:

, (5.7)

где Р – мощность потребления электрической энергии, Р = 1 кВт;

С*э* – стоимость одного кВт∙ч электрической энергии, С*э* = 1,87 руб;

t – общая трудоемкость, чел./ч;

R*заг* – коэффициент загрузки компьютера.

Затраты на электроэнергию на технологические цели по формуле (5.7) будут равны:

Э = 1 \* 1,87 \* 288,98 \* 8 \* 0,7 = 3026,20 руб.

Основная заработная плата разработчика З*ос*, руб., определяется по формуле (4.8):

, (5.8)

где С*чтс* – часовая тарифная ставка разработчика, С*чтс* = 30 руб./ч;

t – общая трудоемкость, чел./ч.

Рассчитаем затраты на основную заработную плату разработчика по формуле (4.8):

З*ос* = 30 \* 288,98 \* 8 = 69355,2 руб.

Дополнительная заработная плата разработчика З*д*, руб., определяется по формуле:

, (5.9)

где З*ос* – основная заработная плата разработчика, руб.;

R*д* – коэффициент отчислений на дополнительную заработную плату, R*д* = 0,15 руб./ч.

Рассчитаем затраты на дополнительную заработную плату разработчика по формуле (5.9):

З*д* = 69355,2 \* 0.15 = 10403,28 руб.

Отчисления на социальные нужды З*сн*, руб., определяются по формуле (5.10):

, (5.10)

где З*ос* – основная заработная плата разработчика, руб.;

З*д* – дополнительная заработная плата разработчика, руб.;

R*сн* – коэффициент отчисления на социальные нужды, R*сн* = 0,35.

Отчисления на социальные нужды по формуле (4.10) будут равны:

З*сн* = (69355,2 + 10403,28) \* 0,35 = 27915,47 руб.

Накладные расходы Н, руб., определяются по формуле:

, (5.11)

где З*ос* – основная заработная плата разработчика, руб.;

З*д* – дополнительная заработная плата разработчика, руб.;

R*нр* – коэффициент отчисления на накладные расходы, R*нр* = 0,1

Рассчитаем накладные расходы по формуле (4.11):

Н = (69355,2 + 10403,28) \* 0.1 = 7975,85 руб.

Сложив все полученные затраты найдем полную себестоимость программного продукта по формуле (4.5)

С = 3040 + 3026,20 + 69355,2 + 10403,28 + 27915,47 + 7975,848 = 121715,99 руб.

# 5.4 Расчёт экономического эффекта

Экономический эффект рассчитывается по формуле (4.12):

Эф =N\*(t1-t2) \* Счтс \* Кдоп \* Ксн - Ен \* К, (5.12)

где N - среднее количество информационных потоков с вредоносным трафиком в корпоративных сетях за год (примерно 3500);

t1,t2-трудоёмкость обработки заявок до и после внедрения программы;

Счтс - часовая тарифная ставка пользователя, *Счтс* = 30 руб.;

Кдоп - коэффициент отчислений на доп. зарплату, *Кдоп=* 1,15;

Ксн - коэффициент отчислений на социальные нужды, *Ксн* = 1,35;

Ен - нормативный коэффициент окупаемости вложений, *Ен* = 0,39;

К - дополнительные капитальные вложения, связанные с разработкой программного продукта, *K* = 121715,99р.

Используя ранее рассчитанные значения и подставив их в формулу (4.12), получим значение экономического эффекта:

Эф = 3500 \* (2 – 0,25) \* 30 \* 1,15 \* 1,35 - 0,39 \* 121715,99 = 237802,64 руб.

Срок окупаемости проекта рассчитывается по следующей формуле:

Т(ок) = (Ен \* К)/Эф = (0,39 \* 121715,99)/237802,64 руб = 0,2 года=2,4 мес.

Рассчитаем стоимостные затраты С1 с использованием старой системы и С2 с использованием разработанной системы:

С1 =N \* t1 \* Счтс \* Кдоп \* Ксн (5.13)

С2 =N \* t2 \* Счтс \* Кдоп \* Ксн (5.14)

С1 = 3500 ∙ 2 ∙ 30 ∙ 1,15 ∙ 1,35 = 326025

С2 = 3500 ∙ 0,25 ∙ 30 ∙ 1,15 ∙ 1,35= 40753,125

Полученные экономические показатели проекта сведены в таблице 5.14.

Таблица 5.14 - Экономические показатели проекта

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Значение | Ед. изм. |
| Вспомогательные материалы | 3040 | руб. |
| Основная зарплата | 69355,2 | руб. |
| Дополнительная зарплата | 10403,28 | руб. |
| Отчисления на социальные нужды | 27915,47 | руб. |
| Затраты на электроэнергию | 3026,20 | руб. |
| Накладные расходы | 7975,848 | руб. |
| Экономический эффект | 237802,64 | руб. |
| Срок окупаемости | 2,4 | месяца |

Из таблицы видно, что использование формализованной методики построения правил для СОВ выгодно не только с технической точки зрения, но и с экономической. Детальный анализ экономического эффекта от внедрения методики положительный. При сроке окупаемости в два с половиной месяца внедрение такой методики принесет большую прибыль компании

# Заключение

Целью выпускной квалификационной работы являлось повышение технической и экономической эффективности работы СОВ, для достижения которой были решены следующие задачи:

* проведен анализ современных СОВ и методов повышении эффективности работы СОВ, в результате чего была составлена признаковая классификация современных СОВ. Так же детальный анализ различных методов повышения эффективности показал, что в данной сфере существует множество проблем и множество задач до сих пор не решены, следовательно данная тема является актуальной;
* были разработаны требования к проектируемой СОВ;
* была разработана структурная модель формализованной методики выбора правил для СОВ;
* была разработана математическая модель формализованной методики выбора правил для СОВ. На основе структурной и математической моделей подробно расписана суть работы разрабатываемой методики;
* был произведен выбор СОВ и АУ, которые будут использоваться для реализации формализованной методики построения правил для СОВ;
* была проведена разработка алгоритмической модели формализованной методики построения правил для СОВ. Алгоритмическая модель представляет собой детальное описание действий в рамках разрабатываемой методики на псевдоязыке;
* был разработан алгоритм работы программного обеспечения, на основе которого можно реализовать формализованную методику построения правил для СОВ;
* проведено тестирование разрабатываемой методики. Для оценки эффективности был выбран метод сравнения количественных оценок. Был представлен расчет с предполагаемыми значениями. Затем был выбран метод атаки (DDoS) и ПО для генерации DDoS атак. Так же было установлено выбранное ПО и сконфигурировано таким образом, что бы реализовывать разрабатываемую методику. Был проведен тест на некоторой выборке атак. Данная выборка не является достаточной, но частично подтвердила повышение эффективности СОВ от использования формализованной методики построения правил для СОВ;
* был проведен расчет безопасности труда, который показал, что труд специалиста безопасен, а условия соответствуют санитарным нормам;
* была произведена оценка экономического эффекта от реализации формализованной методики построения правил для СОВ, которая показала, что с экономической точки зрения данная методика является более эффективной, чем построение правил для СОВ вручную.

