МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра систем информационной безопасности

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой к.п.н., доцент Р. Ш. Бердибаев
_____ «_____» _____2019 г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка системы автоматизированного процесса сбора персональных данных
Специальность: 5В100200 – «Системы информационной безопасности»
Выполнил: Дифу Идрис Хусаинович Группа СИБ-15-3
Научный руководитель: Дмитриева Маргарита Валерьевна

Консультант:
по экономической части:
к.э.н., профессор Аренбаева Нс.Г.
(ученая степень, звание, Ф.И.О)
(подпись) «31» мая 2019 г.

по безопасности жизнедеятельности:
д.т.н., ст.преп Бейбасаров Н.Н.
(ученая степень, звание, Ф.И.О)
(подпись) «29» мая 2019 г.

по применению вычислительной техники:
ст. преподаватель Дмитриева М.В.
(ученая степень, звание, Ф.И.О)
(подпись) «24» мая 2019 г.

Нормоконтролер:
Ст. преподаватель, м.т.н. Аскарова Ж.Б.
(ученая степень, звание, Ф.И.О)
(подпись) «31» 05 2019 г.

Рецензент:
К.Т.Н, доцент пос. проф. кафедры Ананжанова С.Т.
(ученая степень, звание, Ф.И.О)
(подпись) «30» мая 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра систем информационной безопасности

Специальность 5В100200 – «Системы информационной безопасности»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Дифу Идрису Хусаиновичу

Тема проекта: Построение доказательной базы при расследовании проникновений в объекты сетевой инфраструктуры

Утверждена приказом по университету №124 от «26» 10 2019 г.

Срок сдачи законченного проекта «_____» _____2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): проект подразумевает разработку программного обеспечения, которое позволяет ускорить процесс сбора доказательной базы на возможный источник угрозы (определенный IP-адрес) сетевой инфраструктуре. Программное обеспечение основано на работе с DNSBL (Domain Name System Black Lists), принцип работы состоит в том, что пользователь вводит определенный IP-адрес, и разрабатываемое программное обеспечение будет проверять его в большом количестве общедоступных, ежедневно обновляемых базах данных IP с плохой репутацией. Реализовано описанное программное обеспечение будет на скриптовом языке программирования BASH.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта: дипломный проект включает в себя 5 глав, разделенных на подглавы, каждая из которых освещает определенную тематику, используемую при сборе доказательной базы.

В первой главе дипломного проекта представлена общая информация по доказательной базе: типы доказательных баз, плюсы и минусы.

Во второй главе дипломного проекта представлена общая структура разрабатываемого программного обеспечения: подробное описание исходного кода разрабатываемого программного обеспечения.

В третьей главе подробно описывается функционал параметром запуска ПО в отдельности.

В четвертой главе приводится технико-экономическое обоснование, показывающее актуальность разработки ПО с финансовой точки зрения.

В пятой главе рассматриваются необходимые условия для комфортной разработки программного обеспечения.

Конструкции по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Вычислительной ча	Дмитриева М.В	18.04-15.04	
Вычислительной техники	Дмитриева М.В	16.04-23.05	
Вычислительной техники	Дмитриева М.В	20.05-24.05	
Безопасности жизнеде.	Бебасаров Ш.Ш.	04.03-29.05	
экономической части	Арендаева Ж.Г.	04.03-31.05.	

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Построение доказательной базы	04.04.2019-24.05.2019	
Доказательная база	12.04.2019-24.05.2019	
Расследования преступлений ведомств	18.04.2019-20.05.2019	
Ядро межкомпьют. инт. Shell и комм. проц BASH	19.04.2019-21.05.2019	
Unix сист. и Shell разработки	20.04.2019-22.05.2019	
Команды и перемен Shell	22.04.2019-23.05.2019	
Среда выполнение команд	22.04.2019-23.05.2019	
Управление потоком выполнения	23.04.2019-24.05.2019	
DHSBd и методы использования	23.04.2019-24.05.2019	
Разработка программного обеспечения	25.04.2019 24.05.2019	
Производение тестовой атаки на сервер	13.05.2019-24.05.2019	
Безопасность жизнедеятельности	04.03.2019-29.05.2019	
Расчет тепловых нагрузок в помещении	15.04.2019-29.2019	
Расчет количества возд. необх. в пом.	15.05.2019-29.05.2019	
Осн. хар-ки и схема располож.	20.05.2019-29.05.2019	
Технико-эконом. обоснование	03.04.2019 - 31.05.2019	
Расчет трудоемкости разраб. прог.-прод	03.04.2019 - 31.05.2019	
Расчет затрат на опл. труда	04.04.2019 - 31.05.2019	
Расчет затрат по соц. налогу	05.04.2019 - 31.05.2019	
Амортиз. осн. фондов и проч. затр	07.04.2019-24.05.2019	
Определение возмож. цены	09.04.2019-24.05.2019	

Дата выдачи задания «___» ________________ 2019 г.

Заведующий кафедрой __________ (______________________)
(Подпись) (Ф.И.О)

Научный руководитель проекта __________ (Дмитриева И.В)
(Подпись) (Ф.И.О)

Задание принял к исполнению студент __________ (Дауру Идрис Хусаинович)
(Подпись) (Ф.И.О)

АННОТАЦИЯ

В дипломном проекте реализовано программное обеспечение, которое проверяет введенный IP-адрес сразу в большом количестве DNSBL (Domain Name System Black List). ПО работает на скриптовом языке программирования Shell и разработано в операционной системе Linux Ubuntu.

DNSBL (Domain Name System Black List) – черные списки доменов, находятся в открытом доступе, различные IP-адреса попадают в данные черные списки только в том случае, если их активность в открытой сети интернет была подозрительной.

Глава по безопасности жизнедеятельности характеризует благоприятные условия труда. В экономической части были приведены расчеты затрат на создание ПО и прибыль предприятия в случае внедрения предлагаемой модели.

АҢДАТПА

Дипломдық жобада енгізілген IP-адресті DNSBL (Domain Name System Black List) үлкен мөлшерде тексеретін бағдарлама жүзеге асырылған. Бағдарлама Shell скрипттік бағдарламалау тілінде іске қосылады және Linux Ubuntu операциялық жүйесінде әзірленген.

DNSBL (Domain Name System Black List) – домендердің қара тізімдері жалпыға қол жетімді болып табылады, әр түрлі IP мекенжайлар осы қоғамдық тізімдерде олардың қоғамдық интернет-қызметтеріндегі күдікті болған жағдайда ғана қосылады.

Өмір сүру қауіпсіздігі тарауында еңбек жағдайлары бойынша қолайлы екені сипатталады. Экономикалық тарауда бағдарламалық құралды құру құны есептелген және кәсіпорынның жобаны іске асырған жағдайдағы пайдасы есептелген.

ANNOTATION

In the thesis project implemented software that checks the entered IP-address immediately in a large number of DNSBL (Domain Name System Black List). The software runs on the Shell scripting programming language and is developed in the Linux operating system Ubuntu.

DNSBL (Domain Name System Black List) – blacklists of domains that are publicly available, various IP addresses are included in these blacklists only if their activity on the public Internet was suspicious.

The chapter on life safety characterizes favorable working conditions. In the economic part, calculations were made of the costs of creating software and the profit of the enterprise in the case of the introduction of the proposed model.