# Список использованной литературы

1. Аграновский, А.В., Новый подход к защите информации – системы обнаружения компьютерных угроз [текст] / Доктор техн. наук, профессор А.В. Аграновский, кандидат техн. наук Р.А. Хади; ФГНУ НИИ «Спецвузавтоматика» – г.Ростов-на-Дону : Инфосистемы Джет, 2007. – 167с.
2. Атаки на DNS-сервера [электронный ресурс] / статья - Режим доступа: [http://www.tadviser.ru](http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%90%D1%82%D0%B0%D0%BA%D0%B8_%D0%BD%D0%B0_DNS-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%B0).
3. Бузуев, И.И., А-92 Оценка тяжести и напряженности трудового процесса при аттестации рабочих мест по условиям труда: лаборат. практ. [текст] / Сост.: И.И. Бузуев, И.А. Сумарченкова, Л.В. Сорокина. – Самара: Самар. гос. техн. ун-т, 2010 – 96с.
4. Генератор атак на отказ в обслуживании [электронный ресурс] / статья - Режим доступа: <http://ru.wikipedia.org/wiki/LOIC>.
5. Жилин, А.Н., Методические указания по выполнению курсовых работ на тему «Оценка химической обстановки при выбросах сильнодействующих ядовитых веществ» [текст] / Жилин А.Н., Винник В.И. – Оренбург: Издательство ОГУ, 2000г – 36с.
6. Котенко, И.В., Труды Моделирование адаптивной кооперативной защиты от компьютерных атак в сети Интернет [текст] / И.В. Котенко, А.В. Уланов // Проблемы управления рисками и безопасностью / Труды института системного анализа РАН – М. : ИСА РАН, 2007. – Т.31 – 336с.
7. Касперски, Крис, Техника Сетевых Атак [текст] / Крис Касперски – М.: СОЛОН – Р, 2001 – 400с.
8. Керниган, Брайн, Язык программирования C [текст] / Брайн Керниган, Деннис Ритчи - второе издание, пер. и доп. – М.: Вильямс, 2009 – 304с.
9. Михайлов, Е. Б., Методические указания к практическим занятиям «Оценка безопасности и тяжести труда на рабочем месте» по курсу «Безопасность жизнедеятельности» [текст] /Сост. Е. Б. Михайлов, А. С. Гузенкова. – М.: Моск. гос. ин-т электроники и математики, 2009 – 33с.
10. Норткат, Стивен, Обнаружение нарушений безопасности в сетях [текст] : [пер. с англ.] / Стивен Норткат, Джуди Новак, В.С. Иващенко – 3-е изд. – М. [и др.] : Вильямс, 2003. – 448 с.
11. Определение терминологии адаптивных систем [электронный ресурс] – Режим доступа: [http://ru.wikipedia.org/wiki/](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B0%D1%86%D0%B8%D1%8F).
12. Петренко С.А., Сравнительный анализ методов обнаружения компьютерных атак [текст] / Петренко С.А., Беляев А.В. // Проблемы информационной безопасности. Компьютерные системы : сб. науч. тр. – [Б. м.] : 2008 – с. 53.
13. Платонов, В.В., Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей : учебное пособие для студентов высших учебных заведений [текст] / В.В. Платонов – М. : Академия, 2006. – 240с.
14. Приказ ФСТЭК России № 638 от 06.12.2011 «Об утверждении требований к системам обнаружения вторжений» [электронный ресурс] – Режим доступа: <http://fstec.ru/>.
15. Проблемы DHCP [электронный ресурс] / статья; журнал Хакер – 2003 – Режим доступа: <http://www.xakep.ru/post/17284/>.
16. Прогноз по атакам на 2013 год от компании Symantec [электроны ресурс] – Режим доступа: [http://www.symantec.com/ru/](http://www.symantec.com/ru/ru/about/news/release/article.jsp?prid=20121219_01).
17. Рахматуллин Р.Р., Выполнение организационно-экономической части дипломных проектов: Методические указания. [текст] / Рахматуллин Р.Р., Тарасов В.Н., Коваленко О.В. – Оренбург : Издательство ОГУ, 1998 – 46 с.
18. Русская группа пользователей Snort [электронный ресурс] – Режим доступа: <http://www.snortgroup.ru/>.
19. Салов, А.И., Охрана труда на предприятиях [текст] / Салов А.И., Беркович Я.М., Васильева И.И. – М.: Транспорт, 1977. – 184с.
20. Самгин, Э.Б., Освещение рабочих мест [текст] / Самгин Э.Б. – М.: МИРЭА, 1989 – 186с.
21. Системы обнаружения вторжений [электронный ресурс] / Курс лекций «Инструменты безопасности с открытым исходным кодом», лекция 7 «Системы обнаружения вторжений» – Режим доступа: [http://www.intuit.ru/](http://www.intuit.ru/department/security/secopen/7/).
22. Система обнаружения вторжений [электроны ресурс] / статья – Режим доступа: [http://ru.wikipedia.org/](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BE%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D1%82%D0%BE%D1%80%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9).
23. Сканеры Уязвимостей [электронный ресурс] / статья - Режим доступа: [http://ru.wikipedia.org/wiki](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BD%D0%B5%D1%80%D1%8B_%D1%83%D1%8F%D0%B7%D0%B2%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B5%D0%B9).
24. Соловьев, Н.А., Особенности задачи распознавания классов сетевых атак [текст] / Н.А. Соловьев, Н.А. Тишина // Материалы VIII Всероссийской научно-практической конференции с международным участием «Современные информационные технологии в науке, образовании и практике» – Оренбург : Издательство ОГУ, 2009 – 590с.
25. Соловьев, Н.А., Программный комплекс идентификации аномальной активности субъектов АСУ [текст] / Н.А. Соловьев, Л.А.Юркевская, Ю.А. Азиатцев // Материалы VI Всероссийской научно-практической конференции с международным участием «Современные информационные технологии в науке, образовании и практике» – Оренбург : Издательство ОГУ, 2007 – 840с.
26. Соловьев, Н.А., Развитие концепции обнаружения вторжений [текст] / Н.А. Соловьев, Е.Н. Чернопрудова // Материалы VIII Всероссийской научно-практической конференции с международным участием «Современные информационные технологии в науке, образовании и практике» – Оренбург : Издательство ОГУ, 2009 – 590с.
27. Список коммерческих СОВ [электронный ресурс] – Режим доступа: <http://devopswiki.net/index.php/NIDS>.
28. Статья от ЗАО «Kaspersky Lab» на тему «Источники угроз сети» [электронный ресурс] - Режим доступа: <http://support.kaspersky.ru/789?el=88446>.
29. Техническая документация по продуктам и решениям компании Corero [электронный ресурс] – Режим доступа: <http://coreroplc.com/>.
30. Техническая документация по продуктам и решениям компании IBM Internet Security System [электронный ресурс] – Режим доступа: <http://xforce.iss.net/>.
31. Техническая документация по продуктам и решениям компании Kaspersky Lab [электронный ресурс] – Режим доступа: <http://www.kaspersky.ru/threats>.
32. Техническая документация по продуктам и решениям компании Symantec [электронный ресурс] – Режим доступа: [http://www.symantec.com/](http://www.symantec.com/en/uk/security_response/).
33. Техническая документация по продуктам и решениям компании Trend Micro [электронный ресурс] – Режим доступа: <http://www.trendmicro.com.ru/>.
34. Техническая документация по продуктам и решениям от производителей СОВ Snort [электронный ресурс] – Режим доступа: <https://www.snort.org/>.
35. Техническая документация по продуктам и решениям от производителей Nessus [электронный ресурс] – Режим доступа: <http://www.tenable.com/>.
36. Ушаков, Д.В., Подходы к оценке эффективности систем обнаружения вторжений [текст] // Ушаков Д.В., Милославская Н.Г. // XII Всероссийская научно-практическая конференция – М.: Издательство МИФИ, 2005 – 136с.
37. Уязвимость (Компьютерная безопасность) [электронный ресурс] / статья - Режим доступа: [http://ru.wikipedia.org/wiki/](http://ru.wikipedia.org/wiki/%D0%A3%D1%8F%D0%B7%D0%B2%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)).
38. Хакерская атака [электронный ресурс] / статья - Режим доступа: [http://ru.wikipedia.org/wiki/](http://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%BA%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%B0%D1%82%D0%B0%D0%BA%D0%B0).
39. Шевченко А.С., Исследование и разработка комплексной методики обнаружения сетевых вторжений [текст] / Шевченко А.С.; диссертация // Московский государственный институт электроники и математики (МГИЭМ) – М.: – Редакция МГИЭМ, 2007 – 174с.
40. Юдин, Е.Я., Борьба с шумом на производстве: Справочник [текст] / Е.Я. Юдин, Л.А. Борисов; Под общ. ред. Е.Я. Юдина – М.: Машиностроение, 1985. – 400с.
41. Bace, Rebecca, Intrusion Detection Systems [электронный ресурс] / Rebecca Bace, Peter Mell – Режим доступа: <http://cryptome.org/sp800-31.htm>.
42. DDoS-атаки. Обзор [электронный ресурс] / статья - Режим доступа: <http://qrator.net/solutions/>.
43. Denning, Dorothy E., An Intrusion Detection Model [текст] / Denning, Dorothy E. // Proceedings of the Seventh IEEE Symposium on Security and Privacy – [Б. м.] : IEEE, 1986 – 131c.
44. Deraison, Renaud, The Nessus Attack Scripting Language Reference Guide [электронный рессурс] / Renaud Deraison; deraison@cvs.nessus.org; Version 1.0.0pre2 – Режим доступа: <http://www.virtualblueness.net/nasl.html>.
45. Gerg, Christopher, Managing Security with Snort and IDS Tools [текст] : [англ.] / Christopher Gerg, Kerry J. Cox – USA : O'Reilly Media, 2004 – 292с.
46. DoS-атака [электронный ресурс] / статья - Режим доступа: [https://ru.wikipedia.org/](https://ru.wikipedia.org/wiki/DoS-%D0%B0%D1%82%D0%B0%D0%BA%D0%B0#.D0.9E.D1.82.D0.BF.D1.80.D0.B0.D0.B2.D0.BA.D0.B0_.C2.AB.D1.82.D1.8F.D0.B6.D0.B5.D0.BB.D1.8B.D1.85.C2.BB_.D0.BF.D0.B0.D0.BA.D0.B5.D1.82.D0.BE.D0.B2).
47. Intrusion Detection Message Exchange Format [электронный ресурс] - Режим доступа: <http://www.intuit.ru/department/security/secopen/7/>.
48. Jacob W. Ulvila, Evaluation of Intrusion Detection Systems [текст] / Jacob W. Ulvila // Journal of Research of NIST / Volume 108 / Number 6 / November-December – [Б. м.] : NIST, 2003 – 473с.
49. Lunt, Teresa F., Detecting Intruders in Computer Systems [текст] / Lunt, Teresa F. // Conference on Auditing and Computer Technology – [Б. м.] : SRI International, 1993 – 425с.
50. Paxson, Vern, Bro: A System for Detecting Network Intruders in Real-Time [текст] / Paxson, Vern // Proceedings of The 7th USENIX Security Symposium – San Antonio : TX, 1998 – 368с.
51. Mell, Peter, An Overview of Issues in Testing Intrusion Detection Systems [текст] / Peter Mell – [Б. м.] : NIST, 2003 – 483с.
52. Skrobanek, Pawel, Intrusion detection system [текст] : [англ.] / Pawel Skrobanek – Хорватия : InTech, 2011 – 334с.

# Приложение А

# Конфигурационный файл Snort

#--------------------------------------------------

# VRT Rule Packages Snort.conf

#

# For more information visit us at:

# http://www.snort.org Snort Website

# http://vrt-blog.snort.org/ Sourcefire VRT Blog

#

# Mailing list Contact: snort-sigs@lists.sourceforge.net

# False Positive reports: fp@sourcefire.com

# Snort bugs: bugs@snort.org

#

# Compatible with Snort Versions:

# VERSIONS : 2.9.4.6

#

# Snort build options:

# OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --enable-active-response --enable-normalizer --enable-reload --enable-react --enable-flexresp3

#

# Additional information:

# This configuration file enables active response, to run snort in

# test mode -T you are required to supply an interface -i <interface>

# or test mode will fail to fully validate the configuration and

# exit with a FATAL error

#--------------------------------------------------

###################################################

# This file contains a sample snort configuration.

# You should take the following steps to create your own custom configuration:

#

# 1) Set the network variables.

# 2) Configure the decoder

# 3) Configure the base detection engine

# 4) Configure dynamic loaded libraries

# 5) Configure preprocessors

# 6) Configure output plugins

# 7) Customize your rule set

# 8) Customize preprocessor and decoder rule set

# 9) Customize shared object rule set

###################################################

###################################################

# Step #1: Set the network variables. For more information, see README.variables

###################################################

# Setup the network addresses you are protecting

ipvar HOME\_NET any

# Set up the external network addresses. Leave as "any" in most situations

ipvar EXTERNAL\_NET any

# List of DNS servers on your network

ipvar DNS\_SERVERS $HOME\_NET

# List of SMTP servers on your network

ipvar SMTP\_SERVERS $HOME\_NET

# List of web servers on your network

ipvar HTTP\_SERVERS $HOME\_NET

# List of sql servers on your network

ipvar SQL\_SERVERS $HOME\_NET

# List of telnet servers on your network

ipvar TELNET\_SERVERS $HOME\_NET

# List of ssh servers on your network

ipvar SSH\_SERVERS $HOME\_NET

# List of ftp servers on your network

ipvar FTP\_SERVERS $HOME\_NET

# List of sip servers on your network

ipvar SIP\_SERVERS $HOME\_NET

# List of ports you run web servers on

portvar HTTP\_PORTS [80,81,82,83,84,85,86,87,88,89,311,383,591,593,631,901,1220,1414,1741,1830,2301,2381,2809,3037,3057,3128,3702,4343,4848,5250,6080,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8088,8090,8118,8123,8180,8181,8222,8243,8280,8300,8800,8888,8899,9000,9060,9080,9090,9091,9443,9999,11371,34443,34444,41080,50002,55555]

# List of ports you want to look for SHELLCODE on.

portvar SHELLCODE\_PORTS !80

# List of ports you might see oracle attacks on

portvar ORACLE\_PORTS 1024:

# List of ports you want to look for SSH connections on:

portvar SSH\_PORTS 22

# List of ports you run ftp servers on

portvar FTP\_PORTS [21,2100,3535]

# List of ports you run SIP servers on

portvar SIP\_PORTS [5060,5061,5600]