Содержание

Введение

Сейчас IT-индустрия очень быстро развивается. Одной из важнейших проблем, что до сих пор решаются руководством каждой крупной компанией в мире, для каждой из них тема безопасности данных является одной из самых важнейших. Говоря в целом, информационная безопасность должна не только предотвращать утечку какой-либо информации, но и отражать атаки на сетевую инфраструктуру предприятия. Времена, когда процесс решения одной задачи длился длительное время, и на данный процесс выделялись грандиозные ресурсы прошли, сфера информационной безопасности не стоит на месте, и сейчас необходимо получать максимально количество результата, при этом затрачивая минимальное количество ресурсов. Различные направления сфер деятельности информационной безопасности усложняет этот процесс, в такой ситуации необходимо мыслить более комплексно. При подключении к корпоративной сети IP, и выявлении его действий как подозрительных, необходимо проанализировать его подключения, действия в сети, и собрать информацию по данному IP адресу.

Прежде, чем приступить к блокировке IP адреса, с которого возможно проводится атака, необходимо в кратчайшие сроки собрать доказательную базу по данному IP адресу, опираясь на информацию, полученную в ходе его обращения и информацию, которая хранится в открытом доступе, уже принимать решение, блокировать ли его или нет.

В моем дипломном проекте будет разрабатываться ПО, которое в кратчайшие сроки соберет большой объем информации по потенциально вредоносному IP-адресу и выведет результаты всех запросов в открытые источники по проверке рейтинга указанного IP-адреса.

Целью моей дипломной работы является создание программного обеспечения, которое в кратчайшие сроки соберет информацию по введенному IP адресу из большого количества открытых источников.

1 Построение доказательной базы

1.1 Доказательная база

В настоящее время существует такая ситуация, когда при выявлении какого-либо инцидента информационной безопасности, высшим руководством предприятий очень часто принимается решение скрывать их. В целом связано такое развитие событий с тем, что по факту разобрать инцидент информационной безопасности до самого его конца является очень сложной задачей. Причиной возникновения таких осложнений можно назвать полнейшее отсутствие у специалистов информационной безопасности необходимого для решения поставленной задачи уровня знаний. Как результат – большинство инцидентов информационной безопасности не рассматриваются, сбор доказательной базы по возникшим инцидентам не доводится до их логического конца.

Безнаказанность в таких ситуациях всегда приводит к увеличению количества инцидентов и росту денежного ущерба, что несут крупнейшие организации, данный факт и вынуждает департаменты безопасности и защиты информации, а также производить закупку и внедрение других средств защиты информации и ее последующего контроля. Одновременно со всем происходящим, управление инцидентами в сфере информационной безопасности и других отраслях имеет довольно-таки большое количество схожих факторов. Существует единственный качественный и действительно работающий подход – превентивный, не допускающий возникновения инцидентов. Для этого должен осуществляться постоянный мониторинг области, в которой потенциально может произойти инцидент, детальное и результативное расследование происходящих инцидентов и анализ вновь возникающих угроз.

Управление инцидентами в сфере информационной безопасности имеет ряд очень специфических особенностей, возникших, прежде всего, из-за того, что именно в сфере информационной безопасности и кибербезопасности человеческий фактор тут играет далеко не самую последнюю роль. Большинство случаев утечки информации возникают чаще всего из-за некомпетентности самих сотрудников компании, которые имеют доступ к конфиденциальной информации на законном уровне.

Целью разработки доказательной базы при расследовании проникновений в объекты сетевой инфраструктуры является сбор информации, ее анализ и последующее ее применение. Иными словами, доказательная база — это фундамент для дальнейшего расследования инцидента информационной безопасности. При расследовании возможного проникновения в объект сетевой инфраструктуры, специалист информационной безопасности работает непосредственно с IP адресом, с которого и происходило потенциальная попытка проникновения в систему.

После анализа действий данного IP в системе часто возникает ситуация, в которой непонятно, блокировать доступ данному IP адресу к ресурсам предприятия, или же нет. Вот тут уже специалист по информационной безопасности начинает проверять его на различных ресурсах по проверке рейтинга IP. Однако проверка лишь на паре таких сервисов не дает работнику увидеть полную картину, необходимо ПО, которое бы собирало доказательную базу по введенному IP с большого количества ресурсов и как можно в кратчайшие сроки. В результате пользователь совершает минимум действий, и получает максимальное количество результата, видит результаты проверки во всех сервисах в одном окне, при необходимости переходит по полученной ссылке и анализирует результат проверки более подробно.

Все выше сказанное подтверждает актуальность моей дипломной работы, которая заключается разработке ПО, которое бы собирало доказательную базу из большого количества открытых источников по проверке в как можно кратчайшие сроки. Откуда вытекают следующие задачи:

1) выбор языка программирования;
2) выбор среды разработки ПО;
3) сделать заключение дипломной работы.

При хранении и поддержке доступа к абсолютно любому информационному ресурсу, владелец данного информационного ресурса, либо уполномоченное им лицо, приступает к написанию перечня правил по использованию данного ресурса, правила должны содержать в себе информацию о том, что делать можно, что нельзя. Умышленное нарушение установленных правил классифицируется как атака на информацию.

С наступлением процесса массового внедрения рабочих станции во многие сферы человеческой деятельности, размер информации, которая теперь хранится в электронном виде на разных серверах во всем мире увеличилась в десятки тысяч раз. На сегодняшний день скопировать за полминуты и унести запоминающее устройство с файлом, содержащим какие-либо внутренние нормативные документы организации, намного проще, чем переписывать огромную стопку бумаг.

В первую очередь, в случае потери информации организацию будут преследовать денежные потери.

Новость о том, что произошла кража большого объема информации всегда очень серьезно влияет на репутацию фирмы.

Фирмы-конкуренты могут воспользоваться кражей информации, если та осталась незамеченной, для того чтобы полностью постараться нанести фирме-конкуренты денежные убытки, навязывая ей фиктивные либо заведомо убыточные сделки.

Многократные успешные атаки на информационные ресурсы фирмы, предоставляющую какой-либо вид информационных услуг, снижают доверие к фирме у клиентов, что, естественно, плохо сказывается на объеме доходов.

Конечно же компьютерные атаки могут принести и огромный моральный ущерб. Само собой, ни одному пользователю компьютерной сети не хочется, чтобы его письма кроме адресата получали еще и сторонние 5-10 человек, или, например, весь текст, что пользователь набирает на своих устройствах ввода, копировался в буфер, а уже позднее при подключении к сети интернет отправлялся на определенный сервер.

Интересная статистика по атакам на информационные ресурсы большого количества предприятий. Самыми часто возникающими причинами нанесения ущерба информации являются: случайная ошибка (человеческий фактор) – 52% случаев, намеренное нанесение ущерба — 10% случаев, отказ оборудования – 10% случаев, результат пожара – 15% случаев, результат работы системы пожаротушения – 10% случаев. Отсюда можно сделать вывод, что примерно каждый 10-ый случай является умышленной атакой на информационные активы предприятия.

Злоумышленники, добравшись до информации: в 50% либо крадут деньги с электронных счетов, в 20% пытались намеренно вывести из строя программное обеспечение, и также часто – в 25% случаев – принималось решение украсть информацию, в 16% злоумышленники фальсифицировали информацию.

Информация обладает следующими категориями:

Конфиденциальность – информация известна только тому кругу лиц, кому она предназначена.

Целостность – информация не была искажена.

Аутентичность – источником информации является именно то лицо, которое заявлено как ее автор.

При работе с информационными ресурсами опираются на две категории – система ведет себя так, как планировалось и что пользователь, подключенный в данный момент к информационному ресурсу, действительно является тем, за кого себя выдает.

1.2 Расследование проникновений в объекты сетевой инфраструктуры

Сетевая инфраструктура – специальное оборудование и программное обеспечение, являющееся основой для работы предприятия и эффективного обмена информацией. Любое современное предприятие обязано иметь отлично управляемую структура, а добиться такого результата можно лишь за счет качественного сбора, хранения и обработки корпоративных данных.

Отсутствие данной хорошо управляемой структуры не позволит решить большинство проблем, связанных и не связанных с информационной безопасностью.

В крупных компаниях в настоящее время информационная безопасность в основном обеспечивается только за счет внедрения межсетевых экранов, систем защиты от спама, различных антивирусных систем и других средств противодействия внешним угрозам. Все эти средства слабо помогают противостоять действиям своих же собственных сотрудников или другим лицам, допущенным к работе с внутренними ресурсами компании. В данном случае, защищаясь от внешнего нарушителя, компания остается практически беззащитной против внутреннего нарушителя безопасности. Внутренний нарушитель – это обладающий определенными правами на доступ к информационным ресурсам легитимный сотрудник компании. В последнее время основные случаи утечки критически важной информации были связаны именно с действиями внутреннего нарушителя, и поэтому защита от действий такого нарушителя, на текущий момент, является одной из наиболее приоритетных задач. Существующие системы безопасности, контролирующие действия сотрудников компании, не обеспечивают гарантированного предотвращения несанкционированных действий и сбора доказательной базы при их совершении. Подавляющие большинство систем умеет записывать только отдельно взятые действия сотрудников компании за компьютером.

Основные действия, записываемые такими системами, включают: запуск приложений, печать документов, подключение внешних записывающих устройств, передача сетевых данных, ввод текста с клавиатуры и копирование отдельных файлов. Поскольку операционная система в совокупности с установленным программным обеспечением является технически сложной, такие системы безопасности не могут предусмотреть всех возможных действий пользователя требующих пристального внимания и регистрации в журнале. Всегда есть возможность обнаружения различных уязвимостей системы позволяющих обойти установленную систему безопасности. Такой случай не будет явно зафиксирован в журнале и его обнаружение будет затруднено.

Специалисты информационной и экономической безопасности чаще всего расследуют инциденты информационной безопасности. Сам процесс расследования можно разделить на три этапа. На первом этапе производится сбор и анализ информации по возникшему инциденту, на втором этапе

начинается процесс выявления виновных, виновным после назначается приговор и происходит установка мер ответственности, далее начинается процесс выявления причин, по которым данный инцидент мог произойти и после назначается вынесение мер, по предотвращению таких же инцидентов в будущем.

Сбор данных об инциденте информационной безопасности – самая основная часть процесса расследования. Результат данного процесс напрямую зависит от качества собранных свидетельств.

На самом начальном этапе расследования необходимо относиться к каждому свидетельству с максимальной ответственностью. В области сбора свидетельств по информационной безопасности существует огромное количество каналов утечки любой информации, основные это копирование конфиденциальной информации от самих же сотрудников, которые имеют доступ к данной информации законно.

Основные каналы утечки в области компьютерных преступлений:

- копирование файлов на съемные носители;
- печать бумажных копий документов;
- сеть (браузер, FTP, облачные хранилища и т.д.);
- электронная почта;
- сервисы мгновенных сообщений;
- мобильные устройства;
- кража или потеря оборудования.

Наиболее элементарным каналом утечки является перемещение/копирование файлов на съемные носители информации. Сейчас существует огромное количество маленьких переносных запоминающих устройств, которые можно пронести незаметно и выкрасть большой объем информации. В роли переносного внешнего съемного запоминающего устройства информации также можно использовать различные плееры и другие устройства со встроенной памятью. В процессе использования таких носителей информации по своим служебным обязанностям, сотрудник обязан предельно бдительно относиться к сохранности данных и не оставлять его в незащищенном месте у всех на виду. Когда свидетельства собраны и проанализированы, надо установить причины возникновения инцидента для принятия превентивных мер и попытаться найти лиц, виновных в возникновении инцидента. Превентивные меры могут быть как организационного плана (разработка регламентов, дополнительное обучение и инструктаж сотрудников), так и технического (модернизация системы защиты информации, контроля доступа и т.п).

Выделяют следующие методы умышленной дезорганизации работы, вывода системы из строя, проникновения в систему и несанкционированного доступа к информации:

Физическое разрушение системы (путем взрыва, поджога и т. п.) или вывод из строя всех или отдельных наиболее важных компонентов

компьютерной системы (устройств, носителей важной системной информации, лиц из числа персонала и т. п.).

Отключение или вывод из строя подсистем обеспечения функционирования вычислительных систем (электропитания, охлаждения и вентиляции, линий связи и т. п.).

Действия по дезорганизации функционирования системы (изменение режимов работы программ, постановка мощных активных радиопомех на частотах работы устройств системы и т. п.): внедрение агентов в число персонала системы; вербовку (путем подкупа, шантажа и т. п.) персонала или отдельных пользователей, имеющих определенные полномочия; перехват побочных электромагнитных, акустических и других излучений устройств и линий связи, а также наводок активных излучений на вспомогательные технические средства, непосредственно не участвующие в обработке информации (телефонные линии, сети питания, отопления и т. п.).

Перехват данных, передаваемых по каналам связи, и их анализ с целью выяснения протоколов обмена, правил вхождения в связь и авторизации пользователя и последующих попыток их имитации для проникновения в систему.

Хищение носителей информации (магнитных дисков, лент, микросхем памяти, запоминающих устройств и др.).

Несанкционированное копирование носителей информации; хищение производственных отходов (распечаток, записей, списанных носителей информации и т. п.).

Чтение остаточной информации из оперативной памяти и с внешних запоминающих устройств.

Незаконное получение паролей и других реквизитов разграничения доступа (путем подбора, путем имитации интерфейса системы и т. д.) с последующей маскировкой под зарегистрированного пользователя; вскрытие шифров криптозащиты информации; незаконное подключение к линиям связи с целью работы в моменты пауз в действиях законного пользователя от его имени с последующим вводом ложных сообщений или модификацией передаваемых сообщений.

Внутренние угрозы представляют из себя угрозы со стороны рабочего персонала организации, а под внешними угрозами - угрозы от сторонних лиц или организаций.

Согласно недавно собранной статистике до 60-70% всех компьютерных преступлений связаны с внутренними нарушениями от самих сотрудников предприятия или организации, чаще всего перед увольнением сотрудник решает забрать с собой какое-либо количество конфиденциальной информации.

Внешние угрозы могут быть разделены на две группы – на локальные и удаленные угрозы. Первые предполагают потенциальное проникновение нарушителя на территорию организации и получения им доступа к

отдельному компьютеру или локальной сети. Вторые характерны для систем, подключенных к общедоступным глобальным сетям.

Сетевые системы отличаются тем, что, наряду с локальными атаками, осуществляемыми в пределах одной компьютерной системы, к ним применим специфический вид атак, обусловленный распределённостью ресурсов и информации в пространстве. Это так называемые сетевые (или удаленные) атаки. С развитием локальных и глобальных сетей именно удаленные атаки становятся лидирующими как по количеству попыток, так и по успешности их применения.