# List of file data ports for file inspection

portvar FILE\_DATA\_PORTS [$HTTP\_PORTS,110,143]

# List of GTP ports for GTP preprocessor

portvar GTP\_PORTS [2123,2152,3386]

# other variables, these should not be modified

ipvar AIM\_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)

# Note for Windows users: You are advised to make this an absolute path,

# such as: c:\snort\rules

var RULE\_PATH ../rules

var SO\_RULE\_PATH ../so\_rules

var PREPROC\_RULE\_PATH ../preproc\_rules

# If you are using reputation preprocessor set these

# Currently there is a bug with relative paths, they are relative to where snort is

# not relative to snort.conf like the above variables

# This is completely inconsistent with how other vars work, BUG 89986

# Set the absolute path appropriately

var WHITE\_LIST\_PATH ../rules

var BLACK\_LIST\_PATH ../rules

###################################################

# Step #2: Configure the decoder. For more information, see README.decode

###################################################

# Stop generic decode events:

config disable\_decode\_alerts

# Stop Alerts on experimental TCP options

config disable\_tcpopt\_experimental\_alerts

# Stop Alerts on obsolete TCP options

config disable\_tcpopt\_obsolete\_alerts

# Stop Alerts on T/TCP alerts

config disable\_tcpopt\_ttcp\_alerts

# Stop Alerts on all other TCPOption type events:

config disable\_tcpopt\_alerts

# Stop Alerts on invalid ip options

config disable\_ipopt\_alerts

# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet

# config enable\_decode\_oversized\_alerts

# Same as above, but drop packet if in Inline mode (requires enable\_decode\_oversized\_alerts)

# config enable\_decode\_oversized\_drops

# Configure IP / TCP checksum mode

config checksum\_mode: all

# Configure maximum number of flowbit references. For more information, see README.flowbits

# config flowbits\_size: 64

# Configure ports to ignore

# config ignore\_ports: tcp 21 6667:6671 1356

# config ignore\_ports: udp 1:17 53

# Configure active response for non inline operation. For more information, see REAMDE.active

# config response: eth0 attempts 2

# Configure DAQ related options for inline operation. For more information, see README.daq

#

# config daq: <type>

# config daq\_dir: <dir>

# config daq\_mode: <mode>

# config daq\_var: <var>

#

# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw

# <mode> ::= read-file | passive | inline

# <var> ::= arbitrary <name>=<value passed to DAQ

# <dir> ::= path as to where to look for DAQ module so's

# Configure specific UID and GID to run snort as after dropping privs. For more information see snort -h command line options

#

# config set\_gid:

# config set\_uid:

# Configure default snaplen. Snort defaults to MTU of in use interface. For more information see README

#

# config snaplen:

#

# Configure default bpf\_file to use for filtering what traffic reaches snort. For more information see snort -h command line options (-F)

#

# config bpf\_file:

#

# Configure default log directory for snort to log to. For more information see snort -h command line options (-l)

#

# config logdir:

###################################################

# Step #3: Configure the base detection engine. For more information, see README.decode

###################################################

# Configure PCRE match limitations

config pcre\_match\_limit: 3500

config pcre\_match\_limit\_recursion: 1500

# Configure the detection engine See the Snort Manual, Configuring Snort - Includes - Config

config detection: search-method ac-split search-optimize max-pattern-len 20

# Configure the event queue. For more information, see README.event\_queue

config event\_queue: max\_queue 8 log 5 order\_events content\_length

###################################################

## Configure GTP if it is to be used.

## For more information, see README.GTP

####################################################

# config enable\_gtp

###################################################

# Per packet and rule latency enforcement

# For more information see README.ppm

###################################################

# Per Packet latency configuration

#config ppm: max-pkt-time 250, \

# fastpath-expensive-packets, \

# pkt-log

# Per Rule latency configuration

#config ppm: max-rule-time 200, \

# threshold 3, \

# suspend-expensive-rules, \

# suspend-timeout 20, \

# rule-log alert

###################################################

# Configure Perf Profiling for debugging

# For more information see README.PerfProfiling

###################################################

#config profile\_rules: print all, sort avg\_ticks

#config profile\_preprocs: print all, sort avg\_ticks

###################################################

# Configure protocol aware flushing

# For more information see README.stream5

###################################################

config paf\_max: 16000

###################################################

# Step #4: Configure dynamic loaded libraries.

# For more information, see Snort Manual, Configuring Snort - Dynamic Modules

###################################################

# path to dynamic preprocessor libraries

dynamicpreprocessor directory /usr/local/lib/snort\_dynamicpreprocessor/

# path to base preprocessor engine

dynamicengine /usr/local/lib/snort\_dynamicengine/libsf\_engine.so

# path to dynamic rules libraries

dynamicdetection directory /usr/local/lib/snort\_dynamicrules

###################################################

# Step #5: Configure preprocessors

# For more information, see the Snort Manual, Configuring Snort - Preprocessors

###################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP

# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize

# Does nothing in IDS mode

preprocessor normalize\_ip4

preprocessor normalize\_tcp: ips ecn stream

preprocessor normalize\_icmp4

preprocessor normalize\_ip6

preprocessor normalize\_icmp6

# Target-based IP defragmentation. For more inforation, see README.frag3

preprocessor frag3\_global: max\_frags 65536

preprocessor frag3\_engine: policy windows detect\_anomalies overlap\_limit 10 min\_fragment\_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly. For more inforation, see README.stream5

preprocessor stream5\_global: track\_tcp yes, \

track\_udp yes, \

track\_icmp no, \

max\_tcp 262144, \

max\_udp 131072, \

max\_active\_responses 2, \

min\_response\_seconds 5

preprocessor stream5\_tcp: policy windows, detect\_anomalies, require\_3whs 180, \

overlap\_limit 10, small\_segments 3 bytes 150, timeout 180, \

ports client 21 22 23 25 42 53 70 79 109 110 111 113 119 135 136 137 139 143 \

161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070 6665 6666 6667 6668 6669 \

7000 8181 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \

ports both 80 81 82 83 84 85 86 87 88 89 110 311 383 443 465 563 591 593 631 636 901 989 992 993 994 995 1220 1414 1830 2301 2381 2809 3037 3057 3128 3702 4343 4848 5250 6080 6988 7907 7000 7001 7144 7145 7510 7802 7777 7779 \

7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914 7915 7916 \

7917 7918 7919 7920 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180 8222 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 11371 34443 34444 41080 50002 55555

preprocessor stream5\_udp: timeout 180

# performance statistics. For more information, see the Snort Manual, Configuring Snort - Preprocessors - Performance Monitor

# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

# HTTP normalization and anomaly detection. For more information, see README.http\_inspect

preprocessor http\_inspect: global iis\_unicode\_map unicode.map 1252 compress\_depth 65535 decompress\_depth 65535

preprocessor http\_inspect\_server: server default \

http\_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC\_CONNECT PROXY\_SUCCESS BITS\_POST CCM\_POST SMS\_POST RPC\_IN\_DATA RPC\_OUT\_DATA RPC\_ECHO\_DATA } \

chunk\_length 500000 \

server\_flow\_depth 0 \

client\_flow\_depth 0 \

post\_depth 65495 \

oversize\_dir\_length 500 \

max\_header\_length 750 \

max\_headers 100 \

max\_spaces 200 \

small\_chunk\_length { 10 5 } \

ports { 80 81 82 83 84 85 86 87 88 89 311 383 591 593 631 901 1220 1414 1741 1830 2301 2381 2809 3037 3057 3128 3702 4343 4848 5250 6080 6988 7000 7001 7144 7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180 8181 8222 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 11371 34443 34444 41080 50002 55555 } \

non\_rfc\_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \

enable\_cookie \

extended\_response\_inspection \

inspect\_gzip \

normalize\_utf \

unlimited\_decompress \

normalize\_javascript \

apache\_whitespace no \

ascii no \

bare\_byte no \

directory no \

double\_decode no \

iis\_backslash no \

iis\_delimiter no \

iis\_unicode no \

multi\_slash no \

utf\_8 no \

u\_encode yes \

webroot no

# ONC-RPC normalization and anomaly detection. For more information, see the Snort Manual, Configuring Snort - Preprocessors - RPC Decode

preprocessor rpc\_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no\_alert\_multiple\_requests no\_alert\_large\_fragments no\_alert\_incomplete

# Back Orifice detection.

preprocessor bo

# FTP / Telnet normalization and anomaly detection. For more information, see README.ftptelnet

preprocessor ftp\_telnet: global inspection\_type stateful encrypted\_traffic no check\_encrypted

preprocessor ftp\_telnet\_protocol: telnet \

ayt\_attack\_thresh 20 \

normalize ports { 23 } \

detect\_anomalies

preprocessor ftp\_telnet\_protocol: ftp server default \

def\_max\_param\_len 100 \

ports { 21 2100 3535 } \

telnet\_cmds yes \

ignore\_telnet\_erase\_cmds yes \

ftp\_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \

ftp\_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \

ftp\_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \

ftp\_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \

ftp\_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \

ftp\_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \

ftp\_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \

ftp\_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \

ftp\_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \

ftp\_cmds { XSEN XSHA1 XSHA256 } \

alt\_max\_param\_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP PASV PWD QUIT REIN STOU SYST XCUP XPWD } \

alt\_max\_param\_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR STOU XMKD } \

alt\_max\_param\_len 256 { CWD RNTO } \

alt\_max\_param\_len 400 { PORT } \

alt\_max\_param\_len 512 { SIZE } \

chk\_str\_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \

chk\_str\_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \

chk\_str\_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \

chk\_str\_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \

chk\_str\_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \

chk\_str\_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \

chk\_str\_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \

chk\_str\_fmt { XSEM XSEN XSHA1 XSHA256 } \

cmd\_validity ALLO < int [ char R int ] > \

cmd\_validity EPSV < [ { char 12 | char A char L char L } ] > \

cmd\_validity MACB < string > \

cmd\_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \

cmd\_validity MODE < char ASBCZ > \

cmd\_validity PORT < host\_port > \

cmd\_validity PROT < char CSEP > \

cmd\_validity STRU < char FRPO [ string ] > \

cmd\_validity TYPE < { char AE [ char NTC ] | char I | char L [ number ] } >

preprocessor ftp\_telnet\_protocol: ftp client default \

max\_resp\_len 256 \

bounce yes \

ignore\_telnet\_erase\_cmds yes \

telnet\_cmds yes

# SMTP normalization and anomaly detection. For more information, see README.SMTP

preprocessor smtp: ports { 25 465 587 691 } \

inspection\_type stateful \

b64\_decode\_depth 0 \

qp\_decode\_depth 0 \

bitenc\_decode\_depth 0 \

uu\_decode\_depth 0 \

log\_mailfrom \

log\_rcptto \

log\_filename \

log\_email\_hdrs \

normalize cmds \

normalize\_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY } \

normalize\_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SOML } \

normalize\_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-DRCP X-ERCP X-EXCH50 } \

normalize\_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \

max\_command\_line\_len 512 \

max\_header\_line\_len 1000 \

max\_response\_line\_len 512 \

alt\_max\_command\_line\_len 260 { MAIL } \

alt\_max\_command\_line\_len 300 { RCPT } \

alt\_max\_command\_line\_len 500 { HELP HELO ETRN EHLO } \

alt\_max\_command\_line\_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \

alt\_max\_command\_line\_len 246 { SEND SAML SOML AUTH TURN ETRN DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \

valid\_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY } \

valid\_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SOML } \

valid\_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-DRCP X-ERCP X-EXCH50 } \

valid\_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \

xlink2state { enabled }

# Portscan detection. For more information, see README.sfportscan

# preprocessor sfportscan: proto { all } memcap { 10000000 } sense\_level { low }

# ARP spoof detection. For more information, see the Snort Manual - Configuring Snort - Preprocessors - ARP Spoof Preprocessor