Удаленные атаки можно классифицировать по нескольким признакам.

Во-первых, в зависимости от характера воздействия на работу информационной системы удаленные атаки могут быть разделены на пассивные и активные.

Пассивное воздействие на распределенную вычислительную систему не связано с непосредственным влиянием на работу системы, но нарушает политику ее безопасности. Например, прослушивание канала связи в сети.

Активным воздействием на распределенную вычислительную систему называется воздействие, оказывающее непосредственное влияние на работу системы (изменение конфигурации, нарушение работоспособности и т. д.) и нарушающее принятую в ней политику безопасности. Практически все типы удаленных атак являются активными воздействиями.

Во-вторых, в зависимости от цели осуществления удаленной атаки они могут быть классифицированы на угрозы, связанные с нарушением конфиденциальности, а также целостности и отказа в обслуживании.

Основная цель практически любой атаки - получить несанкционированный доступ к информации. Существуют две принципиальные возможности доступа к информации - перехват и искажение. Возможность перехвата информации означает получение к ней доступа, но невозможность ее модификации. Примером перехвата информации может служить прослушивание канала в сети.

Возможность искажения информации означает либо полный контроль над информационным потоком между объектами системы, либо возможность передачи сообщений от имени другого объекта.

При нарушении работоспособности системы основная цель атакующего - добиться, чтобы операционная система на атакуемом объекте вышла из строя, и для всех остальных объектов системы доступ к ресурсам атакованного объекта был бы невозможен.

Возможны и другие классификации удаленных атак, например: в зависимости от условий начала воздействия, в зависимости от наличия (или отсутствия) обратной связи, от расположения субъекта атаки и т. д.

2 Язык командного интерпретатора Shell и командный процессор BASH

2.1 Unix системы и Shell разработки

При работе пользователей с системой Unix происходит процесс с одним из множества видов интерпретаторов команд. Интерпретатор команд shell, sh является программной UNIX, она же в свою очередь обладает большими возможностями, как командный язык. Каждый вызов данного интерпретатора называется оболочной, любая оболочка в свою очередь выполняет единственную функцию – считывание с устройства ввода команды, и ее последующее выполнение. Так как Shell дает пользователю возможность общения с операционной системой на языке высокого уровня, Unix же в свою очередь может выполнять задачи, которые недоступны менее сложным ОС. Благодаря Shell в Unix-системах доступны такие привилегии, как:

1) объединение для создания новых команд;
2) передача позиционных параметров;
3) добавление или переименование объектов пользователем;
4) выполнение операций внутри циклов, либо же выполнение операций по определенному условию;
5) выполнение команд исключительно локально, без риска конфликта с командами других пользователей;
6) выполнение команд в фоновом режиме.

Shell это командный интерпретатор, который используется в операционных системах из семейства Unix, при помощи данного интерпретатора пользователь имеет возможность обращаться к операционной системе, отдавать ей команды, запускать существующие скрипты, которые в свою очередь состоят из списка команд. Существует большой количество разновидностей Shell, однако в народе самым лучшим проявил себя sh. SH – является оригинальным шеллом Борна, в него входят ksh, pdksh, bash. BASH является самой оптимизированной разновидностью и по умолчанию его использовать можно почти по всех операционных системах семейства UNIX. BASH – это более совершенная версия командной оболочки Shell. BASH пользуется популярностью, по сравнению с другими современными типами командных оболочек UNIX. В

BASH представляет из себя некий командный процессор, который работает в интерактивном режиме в текстовом окне. Благодаря BASH можно запускать скрипты, используя команду sh и далее название вашего файла со скриптами. Как и все UNIX подобные оболочки, SHELL поддерживает автодополнение названий файлов и каталогов, он имеет возможность подставлять вывод результата команд, переменных, может обеспечить контроль на порядком выполнения, операторами ветвления и циклами.

2.2 Команды и переменные shell

Традиционный метод использования - это ввод простых команд. Простая команда – это любая очередность аргументов, которые разделены знаками табуляции или же пробелами. Первый аргумент узнает имя команды, которая по идее будет выполнена. Остальные аргументы, за исключениями, передаются команде с нулевым номером. Например, чтобы вызвать печать на принтер файлов с именами alibek, bayan и vitya можно дать следующую команду:

```
lpr alibek bayan vitya
```

В случае, если нулевой аргумент команды несет в себе определение имени файла, который помечен выполняемым, и на самом деле является скомпилированной программной, то в таком случае Shell создает процесс, который в свою очередь немедленно начинает выполняет данную скомпилированную программу. В противном случае Shell в свою очередь, запускает еще один интерпретатор команд, с целью прочитать файл и выполнить команды, которые в нем содержатся. В Shell существует множество способов объявления переменных. Переменная – имя, которому присвоили какое-либо имя. Оставшиеся типы переменных – это лишь имена, которым разработчик, либо же сам интерпретатор присвоил какие-либо текстовые значения. Также Shell распознает и буквенно-цифровые переменные, которым были присвоены текстовые значения. Самый простой вид присвоения имеет вид:

```
Name=string
```

После процесса присвоения, при указании $имя в исходном коде, на экран будет выводиться значение строки, которое задал разработчик. Пробелы вокруг знака равенства категорически запрещены. Основные параметры таким способом определить нельзя. Эти параметры можно установить исключительно командой set.

В команде может быть более одного присвоения, необходимо понимать, что Shell присваивает значения переменным справа налево.

Таким образом, результат выполнения будет выглядеть следующим образом:

```
A=$B  B=abc
```

Ниже приведен пример элементарных присвоений. При написании присвоений есть возможность использовать двойные кавычки, знаки табуляции и т.д. Также следуют указать, что при обнаружении позиционных параметров и других названий значений, перед которыми есть $, то в таком случае они будут заменены на другие значения. Одинарные кавычки подавляют подстановку переменных:

```
MAIL=/usr/mail/gas
Echous="echo $1 $2 $3 $4"
```

```
stars=*****
asterisks='$stars'
```

В примере выше разработчиком была указана переменная echous, в которой есть значение echo и обращения к переменным с названием, в начале которых есть $. Вокруг переменной stars кавычки не нужны, так как это правило не распространяется на такой тип данных. Переменной с именем asterisks присвоили "$stars", а не "*****", т.к. там одинарные кавычки.

При выполнении команд по подстановке пробелы повторно не замещаются, поэтому переменные $first и $second в примере ниже имеют одинаковые значения:

```
first='a string with embedded spaces'
second=$first
```

В процессе обращения к переменным есть возможность заключать их имена в фигурные скобки, это нужно для того, чтобы отделить текстовые символы от цифр, однако, если в значении переменной будет указано обращение к другой переменной, то фигурные скобки необходимы.

```
a='This is a string'
echo "${a}ent test of variables."
```

Shell поддерживает множество переменных. Большинство из них уже изначально заложены в сам интерпретатор, но все они могут быть изменены пользователем.

Присваиваем программе login каталог загрузки пользователя, другими словами, присваивает значение каталога, который на данный момент времени является текущим, аналогом является известная всем команда cd. Использование этой переменной позволяет не следить за полными именами процедур Shell.

IFS переменная может определить символы, которые в свою очередь являются внутренними разделителями полей. В частности, они используются для интерпретации.

Переменная MAIL несет в себе важнейший смысл, она содержит в себе полные пути файлов, куда и откуда приходят письма по почте. В случае, если переменная MAIL установлена, то интерпретатор проверяет, было ли что-то добавлено в указанный файл и уведомляет об этом пользователя. Переменная MAIL по умолчанию не настроена автоматически, ее необходимо установить в пользовательском интерфейсе, в файле .profile.

Переменная MAILCHECK задает частоту проверки файлов на наличие в них новых записей, по умолчанию установлено значение 600 секунд, если же установить 0, то интерпретатор будет проверять наличие почты каждый раз при появлении символа приглашения.

Переменная MAILPATH отвечает на список имён файлов, которые в свою очередь разделены двоеточием. При правильной настройке, при

произведении каких-либо изменений будет выдаваться автоматическое уведомление «u have a new mail»

Если в SHACCT параметр пользователем были занесены имя файла, доступного пользователю для записи, интерпретатор будет заносить в него соответствующие записи для выполняемых процедур интерпретатора.

SHELL Когда вызывается интерпретатор Shell, он ищет эту переменную и если она имеется и в ее значении вместо имени файла указан символ 'r', то Shell загружается в укороченном виде.

PATH Эта переменная определяет пути, которые использует Shell при поиске команд. Ее значение представляет собой список имен каталогов, разделенный двоеточиями. Shell присваивает переменной PATH значение :/bin:/usr/bin, где перед первым двоеточием ничего не указано. Пустой аргумент в любом месте в списке путей всегда обозначает текущий каталог. В некоторых системах поиск в текущем каталоге по умолчанию отсутствует, и поэтому переменная PATH определяется как /bin:/usr/bin. Если вы хотите, чтобы текущий каталог при поиске команд просматривался последним, а не первым, используйте: PATH=/bin:/usr/bin:

2.3 Среда выполнения команд

Абсолютно каждые переменные, что известны команде перед выполнением, образуют ее оболочку (среду выполнения). В этой же оболочке интерпретатора значения переменных имею возможность передавать свои параметры «по наследству»

Интерпретатор передает переменные дочернему процессу - это переменные, перечисленные как аргументы в команде export. Эта команда помещает указанные переменные в оболочки как самого Shell, так и всех его будущих дочерних процессов.

Ключевым параметром является пара переменная-значение, которые появляются в форме присвоений, обычно перед именем процедуры в командной строке. Такие переменные помещаются в оболочку процедуры, что вызвал сам пользователь. Например:

```
#keycommand
echo $a $b
```

Демонстрация примера, что выводит значение двух процедур:

```
a=key1 b=key2 keycommand
```

то результирующий вывод будет:

```
key1 key2
```

Ключевые параметры не могу быть аргументами $#.

В разрабатываемом программном обеспечении процедуры имеют доступ к любым переменным, это является особенностью интерпретатора. Однако, даже если значение переменной изменили, эти изменения не

отражаются на параметрах Shell, они относятся только к конкретной процедуре. Для того, чтобы эти изменения были переданы в дочерние процессы данной процедуры, они должны быть перечислены в команде export в этой процедуре. Чтобы получить список переменных, которые являются передаваемыми из текущего интерпретатора Shell, пользуемся этой командой:

```
Export
```

2.4 Управление потоком выполнения

Интерпретатор имеет несколько команд, которые используют множество управляющих структур, полезных при управлении потоком выполнения процедур. Перед тем как описывать эти структуры, необходимо определить некоторые термины.

Простая команда - это любая неделимая команда, определяемая именем выполняемого файла. Аргументы переадресации ввода-вывода могут указываться в простых командах, и они передаются интерпретатору, а не самой команде.

Команда - это простая команда или любая из управляющих команд Shell, описанных ниже. Конвейер - это последовательность из одной или более команд, разделенных вертикальными чертами (|). В конвейере стандартный вывод каждой команды, кроме последней, соединен посредством программного канала со стандартным вводом следующей команды. Каждая команда в конвейере выполняется отдельно; Shell ждет завершения последней команды. Код завершения конвейера равен коду завершения последнего процесса в конвейере. В таблице 2.1 указан перечень основных команд ввода-вывода.

Таблица 2.1 – Перечень основных команд, отвечающих за ввод-вывод

Ввод-вывод	
Echo	Вывод содержимого какой-либо переменной, или просто вывод выражения
Printf	Отвечает за форматированный вывод, является расширенной функцией echo
Read	Если использовать в интерактивном режиме, то является таблицей

Список команд - это последовательность из одного или более конвейеров, разделенных точками с запятой (;), амперсандами (&), символами "и-если" (&&) или "или-если" (||) и заканчивающаяся (необязательно) точкой с запятой или амперсандом. Точка с запятой вызывает последовательное выполнение предыдущего конвейера. Это значит, что Shell ждет конца выполнения конвейера и только после этого считывает следующий. С другой стороны, амперсанд вызывает асинхронное выполнение в фоновом режиме предыдущего конвейера. Таким образом допускаются как последовательное, так и фоновое выполнение. Фоновый конвейер продолжает выполнение до тех пор, пока не завершится самостоятельно или будет уничтожен.

Другие области применения амперсанда включают в себя фоновую компиляцию и создание заданий, которые будут пересылаться другим компьютерам. Допустим, вы ввели:

```
nohup cc prog.c&
```

Вы можете продолжать работать в то время как Си-компилятор выполняется в фоновом режиме. Командная строка, заканчивающаяся амперсандом, защищена от прерываний и выхода, которые вы указываете, набирая с клавиатуры INTERRUPT или QUIT. Только одновременное нажатие клавиш Ctrl-d может снять команду в случае, если вы работаете через коммутируемую линию или имеете stty hupcl. В этом случае мы также советуем устанавливать защиту от прерываний. Для этого используется команда nohup. Допустим, что в приведенном примере вы не указали nohup, тогда если вы выходите из подзадачи до того, как завершится процесс cc, он будет уничтожен со всем своим выводом.

Амперсанд нужно применять с ограничениями, особенно в сильно загруженных системах. В противном случае возникнут претензии со стороны других пользователей, если вы будете запускать много процессов в фоновом режиме без достаточных на это оснований.

Операторы "и-если" и "или-если" (&& и ||) вызывают выполнение конвейеров при соответствии определенных условий. Оба эти оператора имеют одинаковый приоритет при расшифровке командной строки, но меньший, чем у амперсанда и вертикальной черты. В командной строке:

```
cmd1 || cmd2
```

Первая команда cmd1 выполняется и анализируется ее код завершения. Команда cmd2 выполняется только в том случае, если cmd1 завершилась с кодом, не равным 0. Т.е. это является краткой записью следующих команд:

```
if cmd1 test $? !=0 then cmd2 fi
```

Оператор "и-если" (&&) выполняет проверку на равенство. Например, в командной строке:

```
cmd1 && cmd2
```

Вторая команда выполняется только в том случае, если первая завершилась успешно (код завершения равен 0). В следующей строке каждая команда выполняется по очереди до тех пор, пока какая-нибудь не закончится с кодом, не равным 0:

```
cmd1 && cmd2 && cmd3 && ... && cmdn
```

Сценарии командной строки – это наборы тех же самых команд, которые можно вводить с клавиатуры, собранные в файлы и объединённые некоей общей целью. При этом результаты работы команд могут представлять либо самостоятельную ценность, либо служить входными данными для

других команд. Сценарии – это мощный способ автоматизации часто выполняемых действий.