# preprocessor arpspoof

# preprocessor arpspoof\_detect\_host: 192.168.40.1 f0:0f:00:f0:0f:00

# SSH anomaly detection. For more information, see README.ssh

preprocessor ssh: server\_ports { 22 } \

autodetect \

max\_client\_bytes 19600 \

max\_encrypted\_packets 20 \

max\_server\_version\_len 100 \

enable\_respoverflow enable\_ssh1crc32 \

enable\_srvoverflow enable\_protomismatch

# SMB / DCE-RPC normalization and anomaly detection. For more information, see README.dcerpc2

preprocessor dcerpc2: memcap 102400, events [co ]

preprocessor dcerpc2\_server: default, policy WinXP, \

detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \

autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \

smb\_max\_chain 3, smb\_invalid\_shares ["C$", "D$", "ADMIN$"]

# DNS anomaly detection. For more information, see README.dns

preprocessor dns: ports { 53 } enable\_rdata\_overflow

# SSL anomaly detection and traffic bypass. For more information, see README.ssl

preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 7801 7802 7900 7901 7902 7903 7904 7905 7906 7907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918 7919 7920 }, trustservers, noinspect\_encrypted

# SDF sensitive data preprocessor. For more information see README.sensitive\_data

preprocessor sensitive\_data: alert\_threshold 25

# SIP Session Initiation Protocol preprocessor. For more information see README.sip

preprocessor sip: max\_sessions 40000, \

ports { 5060 5061 5600 }, \

methods { invite \

cancel \

ack \

bye \

register \

options \

refer \

subscribe \

update \

join \

info \

message \

notify \

benotify \

do \

qauth \

sprack \

publish \

service \

unsubscribe \

prack }, \

max\_uri\_len 512, \

max\_call\_id\_len 80, \

max\_requestName\_len 20, \

max\_from\_len 256, \

max\_to\_len 256, \

max\_via\_len 1024, \

max\_contact\_len 512, \

max\_content\_len 2048

# IMAP preprocessor. For more information see README.imap

preprocessor imap: \

ports { 143 } \

b64\_decode\_depth 0 \

qp\_decode\_depth 0 \

bitenc\_decode\_depth 0 \

uu\_decode\_depth 0

# POP preprocessor. For more information see README.pop

preprocessor pop: \

ports { 110 } \

b64\_decode\_depth 0 \

qp\_decode\_depth 0 \

bitenc\_decode\_depth 0 \

uu\_decode\_depth 0

# Modbus preprocessor. For more information see README.modbus

preprocessor modbus: ports { 502 }

# DNP3 preprocessor. For more information see README.dnp3

preprocessor dnp3: ports { 20000 } \

memcap 262144 \

check\_crc

# Reputation preprocessor. For more information see README.reputation

preprocessor reputation: \

memcap 500, \

priority whitelist, \

nested\_ip inner, \

whitelist $WHITE\_LIST\_PATH/white\_list.rules, \

blacklist $BLACK\_LIST\_PATH/black\_list.rules

###################################################

# Step #6: Configure output plugins

# For more information, see Snort Manual, Configuring Snort - Output Modules

###################################################

# unified2

# Recommended for most installs

# output unified2: filename merged.log, limit 128, nostamp, mpls\_event\_types, vlan\_event\_types

# Additional configuration for specific types of installs

# output alert\_unified2: filename snort.alert, limit 128, nostamp

# output log\_unified2: filename snort.log, limit 128, nostamp

# syslog

# output alert\_syslog: LOG\_AUTH LOG\_ALERT

# pcap

# output log\_tcpdump: tcpdump.log

# metadata reference data. do not modify these lines

include classification.config

include reference.config

###################################################

# Step #7: Customize your rule set

# For more information, see Snort Manual, Writing Snort Rules

#

# NOTE: All categories are enabled in this conf file

###################################################

# site specific rules

include $RULE\_PATH/local.rules

include $RULE\_PATH/app-detect.rules

include $RULE\_PATH/attack-responses.rules

include $RULE\_PATH/backdoor.rules

include $RULE\_PATH/bad-traffic.rules

include $RULE\_PATH/blacklist.rules

include $RULE\_PATH/botnet-cnc.rules

include $RULE\_PATH/browser-chrome.rules

include $RULE\_PATH/browser-firefox.rules

include $RULE\_PATH/browser-ie.rules

include $RULE\_PATH/browser-other.rules

include $RULE\_PATH/browser-plugins.rules

include $RULE\_PATH/browser-webkit.rules

include $RULE\_PATH/chat.rules

include $RULE\_PATH/content-replace.rules

include $RULE\_PATH/ddos.rules

include $RULE\_PATH/dns.rules

include $RULE\_PATH/dos.rules

include $RULE\_PATH/experimental.rules

include $RULE\_PATH/exploit-kit.rules

include $RULE\_PATH/exploit.rules

include $RULE\_PATH/file-executable.rules

include $RULE\_PATH/file-flash.rules

include $RULE\_PATH/file-identify.rules

include $RULE\_PATH/file-image.rules

include $RULE\_PATH/file-multimedia.rules

include $RULE\_PATH/file-office.rules

include $RULE\_PATH/file-other.rules

include $RULE\_PATH/file-pdf.rules

include $RULE\_PATH/finger.rules

include $RULE\_PATH/ftp.rules

include $RULE\_PATH/icmp-info.rules

include $RULE\_PATH/icmp.rules

include $RULE\_PATH/imap.rules

include $RULE\_PATH/indicator-compromise.rules

include $RULE\_PATH/indicator-obfuscation.rules

include $RULE\_PATH/indicator-shellcode.rules

include $RULE\_PATH/info.rules

include $RULE\_PATH/malware-backdoor.rules

include $RULE\_PATH/malware-cnc.rules

include $RULE\_PATH/malware-other.rules

include $RULE\_PATH/malware-tools.rules

include $RULE\_PATH/misc.rules

include $RULE\_PATH/multimedia.rules

include $RULE\_PATH/mysql.rules

include $RULE\_PATH/netbios.rules

include $RULE\_PATH/nntp.rules

include $RULE\_PATH/oracle.rules

include $RULE\_PATH/os-linux.rules

include $RULE\_PATH/os-other.rules

include $RULE\_PATH/os-solaris.rules

include $RULE\_PATH/os-windows.rules

include $RULE\_PATH/other-ids.rules

include $RULE\_PATH/p2p.rules

include $RULE\_PATH/phishing-spam.rules

include $RULE\_PATH/policy-multimedia.rules

include $RULE\_PATH/policy-other.rules

include $RULE\_PATH/policy.rules

include $RULE\_PATH/policy-social.rules

include $RULE\_PATH/policy-spam.rules

include $RULE\_PATH/pop2.rules

include $RULE\_PATH/pop3.rules

include $RULE\_PATH/protocol-finger.rules

include $RULE\_PATH/protocol-ftp.rules

include $RULE\_PATH/protocol-icmp.rules

include $RULE\_PATH/protocol-imap.rules

include $RULE\_PATH/protocol-pop.rules

include $RULE\_PATH/protocol-services.rules

include $RULE\_PATH/protocol-voip.rules

include $RULE\_PATH/pua-adware.rules

include $RULE\_PATH/pua-other.rules

include $RULE\_PATH/pua-p2p.rules

include $RULE\_PATH/pua-toolbars.rules

include $RULE\_PATH/rpc.rules

include $RULE\_PATH/rservices.rules

include $RULE\_PATH/scada.rules

include $RULE\_PATH/scan.rules

include $RULE\_PATH/server-apache.rules

include $RULE\_PATH/server-iis.rules

include $RULE\_PATH/server-mail.rules

include $RULE\_PATH/server-mssql.rules

include $RULE\_PATH/server-mysql.rules

include $RULE\_PATH/server-oracle.rules

include $RULE\_PATH/server-other.rules

include $RULE\_PATH/server-webapp.rules

include $RULE\_PATH/shellcode.rules

include $RULE\_PATH/smtp.rules

include $RULE\_PATH/snmp.rules

include $RULE\_PATH/specific-threats.rules

include $RULE\_PATH/spyware-put.rules

include $RULE\_PATH/sql.rules

include $RULE\_PATH/telnet.rules

include $RULE\_PATH/tftp.rules

include $RULE\_PATH/virus.rules

include $RULE\_PATH/voip.rules

include $RULE\_PATH/web-activex.rules

include $RULE\_PATH/web-attacks.rules

include $RULE\_PATH/web-cgi.rules

include $RULE\_PATH/web-client.rules

include $RULE\_PATH/web-coldfusion.rules

include $RULE\_PATH/web-frontpage.rules

include $RULE\_PATH/web-iis.rules

include $RULE\_PATH/web-misc.rules

include $RULE\_PATH/web-php.rules

include $RULE\_PATH/x11.rules

###################################################

# Step #8: Customize your preprocessor and decoder alerts

# For more information, see README.decoder\_preproc\_rules

###################################################

# decoder and preprocessor event rules

# include $PREPROC\_RULE\_PATH/preprocessor.rules

# include $PREPROC\_RULE\_PATH/decoder.rules

# include $PREPROC\_RULE\_PATH/sensitive-data.rules

###################################################

# Step #9: Customize your Shared Object Snort Rules

# For more information, see http://vrt-blog.snort.org/2009/01/using-vrt-certified-shared-object-rules.html

###################################################

# dynamic library rules

# include $SO\_RULE\_PATH/bad-traffic.rules

# include $SO\_RULE\_PATH/chat.rules

# include $SO\_RULE\_PATH/dos.rules

# include $SO\_RULE\_PATH/exploit.rules

# include $SO\_RULE\_PATH/icmp.rules

# include $SO\_RULE\_PATH/imap.rules

# include $SO\_RULE\_PATH/misc.rules

# include $SO\_RULE\_PATH/multimedia.rules

# include $SO\_RULE\_PATH/netbios.rules

# include $SO\_RULE\_PATH/nntp.rules

# include $SO\_RULE\_PATH/p2p.rules

# include $SO\_RULE\_PATH/smtp.rules

# include $SO\_RULE\_PATH/snmp.rules

# include $SO\_RULE\_PATH/specific-threats.rules

# include $SO\_RULE\_PATH/web-activex.rules

# include $SO\_RULE\_PATH/web-client.rules

# include $SO\_RULE\_PATH/web-iis.rules

# include $SO\_RULE\_PATH/web-misc.rules

# Event thresholding or suppression commands. See threshold.conf

include threshold.conf

# Приложение Б

# Конфигурационный файл программы

This file contains any messages produced by compilers while

running configure, to aid debugging if configure makes a mistake.