Итак, если говорить о командной строке, она позволяет выполнить несколько команд за один раз, введя их через точку с запятой:

```
pwd ; whoami
```

На самом деле, если вы опробовали это в своём терминале, ваш первый bash-скрипт, в котором задействованы две команды, уже написан. Работает он так. Сначала команда pwd выводит на экран сведения о текущей рабочей директории, потом команда whoami показывает данные о пользователе, под которым вы вошли в систему.

Используя подобный подход, вы можете совмещать сколько угодно команд в одной строке, ограничение — лишь в максимальном количестве аргументов, которое можно передать программе. Определить это ограничение можно с помощью такой команды:

```
getconf ARG_MAX
```

Командная строка – отличный инструмент, но команды в неё приходится вводить каждый раз, когда в них возникает необходимость. Что если записать набор команд в файл и просто вызывать этот файл для их выполнения? Собственно говоря, тот файл, о котором мы говорим, и называется сценарием командной строки.

Создайте пустой файл с использованием команды touch. В его первой строке нужно указать, какую именно оболочку мы собираемся использовать. Нас интересует bash, поэтому первая строка файла будет такой:

```
#!/bin/bash
```

В других строках этого файла символ решётки используется для обозначения комментариев, которые оболочка не обрабатывает. Однако, первая строка – это особый случай, здесь решётка, за которой следует восклицательный знак (эту последовательность называют шебанг) и путь к bash, указывают системе на то, что сценарий создан именно для bash.

Команды оболочки отделяются знаком перевода строки, комментарии выделяют знаком решётки. Вот как это выглядит:

```
#!/bin/bash
# This is a comment
pwd
whoami
```

Тут, так же, как и в командной строке, можно записывать команды в одной строке, разделяя точкой с запятой. Однако, если писать команды на разных строках, файл легче читать. В любом случае оболочка их обработает.

В bash также есть возможность работы с массивами. При работе с ними часто пользуются переменной окружения IFS — разделителя полей для входных строк (IFS — Input Field Separator). По умолчанию IFS равен символу

пробела, но может быть изменен для разбиения строки на элементы массива, например, запятыми. Обратите внимание, что для формирования переменных оболочки, которые доступны через $ 1, $ 2и т.д., используется именно переменная IFS, то есть введенная после имени скрипта строка аргументов будет разделена именно первым символом, который хранится в этой переменной.

Объявить массив можно следующим образом:

```
files[0]=Яблоко
files[1]=Груша
echo ${files[*]} # напечатает элементы массива без учета
IFS echo ${files[@]} # напечатает элементы массива с IFS в качестве
разделителя
```

Получить доступ к элементу массива можно с помощью срезов: $ {arr: 0: 1}. Удалить первый элемент массива можно с помощью сдвига: shift arr. Добавить в элементы в массив: arr = ("$ {arr [@]}" "Item 1" "Item 2"). Проверка вхождения элемента в массив реализуется с помощью несколько более сложной конструкции:

```
if [[ ${arr[(r)some]} == some ]]; then # команды,если элемент входит
else # команды, если не входит
fi
```

В этом примере arr — некоторый массив, a some — это элемент, который мы проверяем на вхождение. В таблице 2.2 указаны основные команды для работы и перемещению по файловой системе.

Таблица 2.2 – основные команды для работы с файловой системой

Файловая система	
Cd	Изменение текущего каталога
Pwd	Вывод текущего каталога
Pushd	Изменение текущего каталога, но при этом с сохранением возможности возвращения в текущем порядке.
Popd	Возврат текущего каталога после pushd
Dirs.	Вывод и очистка содержимого стека каталогов

2.5 DNSBL и методы использования

Существует более 300 публичных чёрных списков, в которые могут попасть IP-адреса и домены. Создать блеклист может кто угодно – как крупные надёжные компании, так и небольшие независимые сети. Но не все списки одинаково влияют на доставляемость. Почтовые провайдеры и фильтрационные программы не проверяют каждый из них, они объединяют данные из разных публичных блеклистов и собственные данные, чтобы определить репутацию отправителя.

Емейл-маркетологи часто связывают блокировку писем у почтового провайдера с попаданием в чёрные списки. Но это не одно и то же. Два этих события может объединять одна причина – например, часто такая ситуация

связана с плохим качеством базы, то есть большим числом несуществующих ящиков и жалоб от пользователей.

Существует 2 типа чёрных списков: содержащие IP-адреса и домены.

Первый тип работает в режиме реального времени и называются Real-time Black Lists и Domain Name Server Black Lists.

Самые известные RBL/DNSBLs-листы:

- Return Path Reputation Network Blacklist (RNBL) – этим листом IP-адресов владеет Return Path. Компания сама определяет, блокировать ли рассыльщика, на основании своих данных из Return Path Provider Network. Алгоритм сложный: это и модели предсказаний, и данные о спам-ловушках, и жалобы.

- Sbl.spamhaus.org (SBL) – это база IP-адресов, от которых крупнейшая в мире организация по борьбе со спамом Spamhaus не рекомендует принимать письма. SBL включает спамеров, спам-операции и сервисы, через которые отправляется спам. Если вы просите Spamhaus удалить вас из их блеклиста, в ответ они потребуют план действий, как исправить проблему, которая вызвала попадание в чёрный список.

- Xbl.spamhaus.org (XBL): Exploits Bot List (XBL) – это список известных открытых источников и нелегальных сторонних эксплоитов для отправки спама и вирусов. Эксплоит – компьютерная программа, атакующая вычислительные системы через уязвимости. XBL включает в себя информацию Spamhaus.

- Cbl.abuseat.org (CBL): Spamhaus Composite Blocking List (CBL) – это блэклист, основанный на DNS-записях и оперирующий IP-адресами, которых подозревают в отправке спама и заражении вирусным ПО. CBL получает эти данные от крупных почтовых серверов и их спам-ловушек. У CBL есть простой вариант самостоятельного удаления IP-адресов.

- SpamCop (SCBL) – это сервис спам-отчётов. Частные лица могут написать в спамкоп и зарепортить массовую коммерческую рассылку, которую не ждали, как спамерскую. Удаление из SCBL происходит автоматически через 24 часа, если не было повторного запроса на добавление в лист.

- Psbl.surriel.com: Passive Spam Block List (PSBL) – список IP-адресов, с которых отправляются письма на спам-ловушки. У PSBL существуют белые списки, и если IP-адрес находится в таком списке, то вероятность попадания в блеклист снижается.

Invaluement: Invaluement Anti-Spam DNSBL – это компилятор трёх коммерческих анти-спам блеклистов:

ivmURI – содержит домены, принадлежащие спамерам;

ivmSIP – IP-адреса ботов, неуловимых спамеров или мошенников, которые пропускаются Spamhaus или ещё не добавлены туда;

ivmSIP/24 – диапазон IP-адресов, в которых обнаружены модели поведения спамеров.

Второй тип, это работающие в режиме реального времени чёрные списки ссылок, содержащихся в теле письма.

- Dbl.spamhaus.org: Spamhaus DBL – это база доменов с низкими репутациями, которые обнаружены в спамерских рассылках. Списки Spamhaus DBL поддерживают команда специалистов и автоматическая система – они непрерывно анализируют мировой поток писем со спамом.

URIBL – это список доменов, которые замечены в спамерских письмах. Сейчас у сервиса есть несколько публичных листов, наиболее популярный из которых black.uribl.com. Список обновляется по мере поступления новых данных, поэтому удаление из него может происходить автоматически.

Владелец домена может также отправить запрос на удаление из чёрного списка после регистрации на сайте.

- SURBL – это список сайтов, которые появились в спам-сообщениях. Владелец домена может запросить удалить домен из блеклиста, выполнив условия инструкции по удалению.

Обычная реакция отправителя после попадания в чёрный список – сразу же требовать удаления из него (делист). Но иногда это может больше навредить, чем помочь. Если отправитель часто запрашивает удаление и не предпринимает ничего, чтобы исправить ситуацию, он рискует попасть в немилость, после чего все его последующие запросы будут отклоняться автоматически.

По нашему опыту, попадание доменов или IP-адресов в чёрные списки несильно влияет на доставляемость у российских провайдеров и редко приводит к блоку рассылок на них. Чего не скажешь о зарубежных провайдерах – Gmail, Yahoo и т.д., поэтому мы попросили поделиться опытом взаимодействия с ЧС наших польских коллег. По их словам, при попадании IP-адреса или домена в чёрный список первым делом нужно проверить:

- последнее отправленное письмо (наличие спам-слов, ссылки отписки, причины получения письма, количество изображений по отношению к тексту);
- был ли массовый импорт новых подписчиков в базу.

При этом все рассылки на время прекращаются во избежание ещё больших проблем.

Параллельно нужно отправить запрос на делист IP-адреса/домена из чёрного списка. Это должен сделать владелец домена или IP, так как в некоторых случаях потребуется подтверждение владением. Обычно IP-адреса автоматически удаляют из чёрного списка через 3 дня, если не было новых нарушений. Для доменов может потребоваться больше времени – до двух недель. Однако даже по истечении этого срока делиста может не произойти, если отправитель не предпринимает ничего, чтобы исправить проблему.

Использование сервисов, которые основаны на технологии DNSBL, позволяют достичь эффективности фильтрации писем, содержащих спам и вредоносное ПО в 95-96,8% случаев. Недостаток данных черных списков

состоит в том, что, видимо, туда могут попасть по ошибке и вполне безопасные IP адреса, если они пропустили через себя спам, разосланный каким-либо компьютером внутри своей сети.

Согласно информации по тестированию, в котором происходит тест эффективности наиболее популярных антивредоносных средств, ни одно из писем, не являющееся спамом, либо рассылающим вредоносное ПО, не было ошибочно определено как спам.

Решения, что были основаны на использовании DNSBL-списков, большинство квалифицированных специалистов считают одними из самых надёжных. По этому поводу функциональные возможности Smart Message Transfer Protocol-шлюза и является дополнением к модулю RBL. Его принцип работы основан на доскональной проверке IP-адреса, принимаемого сообщения в RBL-службах путём отправки на них DNS-запросов. RBL-модуль почтового сервера в момент приёма сообщения запрашивает RBL-сервис, является ли IP-адрес отправителя письма «плохим» и на основании ответа RBL принимает или отвергает письмо.

Опасна массовая рассылка и тем, что таким образом, рассылаются письма, прямо или косвенно побуждающие получателя посещать фишинговые сайты. Согласно опросу, электронная почта занимает первое место среди потенциально опасных каналов утечки данных:

Мошенники рассылают письма якобы от лица крупных компаний, письма эти очень похожи на настоящие, в содержании содержится ссылка, перейдя по которой, на рабочую станцию человека либо попадет вредоносный объект, либо человек попадет на страницу, на которой его под каким-то предлогом, который кажется очень выгодным человеку, ввести данные от своей корпоративной почты, данные банковской карты, пароли от других своих личных сервисов и так далее.

В среднем фишинговые атаки заканчиваются успехом злоумышленников примерно в 40% случаях, а порядка 4% писем, которые получает Gmail (почтовый сервис Google Mail), разработаны специальным образом, чтобы выманить у людей их личные данные.

Эффективный способ обеспечить себе защиту от фишинговых сайтов это сервис Яндекс.DNS, который на данный момент доступен в большом количестве роутеров от ASUS, D-Link, TP-Link. При попытке открыть страницу с фишинговым содержанием, переход по ней блокируется, что-то вроде встроенного Proxy-сервера.

Большинство браузеров тоже имеют возможность блокировки фишинговых сайтов. Chrome, Firefox и Safari используют технологию Safe Browsing API, IE – Smart Screen.

Интересную систему защиты от фишинга «Protect» выпустил «Яндекс» чуть больше месяца назад. Protect отслеживает действия пользователя и следит, чтобы пароли не вводились на сайтах, похожих на известные сервисы.

Кроме того, технология Protect включает в себя проверку всех

загружаемых файлов. Функция Protect защищает личные данные пользователя при подключении к открытой сети Wi-Fi в общественных местах. Protect интегрирован в версии «Яндекс.Браузера» для Windows и OS X.

Функционально похожее расширение проверки паролей Password Alert есть в Chrome, однако, оно работает только на аккаунтах Google и Google Apps for Work.

Модуль защиты от фишинга Phishing Blocker в Traffic Inspector использует условно-бесплатный проект API Google Safe Browsing. Phishing Blocker проверяет URL на наличие угроз в обновляемом Google чёрном списке потенциально фишинговых сайтов и страниц. Если ответ положительный, то хост или IP-адрес приписывается к одной из предварительно созданных категории контента. Это позволяет предотвратить посещение пользователями заведомо мошеннических веб-ресурсов.

Ещё одна возможность Phishing Blocker – присвоение ресурсу определённого рейтинга, что позволяет произвести фильтрацию нежелательного контента, разрешить доступ только к контенту, имеющему доверие и получить отчеты по посещаемым ресурсам в соответствии с рейтингом.

Черные списки используют спам-ловушки и сведения о жалобах на спам для идентификации IP-адресов и доменов, отправляющих нежелательные рассылки. Спам-ловушки — это локальные email адреса, которые разбросаны на разных сайтах и форумах с целью выявления отправителей спама.

Спам-ловушки делятся на две категории:

Преобразованные email адреса — это адреса, которые неактивны больше года, из-за чего почтовые сервисы сделали из них спам-ловушки.

Классические ловушки — это специально созданные email адреса. Их размещают на популярных ресурсах, где можно спарсить базу адресов для дальнейшей продажи или рассылки по ней спама.

У всех черных списков разные способы вычисления спамеров. Но почти все они применяют некоторую комбинацию спам-ловушек и данных о жалобах на спам.

Операторы черных списков имеют большую сеть спам-ловушек. Они оставляют специальные email адреса на популярных сайтах и форумах, отслеживают их и вносят в черный список любые IP-адреса или домены, которые незаконно парсят ловушки и делают по ним рассылки.

Оператор черного списка может инициировать добавление в черный список IP-адреса, если на него идет чрезмерный поток жалоб за рассылку спама. Чтобы избежать черных списков, нужно обеспечить отправление рассылок на валидные и активные адреса, собранные самостоятельно.

Практически все почтовые сервисы, в том числе и хостинг-провайдеры, используют такие списки для фильтрации спама. При получении письма, почтовый сервер сначала проверяет IP адрес отправителя на наличие в черных списках, и только после этого принимает решение, спам это или не спам. При

наличии IP адреса в таких списках, почта с этого IP может просто отклонятся или отправляются в папку спам, это зависит от настроек почтового сервера получателя. Соответственно, если вы используете почту от вашего хостинга, наличие IP адреса его почтового сервера в популярных DNSBL, гарантирует вам проблемы с отправкой почты. Поэтому хостеры и почтовые сервисы следят за использованием своей почты и блокируют спамеров при обнаружении.