It was created by gtk+ configure 2.10.9, which was

generated by GNU Autoconf 2.60. Invocation command line was

$ ./configure

## --------- ##

## Platform. ##

## --------- ##

hostname = afro-PC

uname -m = i686

uname -r = 3.2.0-44-generic-pae

uname -s = Linux

uname -v = #69-Ubuntu SMP Thu May 16 18:50:07 UTC 2013

/usr/bin/uname -p = unknown

/bin/uname -X = unknown

/bin/arch = unknown

/usr/bin/arch -k = unknown

/usr/convex/getsysinfo = unknown

/usr/bin/hostinfo = unknown

/bin/machine = unknown

/usr/bin/oslevel = unknown

/bin/universe = unknown

PATH: /usr/lib/lightdm/lightdm

PATH: /usr/local/sbin

PATH: /usr/local/bin

PATH: /usr/sbin

PATH: /usr/bin

PATH: /sbin

PATH: /bin

PATH: /usr/games

## ----------- ##

## Core tests. ##

## ----------- ##

configure:2167: checking for a BSD-compatible install

configure:2223: result: /usr/bin/install -c

configure:2234: checking whether build environment is sane

configure:2277: result: yes

configure:2310: checking for gawk

configure:2340: result: no

configure:2310: checking for mawk

configure:2326: found /usr/bin/mawk

configure:2337: result: mawk

configure:2348: checking whether make sets $(MAKE)

configure:2369: result: yes

configure:2614: checking build system type

configure:2632: result: i686-pc-linux-gnu

configure:2654: checking host system type

configure:2669: result: i686-pc-linux-gnu

configure:2693: checking for native Win32

configure:2704: result: no

configure:2760: checking for gcc

configure:2776: found /usr/bin/gcc

configure:2787: result: gcc

configure:3025: checking for C compiler version

configure:3032: gcc --version >&5

gcc (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3

Copyright (C) 2011 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

configure:3035: $? = 0

configure:3042: gcc -v >&5

Using built-in specs.

COLLECT\_GCC=gcc

COLLECT\_LTO\_WRAPPER=/usr/lib/gcc/i686-linux-gnu/4.6/lto-wrapper

Target: i686-linux-gnu

Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.6.3-1ubuntu5' --with-bugurl=file:///usr/share/doc/gcc-4.6/README.Bugs --enable-languages=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.6 --enable-shared --enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.6 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --enable-plugin --enable-objc-gc --enable-targets=all --disable-werror --with-arch-32=i686 --with-tune=generic --enable-checking=release --build=i686-linux-gnu --host=i686-linux-gnu --target=i686-linux-gnu

Thread model: posix

gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)

configure:3045: $? = 0

configure:3052: gcc -V >&5

gcc: error: unrecognized option '-V'

gcc: fatal error: no input files

compilation terminated.

configure:3055: $? = 4

configure:3078: checking for C compiler default output file name

configure:3105: gcc conftest.c >&5

configure:3108: $? = 0

configure:3154: result: a.out

configure:3159: checking whether the C compiler works

configure:3169: ./a.out

configure:3172: $? = 0

configure:3189: result: yes

configure:3196: checking whether we are cross compiling

configure:3198: result: no

configure:3201: checking for suffix of executables

configure:3208: gcc -o conftest conftest.c >&5

configure:3211: $? = 0

configure:3235: result:

configure:3241: checking for suffix of object files

configure:3267: gcc -c conftest.c >&5

configure:3270: $? = 0

configure:3293: result: o

configure:3297: checking whether we are using the GNU C compiler

configure:3326: gcc -c conftest.c >&5

configure:3332: $? = 0

configure:3339: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:3342: $? = 0

configure:3349: test -s conftest.o

configure:3352: $? = 0

configure:3366: result: yes

configure:3371: checking whether gcc accepts -g

configure:3401: gcc -c -g conftest.c >&5

configure:3407: $? = 0

configure:3414: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:3417: $? = 0

configure:3424: test -s conftest.o

configure:3427: $? = 0

configure:3557: result: yes

configure:3574: checking for gcc option to accept ISO C89

configure:3648: gcc -c -g -O2 conftest.c >&5

configure:3654: $? = 0

configure:3661: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:3664: $? = 0

configure:3671: test -s conftest.o

configure:3674: $? = 0

configure:3694: result: none needed

configure:3723: checking for style of include used by make

configure:3751: result: GNU

configure:3779: checking dependency style of gcc

configure:3862: result: gcc3

configure:3968: checking for a sed that does not truncate output

configure:4022: result: /bin/sed

configure:4025: checking for grep that handles long lines and -e

configure:4099: result: /bin/grep

configure:4104: checking for egrep

configure:4182: result: /bin/grep -E

configure:4198: checking for ld used by gcc

configure:4265: result: /usr/bin/ld

configure:4274: checking if the linker (/usr/bin/ld) is GNU ld

configure:4289: result: yes

configure:4294: checking for /usr/bin/ld option to reload object files

configure:4301: result: -r

configure:4319: checking for BSD-compatible nm

configure:4368: result: /usr/bin/nm -B

configure:4372: checking whether ln -s works

configure:4376: result: yes

configure:4383: checking how to recognise dependent libraries

configure:4559: result: pass\_all

configure:5099: checking how to run the C preprocessor

configure:5139: gcc -E conftest.c

configure:5145: $? = 0

configure:5183: gcc -E conftest.c

conftest.c:9:28: fatal error: ac\_nonexistent.h: No such file or directory

compilation terminated.

configure:5189: $? = 1

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| /\* end confdefs.h. \*/

| #include <ac\_nonexistent.h>

configure:5229: result: gcc -E

configure:5258: gcc -E conftest.c

configure:5264: $? = 0

configure:5302: gcc -E conftest.c

conftest.c:9:28: fatal error: ac\_nonexistent.h: No such file or directory

compilation terminated.

configure:5308: $? = 1

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| /\* end confdefs.h. \*/

| #include <ac\_nonexistent.h>

configure:5353: checking for ANSI C header files

configure:5383: gcc -c -g -O2 conftest.c >&5

configure:5389: $? = 0

configure:5396: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5399: $? = 0

configure:5406: test -s conftest.o

configure:5409: $? = 0

configure:5505: gcc -o conftest -g -O2 conftest.c >&5

configure:5508: $? = 0

configure:5514: ./conftest

configure:5517: $? = 0

configure:5534: result: yes

configure:5558: checking for sys/types.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for sys/stat.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for stdlib.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for string.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for memory.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for strings.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for inttypes.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for stdint.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5558: checking for unistd.h

configure:5579: gcc -c -g -O2 conftest.c >&5

configure:5585: $? = 0

configure:5592: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5595: $? = 0

configure:5602: test -s conftest.o

configure:5605: $? = 0

configure:5618: result: yes

configure:5645: checking dlfcn.h usability

configure:5662: gcc -c -g -O2 conftest.c >&5

configure:5668: $? = 0

configure:5675: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:5678: $? = 0

configure:5685: test -s conftest.o

configure:5688: $? = 0

configure:5699: result: yes

configure:5703: checking dlfcn.h presence

configure:5718: gcc -E conftest.c

configure:5724: $? = 0

configure:5745: result: yes

configure:5778: checking for dlfcn.h

configure:5786: result: yes

configure:5857: checking for g++

configure:5887: result: no

configure:5857: checking for c++

configure:5887: result: no

configure:5857: checking for gpp

configure:5887: result: no

configure:5857: checking for aCC

configure:5887: result: no

configure:5857: checking for CC

configure:5887: result: no

configure:5857: checking for cxx

configure:5887: result: no

configure:5857: checking for cc++

configure:5887: result: no

configure:5857: checking for cl.exe

configure:5887: result: no

configure:5857: checking for FCC

configure:5887: result: no

configure:5857: checking for KCC

configure:5887: result: no

configure:5857: checking for RCC

configure:5887: result: no

configure:5857: checking for xlC\_r

configure:5887: result: no

configure:5857: checking for xlC

configure:5887: result: no

configure:5915: checking for C++ compiler version

configure:5922: g++ --version >&5

./configure: line 5923: g++: command not found

configure:5925: $? = 127

configure:5932: g++ -v >&5

./configure: line 5933: g++: command not found

configure:5935: $? = 127

configure:5942: g++ -V >&5

./configure: line 5943: g++: command not found

configure:5945: $? = 127

configure:5948: checking whether we are using the GNU C++ compiler

configure:5977: g++ -c conftest.cpp >&5

./configure: line 5978: g++: command not found

configure:5983: $? = 127

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| #define STDC\_HEADERS 1

| #define HAVE\_SYS\_TYPES\_H 1

| #define HAVE\_SYS\_STAT\_H 1

| #define HAVE\_STDLIB\_H 1

| #define HAVE\_STRING\_H 1

| #define HAVE\_MEMORY\_H 1

| #define HAVE\_STRINGS\_H 1

| #define HAVE\_INTTYPES\_H 1

| #define HAVE\_STDINT\_H 1

| #define HAVE\_UNISTD\_H 1

| #define HAVE\_DLFCN\_H 1

| /\* end confdefs.h. \*/

|

| int

| main ()

| {

| #ifndef \_\_GNUC\_\_

| choke me

| #endif

|

| ;

| return 0;

| }

configure:6017: result: no

configure:6022: checking whether g++ accepts -g

configure:6052: g++ -c -g conftest.cpp >&5

./configure: line 6053: g++: command not found

configure:6058: $? = 127

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| #define STDC\_HEADERS 1

| #define HAVE\_SYS\_TYPES\_H 1

| #define HAVE\_SYS\_STAT\_H 1

| #define HAVE\_STDLIB\_H 1

| #define HAVE\_STRING\_H 1

| #define HAVE\_MEMORY\_H 1

| #define HAVE\_STRINGS\_H 1

| #define HAVE\_INTTYPES\_H 1

| #define HAVE\_STDINT\_H 1

| #define HAVE\_UNISTD\_H 1

| #define HAVE\_DLFCN\_H 1

| /\* end confdefs.h. \*/

|

| int

| main ()

| {

|

| ;

| return 0;

| }

configure:6107: g++ -c conftest.cpp >&5

./configure: line 6108: g++: command not found

configure:6113: $? = 127

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| #define STDC\_HEADERS 1

| #define HAVE\_SYS\_TYPES\_H 1

| #define HAVE\_SYS\_STAT\_H 1

| #define HAVE\_STDLIB\_H 1

| #define HAVE\_STRING\_H 1

| #define HAVE\_MEMORY\_H 1

| #define HAVE\_STRINGS\_H 1

| #define HAVE\_INTTYPES\_H 1

| #define HAVE\_STDINT\_H 1

| #define HAVE\_UNISTD\_H 1

| #define HAVE\_DLFCN\_H 1

| /\* end confdefs.h. \*/

|

| int

| main ()

| {

|

| ;

| return 0;

| }

configure:6163: g++ -c -g conftest.cpp >&5

./configure: line 6164: g++: command not found

configure:6169: $? = 127

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| #define STDC\_HEADERS 1

| #define HAVE\_SYS\_TYPES\_H 1

| #define HAVE\_SYS\_STAT\_H 1

| #define HAVE\_STDLIB\_H 1

| #define HAVE\_STRING\_H 1

| #define HAVE\_MEMORY\_H 1

| #define HAVE\_STRINGS\_H 1

| #define HAVE\_INTTYPES\_H 1

| #define HAVE\_STDINT\_H 1

| #define HAVE\_UNISTD\_H 1

| #define HAVE\_DLFCN\_H 1

| /\* end confdefs.h. \*/

|

| int

| main ()

| {

|

| ;

| return 0;

| }

configure:6208: result: no

configure:6233: checking dependency style of g++

configure:6316: result: none

configure:6648: checking for g77

configure:6678: result: no

configure:6648: checking for f77

configure:6678: result: no

configure:6648: checking for xlf

configure:6678: result: no

configure:6648: checking for frt

configure:6678: result: no

configure:6648: checking for pgf77

configure:6678: result: no

configure:6648: checking for cf77

configure:6678: result: no

configure:6648: checking for fort77

configure:6678: result: no

configure:6648: checking for fl32

configure:6678: result: no

configure:6648: checking for af77

configure:6678: result: no

configure:6648: checking for f90

configure:6678: result: no

configure:6648: checking for xlf90

configure:6678: result: no

configure:6648: checking for pgf90

configure:6678: result: no

configure:6648: checking for pghpf

configure:6678: result: no

configure:6648: checking for epcf90

configure:6678: result: no

configure:6648: checking for gfortran

configure:6678: result: no

configure:6648: checking for g95

configure:6678: result: no

configure:6648: checking for f95

configure:6678: result: no

configure:6648: checking for fort

configure:6678: result: no

configure:6648: checking for xlf95

configure:6678: result: no

configure:6648: checking for ifort

configure:6678: result: no

configure:6648: checking for ifc

configure:6678: result: no

configure:6648: checking for efc

configure:6678: result: no

configure:6648: checking for pgf95

configure:6678: result: no

configure:6648: checking for lf95

configure:6678: result: no

configure:6648: checking for ftn

configure:6678: result: no

configure:6705: checking for Fortran 77 compiler version

configure:6712: --version >&5

./configure: line 6713: --version: command not found

configure:6715: $? = 127

configure:6722: -v >&5

./configure: line 6723: -v: command not found

configure:6725: $? = 127

configure:6732: -V >&5

./configure: line 6733: -V: command not found

configure:6735: $? = 127

configure:6743: checking whether we are using the GNU Fortran 77 compiler

configure:6762: -c conftest.F >&5

./configure: line 6763: -c: command not found

configure:6768: $? = 127

configure: failed program was:

| program main

| #ifndef \_\_GNUC\_\_

| choke me

| #endif

|

| end

configure:6802: result: no

configure:6808: checking whether accepts -g

configure:6825: -c -g conftest.f >&5

./configure: line 6826: -c: command not found

configure:6831: $? = 127

configure: failed program was:

| program main

|

| end

configure:6864: result: no

configure:6894: checking the maximum length of command line arguments

configure:7003: result: 32768

configure:7014: checking command to parse /usr/bin/nm -B output from gcc object

configure:7119: gcc -c -g -O2 conftest.c >&5

configure:7122: $? = 0

configure:7126: /usr/bin/nm -B conftest.o \| sed -n -e 's/^.\*[ ]\([ABCDGIRSTW][ABCDGIRSTW]\*\)[ ][ ]\*\([\_A-Za-z][\_A-Za-z0-9]\*\)$/\1 \2 \2/p' \> conftest.nm

configure:7129: $? = 0

configure:7181: gcc -o conftest -g -O2 conftest.c conftstm.o >&5

configure:7184: $? = 0

configure:7222: result: ok

configure:7226: checking for objdir

configure:7241: result: .libs

configure:7333: checking for ar

configure:7349: found /usr/bin/ar

configure:7360: result: ar

configure:7429: checking for ranlib

configure:7445: found /usr/bin/ranlib

configure:7456: result: ranlib

configure:7525: checking for strip

configure:7541: found /usr/bin/strip

configure:7552: result: strip

configure:7838: checking if gcc supports -fno-rtti -fno-exceptions

configure:7856: gcc -c -g -O2 -fno-rtti -fno-exceptions conftest.c >&5

cc1: warning: command line option '-fno-rtti' is valid for C++/ObjC++ but not for C [enabled by default]

configure:7860: $? = 0

configure:7873: result: no

configure:7888: checking for gcc option to produce PIC

configure:8098: result: -fPIC

configure:8106: checking if gcc PIC flag -fPIC works

configure:8124: gcc -c -g -O2 -fPIC -DPIC conftest.c >&5

configure:8128: $? = 0

configure:8141: result: yes

configure:8169: checking if gcc static flag -static works

configure:8197: result: yes

configure:8207: checking if gcc supports -c -o file.o

configure:8228: gcc -c -g -O2 -o out/conftest2.o conftest.c >&5

configure:8232: $? = 0

chmod: changing permissions of `.': Operation not permitted

configure:8254: result: yes

configure:8280: checking whether the gcc linker (/usr/bin/ld) supports shared libraries

configure:9266: result: yes

configure:9287: checking whether -lc should be explicitly linked in

configure:9292: gcc -c -g -O2 conftest.c >&5

configure:9295: $? = 0

configure:9310: gcc -shared conftest.o -v -Wl,-soname -Wl,conftest -o conftest 2\>\&1 \| grep -lc \>/dev/null 2\>\&1

configure:9313: $? = 0

configure:9325: result: no

configure:9333: checking dynamic linker characteristics

configure:9942: result: GNU/Linux ld.so

configure:9951: checking how to hardcode library paths into programs

configure:9976: result: immediate

configure:9990: checking whether stripping libraries is possible

configure:9995: result: yes

configure:10908: checking if libtool supports shared libraries

configure:10910: result: yes

configure:10913: checking whether to build shared libraries

configure:10934: result: yes

configure:10937: checking whether to build static libraries

configure:10941: result: no

configure:11033: creating libtool

configure:20315: checking for special C compiler options needed for large files

configure:20442: result: no

configure:20448: checking for \_FILE\_OFFSET\_BITS value needed for large files

configure:20484: gcc -c -g -O2 conftest.c >&5

conftest.c:26:3: warning: left shift count >= width of type [enabled by default]

conftest.c:26:3: warning: left shift count >= width of type [enabled by default]

conftest.c:27:10: warning: left shift count >= width of type [enabled by default]

conftest.c:27:10: warning: left shift count >= width of type [enabled by default]

conftest.c:26:7: error: size of array 'off\_t\_is\_large' is negative

conftest.c:28:9: warning: variably modified 'off\_t\_is\_large' at file scope [enabled by default]

configure:20490: $? = 1

configure: failed program was:

| /\* confdefs.h. \*/

| #define PACKAGE\_NAME "gtk+"

| #define PACKAGE\_TARNAME "gtk+"

| #define PACKAGE\_VERSION "2.10.9"

| #define PACKAGE\_STRING "gtk+ 2.10.9"

| #define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

| #define GETTEXT\_PACKAGE "gtk20"

| #define STDC\_HEADERS 1

| #define HAVE\_SYS\_TYPES\_H 1

| #define HAVE\_SYS\_STAT\_H 1

| #define HAVE\_STDLIB\_H 1

| #define HAVE\_STRING\_H 1

| #define HAVE\_MEMORY\_H 1

| #define HAVE\_STRINGS\_H 1

| #define HAVE\_INTTYPES\_H 1

| #define HAVE\_STDINT\_H 1

| #define HAVE\_UNISTD\_H 1

| #define HAVE\_DLFCN\_H 1

| /\* end confdefs.h. \*/

| #include <sys/types.h>

| /\* Check that off\_t can represent 2\*\*63 - 1 correctly.

| We can't simply define LARGE\_OFF\_T to be 9223372036854775807,

| since some C++ compilers masquerading as C compilers

| incorrectly reject 9223372036854775807. \*/

| #define LARGE\_OFF\_T (((off\_t) 1 << 62) - 1 + ((off\_t) 1 << 62))

| int off\_t\_is\_large[(LARGE\_OFF\_T % 2147483629 == 721

| && LARGE\_OFF\_T % 2147483647 == 1)

| ? 1 : -1];

| int

| main ()

| {

|

| ;

| return 0;

| }

configure:20551: gcc -c -g -O2 conftest.c >&5

configure:20557: $? = 0

configure:20564: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:20567: $? = 0

configure:20574: test -s conftest.o

configure:20577: $? = 0

configure:20591: result: 64

configure:20601: checking for \_LARGE\_FILES value needed for large files

configure:20637: gcc -c -g -O2 conftest.c >&5

configure:20643: $? = 0

configure:20650: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:20653: $? = 0

configure:20660: test -s conftest.o

configure:20663: $? = 0

configure:20744: result: no

configure:20766: checking for nm

configure:20797: result: /usr/bin/nm -B

configure:20806: checking whether to enable maintainer-specific portions of Makefiles

configure:20815: result: no

configure:20831: checking for some Win32 platform

configure:20841: result: no

configure:21161: checking whether build environment is sane

configure:21204: result: yes

configure:21208: checking for library containing strerror

configure:21249: gcc -o conftest -g -O2 conftest.c >&5

configure:21255: $? = 0

configure:21262: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:21265: $? = 0

configure:21272: test -s conftest

configure:21275: $? = 0

configure:21299: result: none required

configure:21310: checking for gcc option to accept ANSI C

configure:21378: gcc -c -g -O2 conftest.c >&5

configure:21384: $? = 0

configure:21391: test -z "$ac\_c\_werror\_flag" || test ! -s conftest.err

configure:21394: $? = 0

configure:21401: test -s conftest.o

configure:21404: $? = 0

configure:21421: result: none needed

configure:21445: checking for a BSD-compatible install

configure:21501: result: /usr/bin/install -c

configure:21512: checking whether make sets $(MAKE)

configure:21533: result: yes

configure:21659: checking for pkg-config

configure:21677: found /usr/bin/pkg-config

configure:21689: result: /usr/bin/pkg-config

configure:21718: checking pkg-config is at least version 0.9.0

configure:21721: result: yes

configure:21732: checking for BASE\_DEPENDENCIES

configure:21740: $PKG\_CONFIG --exists --print-errors "glib-2.0 >= 2.12.0 atk >= 1.9.0 pango >= 1.12.0 cairo >= 1.2.0"

Package glib-2.0 was not found in the pkg-config search path.

Perhaps you should add the directory containing `glib-2.0.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'glib-2.0' found

Package atk was not found in the pkg-config search path.

Perhaps you should add the directory containing `atk.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'atk' found

Package pango was not found in the pkg-config search path.

Perhaps you should add the directory containing `pango.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'pango' found

Package cairo was not found in the pkg-config search path.

Perhaps you should add the directory containing `cairo.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'cairo' found

configure:21743: $? = 1

configure:21758: $PKG\_CONFIG --exists --print-errors "glib-2.0 >= 2.12.0 atk >= 1.9.0 pango >= 1.12.0 cairo >= 1.2.0"

Package glib-2.0 was not found in the pkg-config search path.

Perhaps you should add the directory containing `glib-2.0.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'glib-2.0' found

Package atk was not found in the pkg-config search path.

Perhaps you should add the directory containing `atk.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'atk' found

Package pango was not found in the pkg-config search path.

Perhaps you should add the directory containing `pango.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'pango' found

Package cairo was not found in the pkg-config search path.

Perhaps you should add the directory containing `cairo.pc'

to the PKG\_CONFIG\_PATH environment variable

No package 'cairo' found

configure:21761: $? = 1

No package 'glib-2.0' found

No package 'atk' found

No package 'pango' found

No package 'cairo' found

configure:21799: error: Package requirements (glib-2.0 >= 2.12.0 atk >= 1.9.0 pango >= 1.12.0 cairo >= 1.2.0) were not met:

No package 'glib-2.0' found

No package 'atk' found

No package 'pango' found

No package 'cairo' found

Consider adjusting the PKG\_CONFIG\_PATH environment variable if you

installed software in a non-standard prefix.

Alternatively, you may set the environment variables BASE\_DEPENDENCIES\_CFLAGS

and BASE\_DEPENDENCIES\_LIBS to avoid the need to call pkg-config.

See the pkg-config man page for more details.

## ---------------- ##

## Cache variables. ##

## ---------------- ##

ac\_cv\_build=i686-pc-linux-gnu

ac\_cv\_c\_compiler\_gnu=yes

ac\_cv\_cxx\_compiler\_gnu=no

ac\_cv\_env\_BASE\_DEPENDENCIES\_CFLAGS\_set=

ac\_cv\_env\_BASE\_DEPENDENCIES\_CFLAGS\_value=

ac\_cv\_env\_BASE\_DEPENDENCIES\_LIBS\_set=

ac\_cv\_env\_BASE\_DEPENDENCIES\_LIBS\_value=

ac\_cv\_env\_CCC\_set=

ac\_cv\_env\_CCC\_value=

ac\_cv\_env\_CC\_set=

ac\_cv\_env\_CC\_value=

ac\_cv\_env\_CFLAGS\_set=

ac\_cv\_env\_CFLAGS\_value=

ac\_cv\_env\_CPPFLAGS\_set=

ac\_cv\_env\_CPPFLAGS\_value=

ac\_cv\_env\_CPP\_set=

ac\_cv\_env\_CPP\_value=

ac\_cv\_env\_CXXCPP\_set=

ac\_cv\_env\_CXXCPP\_value=

ac\_cv\_env\_CXXFLAGS\_set=

ac\_cv\_env\_CXXFLAGS\_value=

ac\_cv\_env\_CXX\_set=

ac\_cv\_env\_CXX\_value=

ac\_cv\_env\_F77\_set=

ac\_cv\_env\_F77\_value=

ac\_cv\_env\_FFLAGS\_set=

ac\_cv\_env\_FFLAGS\_value=

ac\_cv\_env\_LDFLAGS\_set=

ac\_cv\_env\_LDFLAGS\_value=

ac\_cv\_env\_PKG\_CONFIG\_set=

ac\_cv\_env\_PKG\_CONFIG\_value=

ac\_cv\_env\_XMKMF\_set=

ac\_cv\_env\_XMKMF\_value=

ac\_cv\_env\_build\_alias\_set=

ac\_cv\_env\_build\_alias\_value=

ac\_cv\_env\_host\_alias\_set=

ac\_cv\_env\_host\_alias\_value=

ac\_cv\_env\_target\_alias\_set=

ac\_cv\_env\_target\_alias\_value=

ac\_cv\_f77\_compiler\_gnu=no

ac\_cv\_header\_dlfcn\_h=yes

ac\_cv\_header\_inttypes\_h=yes

ac\_cv\_header\_memory\_h=yes

ac\_cv\_header\_stdc=yes

ac\_cv\_header\_stdint\_h=yes

ac\_cv\_header\_stdlib\_h=yes

ac\_cv\_header\_string\_h=yes

ac\_cv\_header\_strings\_h=yes

ac\_cv\_header\_sys\_stat\_h=yes

ac\_cv\_header\_sys\_types\_h=yes

ac\_cv\_header\_unistd\_h=yes

ac\_cv\_host=i686-pc-linux-gnu

ac\_cv\_objext=o

ac\_cv\_path\_EGREP='/bin/grep -E'

ac\_cv\_path\_GREP=/bin/grep

ac\_cv\_path\_NM='/usr/bin/nm -B'

ac\_cv\_path\_ac\_pt\_PKG\_CONFIG=/usr/bin/pkg-config

ac\_cv\_path\_install='/usr/bin/install -c'

ac\_cv\_prog\_AWK=mawk

ac\_cv\_prog\_CPP='gcc -E'

ac\_cv\_prog\_ac\_ct\_AR=ar

ac\_cv\_prog\_ac\_ct\_CC=gcc

ac\_cv\_prog\_ac\_ct\_RANLIB=ranlib

ac\_cv\_prog\_ac\_ct\_STRIP=strip

ac\_cv\_prog\_cc\_c89=

ac\_cv\_prog\_cc\_g=yes

ac\_cv\_prog\_cxx\_g=no

ac\_cv\_prog\_f77\_g=no

ac\_cv\_prog\_make\_make\_set=yes

ac\_cv\_search\_strerror='none required'

ac\_cv\_sys\_file\_offset\_bits=64

ac\_cv\_sys\_large\_files=no

ac\_cv\_sys\_largefile\_CC=no

am\_cv\_CC\_dependencies\_compiler\_type=gcc3

am\_cv\_CXX\_dependencies\_compiler\_type=none

am\_cv\_prog\_cc\_stdc=

lt\_cv\_deplibs\_check\_method=pass\_all

lt\_cv\_file\_magic\_cmd='$MAGIC\_CMD'

lt\_cv\_file\_magic\_test\_file=

lt\_cv\_ld\_reload\_flag=-r

lt\_cv\_objdir=.libs

lt\_cv\_path\_LD=/usr/bin/ld

lt\_cv\_path\_NM='/usr/bin/nm -B'

lt\_cv\_path\_SED=/bin/sed

lt\_cv\_prog\_compiler\_c\_o=yes

lt\_cv\_prog\_compiler\_rtti\_exceptions=no

lt\_cv\_prog\_gnu\_ld=yes

lt\_cv\_sys\_global\_symbol\_pipe='sed -n -e '\''s/^.\*[ ]\([ABCDGIRSTW][ABCDGIRSTW]\*\)[ ][ ]\*\([\_A-Za-z][\_A-Za-z0-9]\*\)$/\1 \2 \2/p'\'''

lt\_cv\_sys\_global\_symbol\_to\_c\_name\_address='sed -n -e '\''s/^: \([^ ]\*\) $/ {\"\1\", (lt\_ptr) 0},/p'\'' -e '\''s/^[BCDEGRST] \([^ ]\*\) \([^ ]\*\)$/ {"\2", (lt\_ptr) \&\2},/p'\'''

lt\_cv\_sys\_global\_symbol\_to\_cdecl='sed -n -e '\''s/^. .\* \(.\*\)$/extern int \1;/p'\'''

lt\_cv\_sys\_max\_cmd\_len=32768

lt\_lt\_cv\_prog\_compiler\_c\_o='"yes"'

lt\_lt\_cv\_sys\_global\_symbol\_pipe='"sed -n -e '\''s/^.\*[ ]\\([ABCDGIRSTW][ABCDGIRSTW]\*\\)[ ][ ]\*\\([\_A-Za-z][\_A-Za-z0-9]\*\\)\$/\\1 \\2 \\2/p'\''"'

lt\_lt\_cv\_sys\_global\_symbol\_to\_c\_name\_address='"sed -n -e '\''s/^: \\([^ ]\*\\) \$/ {\\\"\\1\\\", (lt\_ptr) 0},/p'\'' -e '\''s/^[BCDEGRST] \\([^ ]\*\\) \\([^ ]\*\\)\$/ {\"\\2\", (lt\_ptr) \\&\\2},/p'\''"'

lt\_lt\_cv\_sys\_global\_symbol\_to\_cdecl='"sed -n -e '\''s/^. .\* \\(.\*\\)\$/extern int \\1;/p'\''"'

## ----------------- ##

## Output variables. ##

## ----------------- ##

ACLOCAL='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run aclocal-1.7 '

AMDEPBACKSLASH='\'

AMDEP\_FALSE='#'

AMDEP\_TRUE=''

AMTAR='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run tar'

AR='ar'

AS='as'

AUTOCONF='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run autoconf'

AUTOHEADER='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run autoheader'

AUTOMAKE='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run automake-1.7'

AWK='mawk'

BASE\_DEPENDENCIES\_CFLAGS=''

BASE\_DEPENDENCIES\_LIBS=''

BUILD\_DYNAMIC\_MODULES\_FALSE=''

BUILD\_DYNAMIC\_MODULES\_TRUE=''

CATALOGS=''

CATOBJEXT=''

CC='gcc'

CCAS='$(CC)'

CCASFLAGS='$(CFLAGS)'

CCDEPMODE='depmode=gcc3'

CFLAGS='-g -O2 -Wall'

CPP='gcc -E'

CPPFLAGS=''

CROSS\_COMPILING\_FALSE=''

CROSS\_COMPILING\_TRUE=''

CUPS\_API\_MAJOR=''

CUPS\_API\_MINOR=''

CUPS\_CFLAGS=''

CUPS\_CONFIG=''

CUPS\_LIBS=''

CXX='g++'

CXXCPP=''

CXXDEPMODE='depmode=none'

CXXFLAGS=''

CYGPATH\_W='echo'

DATADIRNAME=''

DB2HTML=''

DEFS=''

DEPDIR='.deps'

DISABLE\_EXPLICIT\_DEPS\_FALSE=''

DISABLE\_EXPLICIT\_DEPS\_TRUE=''

DLLTOOL='dlltool'

ECHO='echo'

ECHO\_C=''

ECHO\_N='-n'

ECHO\_T=''

EGREP='/bin/grep -E'

ENABLE\_FB\_MANAGER\_FALSE=''

ENABLE\_FB\_MANAGER\_TRUE=''

ENABLE\_GTK\_DOC\_FALSE=''

ENABLE\_GTK\_DOC\_TRUE=''

ENABLE\_MAN\_FALSE=''

ENABLE\_MAN\_TRUE=''

EXEEXT=''

F77=''

FFLAGS=''

GDK\_DEP\_CFLAGS=''

GDK\_DEP\_LIBS=''

GDK\_EXTRA\_CFLAGS=''

GDK\_EXTRA\_LIBS=''

GDK\_PACKAGES=''

GDK\_PIXBUF\_CSOURCE=''

GDK\_PIXBUF\_DEP\_CFLAGS=''

GDK\_PIXBUF\_DEP\_LIBS=''

GDK\_PIXBUF\_EXTRA\_CFLAGS=''

GDK\_PIXBUF\_EXTRA\_LIBS=''

GDK\_PIXBUF\_MAJOR='2'

GDK\_PIXBUF\_MICRO='9'

GDK\_PIXBUF\_MINOR='10'

GDK\_PIXBUF\_PACKAGES=''

GDK\_PIXBUF\_VERSION='2.10.9'

GDK\_PIXBUF\_XLIB\_DEP\_CFLAGS=''

GDK\_PIXBUF\_XLIB\_DEP\_LIBS=''

GDK\_PIXBUF\_XLIB\_EXTRA\_CFLAGS=''

GDK\_PIXBUF\_XLIB\_EXTRA\_LIBS=''

GDK\_PIXBUF\_XLIB\_PACKAGES=''

GDK\_WIN32\_EXTRA\_CFLAGS=''

GDK\_WLIBS=''

GETTEXT\_PACKAGE='gtk20'

GLIB\_CFLAGS=''

GLIB\_GENMARSHAL=''

GLIB\_LIBS=''

GLIB\_MKENUMS=''

GMOFILES=''

GMSGFMT=''

GOBJECT\_QUERY=''

GREP='/bin/grep'

GTK\_API\_VERSION='2.0'

GTK\_BINARY\_AGE='1009'

GTK\_BINARY\_VERSION='2.10.0'

GTK\_DEBUG\_FLAGS='-DG\_DISABLE\_CAST\_CHECKS'

GTK\_DEP\_CFLAGS=''

GTK\_DEP\_LIBS=''

GTK\_DOC\_USE\_LIBTOOL\_FALSE=''

GTK\_DOC\_USE\_LIBTOOL\_TRUE=''

GTK\_EXTRA\_CFLAGS=''

GTK\_EXTRA\_LIBS=''

GTK\_INTERFACE\_AGE='9'

GTK\_MAJOR\_VERSION='2'

GTK\_MICRO\_VERSION='9'

GTK\_MINOR\_VERSION='10'

GTK\_PACKAGES=''

GTK\_UPDATE\_ICON\_CACHE=''

GTK\_VERSION='2.10.9'

GTK\_XIM\_FLAGS=''

HAVE\_CUPS\_FALSE=''

HAVE\_CUPS\_TRUE=''

HAVE\_DOCBOOK\_FALSE=''

HAVE\_DOCBOOK\_TRUE=''

HAVE\_IE55\_FALSE=''

HAVE\_IE55\_TRUE=''

HAVE\_JPEG\_FALSE=''

HAVE\_JPEG\_TRUE=''

HAVE\_PNG\_FALSE=''

HAVE\_PNG\_TRUE=''

HAVE\_TIFF\_FALSE=''

HAVE\_TIFF\_TRUE=''

HAVE\_WINTAB\_FALSE=''

HAVE\_WINTAB\_TRUE=''

HAVE\_X11R6\_FALSE=''

HAVE\_X11R6\_TRUE=''

HTML\_DIR=''

IE55\_UUID\_LIB=''

INCLUDED\_LOADER\_DEFINE=''

INCLUDED\_LOADER\_OBJ=''

INDENT=''

INSTALL\_DATA='${INSTALL} -m 644'

INSTALL\_PROGRAM='${INSTALL}'

INSTALL\_SCRIPT='${INSTALL}'

INSTALL\_STRIP\_PROGRAM='${SHELL} $(install\_sh) -c -s'

INSTOBJEXT=''

INTLLIBS=''

LDFLAGS=''

LIBJPEG=''

LIBOBJS=''

LIBPNG=''

LIBS=''

LIBTIFF=''

LIBTOOL='$(SHELL) $(top\_builddir)/libtool'

LIBTOOL\_EXPORT\_OPTIONS=''

LN\_S='ln -s'

LTLIBOBJS=''

LT\_CURRENT\_MINUS\_AGE='0'

LT\_VERSION\_INFO='1000:9:1000'

MAINT='#'

MAINTAINER\_MODE\_FALSE=''

MAINTAINER\_MODE\_TRUE='#'

MAKEINFO='${SHELL} /media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/missing --run makeinfo'

MKINSTALLDIRS=''

MSGFMT=''

MS\_LIB\_AVAILABLE\_FALSE=''

MS\_LIB\_AVAILABLE\_TRUE='#'

NM='/usr/bin/nm -B'

OBJDUMP='objdump'

OBJEXT='o'

OS\_LINUX\_FALSE='#'

OS\_LINUX\_TRUE=''

OS\_UNIX\_FALSE='#'

OS\_UNIX\_TRUE=''

OS\_WIN32\_FALSE=''

OS\_WIN32\_TRUE='#'

PACKAGE='gtk+'

PACKAGE\_BUGREPORT='http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B'

PACKAGE\_NAME='gtk+'

PACKAGE\_STRING='gtk+ 2.10.9'

PACKAGE\_TARNAME='gtk+'

PACKAGE\_VERSION='2.10.9'

PATH\_SEPARATOR=':'

PERL=''

PKG\_CONFIG='/usr/bin/pkg-config'

PLATFORM\_WIN32\_FALSE=''

PLATFORM\_WIN32\_TRUE='#'

POFILES=''

POSUB=''

PO\_IN\_DATADIR\_FALSE=''

PO\_IN\_DATADIR\_TRUE=''

RANLIB='ranlib'

REBUILD=''

REBUILD\_PNGS=''

SET\_MAKE=''

SHELL='/bin/bash'

STRIP='strip'

USE\_DIRECTFB\_FALSE=''

USE\_DIRECTFB\_TRUE=''

USE\_LINUX\_FB\_FALSE=''

USE\_LINUX\_FB\_TRUE=''

USE\_MMX\_FALSE=''

USE\_MMX\_TRUE=''

USE\_NLS=''

USE\_QUARTZ\_FALSE=''

USE\_QUARTZ\_TRUE=''

USE\_WIN32\_FALSE=''

USE\_WIN32\_TRUE=''

USE\_X11\_FALSE=''

USE\_X11\_TRUE=''

VERSION='2.10.9'

WINDRES=''

WINTAB\_LIB=''

XGETTEXT=''

XINPUT\_XFREE\_FALSE=''

XINPUT\_XFREE\_TRUE=''

XMKMF=''

XMLCATALOG=''

XML\_CATALOG\_FILE=''

XSLTPROC=''

X\_CFLAGS=''

X\_EXTRA\_LIBS=''

X\_LIBS=''

X\_PRE\_LIBS=''

ac\_ct\_CC='gcc'

ac\_ct\_CXX=''

ac\_ct\_F77=''

am\_\_fastdepCC\_FALSE='#'

am\_\_fastdepCC\_TRUE=''

am\_\_fastdepCXX\_FALSE=''

am\_\_fastdepCXX\_TRUE='#'

am\_\_include='include'

am\_\_leading\_dot='.'

am\_\_quote=''

bindir='${exec\_prefix}/bin'

build='i686-pc-linux-gnu'

build\_alias=''

build\_cpu='i686'

build\_os='linux-gnu'

build\_vendor='pc'

datadir='${datarootdir}'

datarootdir='${prefix}/share'

docdir='${datarootdir}/doc/${PACKAGE\_TARNAME}'

dvidir='${docdir}'

exec\_prefix='NONE'

gdktarget='x11'

gdktargetlib='libgdk-x11-2.0.la'

gtktargetlib='libgtk-x11-2.0.la'

host='i686-pc-linux-gnu'

host\_alias=''

host\_cpu='i686'

host\_os='linux-gnu'

host\_vendor='pc'

htmldir='${docdir}'

includedir='${prefix}/include'

infodir='${datarootdir}/info'

install\_sh='/media/windoza/Linux/Nessus\_instal/gtk/gtk+-2.10.9/install-sh'

libdir='${exec\_prefix}/lib'

libexecdir='${exec\_prefix}/libexec'

localedir='${datarootdir}/locale'

localstatedir='${prefix}/var'

mandir='${datarootdir}/man'

ms\_librarian=''

oldincludedir='/usr/include'

pdfdir='${docdir}'

prefix='NONE'

program\_transform\_name='s,x,x,'

psdir='${docdir}'

sbindir='${exec\_prefix}/sbin'

sharedstatedir='${prefix}/com'

sysconfdir='${prefix}/etc'

target\_alias=''

## ----------- ##

## confdefs.h. ##

## ----------- ##

#define PACKAGE\_NAME "gtk+"

#define PACKAGE\_TARNAME "gtk+"

#define PACKAGE\_VERSION "2.10.9"

#define PACKAGE\_STRING "gtk+ 2.10.9"

#define PACKAGE\_BUGREPORT "http://bugzilla.gnome.org/enter\_bug.cgi?product=gtk%2B"

#define GETTEXT\_PACKAGE "gtk20"

#define STDC\_HEADERS 1

#define HAVE\_SYS\_TYPES\_H 1

#define HAVE\_SYS\_STAT\_H 1

#define HAVE\_STDLIB\_H 1

#define HAVE\_STRING\_H 1

#define HAVE\_MEMORY\_H 1

#define HAVE\_STRINGS\_H 1

#define HAVE\_INTTYPES\_H 1

#define HAVE\_STDINT\_H 1

#define HAVE\_UNISTD\_H 1

#define HAVE\_DLFCN\_H 1

#define \_FILE\_OFFSET\_BITS 64

#define GTK\_COMPILED\_WITH\_DEBUGGING "minimum"

configure: exit 1