По описанию процесса сбора доказательной базы по инцидентам информационной безопасности можно сделать вывод, что это очень рутинная и долгая работа, и есть необходимость и возможность автоматизировать данный процесс.

Анализ сети, возможность блокировки, просмотр соединений, количества запросов с одного IP будет проводиться системой, которые в свою очередь поддерживают специализированные сервисы. Данные сервисы будут обеспечивать полную «прозрачность» сети, контроль приложений, фильтрацию URL-адресов, и даст возможность применить усовершенствованную защиту от вредоносного ПО. За контролем файловой системы на рабочих станциях и банкоматах будет отвечать антивирусное ПО, данное ПО будет предотвращать переход по вредоносным ссылкам, своевременным удалением вредоносного ПО, и проведением плановых проверок на рабочих станциях во внерабочее время, либо пока рабочая станция находится в «спящем» режиме.

За контролем работы и наблюдением за файловой системой серверов может отвечать антивирусное программное обеспечение для серверов.

Все вышеперечисленные программные обеспечения будут выгружать из своих логов события, относящие к категориям возможной попытки проникновения в систему, либо успешного проникновения в систему в одну систему, за которой и будет проводиться наблюдение. К примеру система мониторинга за подключениями к внутренней сети предприятия обнаружила попытку подключения IP, который по своим базам имеет плохую репутацию, и автоматически отбросила данное подключение, именно эту попытку автоматизированная система выгрузила из логов фаервола и вывела ее на экран пользовательского интерфейса в системе. Такая же ситуация и с антивирусным ПО, которое к примеру, обнаружило вредоносное ПО на рабочей станции и произвел определенные действия (удалил/поместил в карантин/вылечил) и вывел именно это событие и результат своих действий на экран пользователя.

В свою очередь, пользователь автоматизированной системы сбора информации со всех ПО, которые производят слежку за состоянием системы, должен будет просматривать все, поступившие к нему инциденты информационной безопасности и проводить действия по отношению к ним.

Логика работы разрабатываемого приложения:

Сбор и перехват: система должна будет перехватывать любые события, и выявлять в них утечку информации и неправомерные действия персонала.

Анализ: система должна будет детектировать наличие конфиденциальных данных в облаке событий.

Автоматическое принятие решений (блокировка/пропуск события): система должна будет автоматически принимать решение по инциденту (блокировать либо обратное), это необходимо в целях оперативности и своевременному предотвращению утечки информации.

3 Разработка программного обеспечения по сбору доказательной базы на SHELL используя BASH

#!bin/sh сообщает родительской оболочке, что интерпретатор должен использоваться для выполнения script. Дальше идет процесс создания различных переменных. Переменные SCRIPT_NAME, VERSION, DESCRIPTION, AUTHOR, EMAIL, VK, INSTAGRAM и LICENSE будут использоваться при запуске скрипта с параметром --version. Наглядно показано на рисунке 3.1

```
#!/bin/sh
#
#
SCRIPT_NAME='Дипломная работа'
VERSION='0.3.5'
DESCRIPTION='Дипломная работа, проверка IP в базах на вредонос. Выполнил Дифу Идрис.'
AUTHOR='Difu Idris AUPET'
EMAIL='df.idris@gmail.com'
VK='vk.com/waless'
INSTAGRAM='http://instagram.com/df.idris'
LICENSE='Ubuntu.com'
SED='/bin/sed'
WGET='/usr/bin/wget'
HOST='/usr/bin/host'
ICONV='/usr/bin/iconv -t UTF-8 '
PING='/bin/ping -c 1 '
```

Рисунок 3.1 – Обращение к родительской оболочке и начало разработки

На рисунке 3.2 происходит процесс задачи списка DNSBL, которые в дальнейшем будут использоваться для проверки в разрабатываемом программном обеспечении.

```
DNSBL_LIST='abuse.rfc-ignorant.org access.redhawk.org aspews.ext.sorbs.net b.barracudacentral.org
blackholes.brainerd.net blackholes.five-ten-sg.com blackholes.wirehub.net
blacklist.junkemailfilter.com blacklist.sci.kun.nl blacklist.woody.ch bl.deadbeef.com
bl.emailbasura.org block.dnsbl.sorbs.net bl.redhatgate.com bl.spamcannibal.org
bl.spamcop.net bl.technovision.dk cl0.rbl.hk cbl.abuseat.org cbl.anti-spam.org.cn cblless.anti-spam.org.cn
cblplus.anti-spam.org.cn combined.njabl.org db.wpbl.info
dialups.mail-abuse.org dialups.visi.com dnsbl-0.uceprotect.net dnsbl-1.uceprotect.net dnsbl-2.uceprotect.net
 dnsbl-3.uceprotect.net dnsbl.ahbl.org dnsbl.cyberlogic.net
dnsbl.jammconsulting.com dnsbl.kempt.net dnsbl.njabl.org dnsbl.sorbs.net duinv.aupads.org dul.dnsbl.sorbs.net
 dul.ru fl.chickenboner.biz hil.habeas.com hostkarma.junkemailfilter.com
http.dnsbl.sorbs.net http.opm.blitzed.org images.rbl.msrbl.net ips.backscatterer.org ircbl.ahbl.org ix.dnsbl.manitu.net
 korea.services.net l2.bbfh.ext.sorbs.net list.dnswl.org
 mail-abuse.blacklist.jippg.org map.spam-rbl.com misc.dnsbl.sorbs.net msgid.bl.gweep.ca multi.surbl.org multi.uribl.com
 no-more-funn.moensted.dk ohps.dnsbl.net.au omrs.dnsbl.net.au
 orid.dnsbl.net.au orvedb.aupads.org osps.dnsbl.net.au osrs.dnsbl.net.au owfs.dnsbl.net.au owps.dnsbl.net.au pbl.spamhaus.org
 phishing.rbl.msrbl.net probes.dnsbl.net.au
 proxy.bl.gweep.ca psbl.surriel.com query.bondedsender.org rbl-plus.mail-abuse.org rbl.snark.net rdts.dnsbl.net.au relays.bl.gweep.ca
 relays.bl.kundenserver.de relays.mail-abuse.org
 relays.nether.net ricn.dnsbl.net.au rmst.dnsbl.net.au rot.blackhole.cantv.net rsbl.aupads.org satos.rbl.cluecentral.net sbl.csma.biz
 sbl.spamhaus.org sbl-xbl.spamhaus.org
 smtp.dnsbl.sorbs.net socks.dnsbl.sorbs.net socks.opm.blitzed.org sorbs.dnsbl.net.au spam.dnsbl.sorbs.net spamguard.leadmon.net
 spam.olsentech.net spamrbl.imp.ch
 spamsites.dnsbl.net.au spamsources.dnsbl.info spamsources.fabel.dk spam.wytnij.to tl.bl.dnsbl.net.au tl.dnsbl.net.au ubl.unsubscore.com
 ucepn.dnsbl.net.au virbl.bit.nl
 virbl.dnsbl.bit.nl virus.rbl.jp virus.rbl.msrbl.net web.dnsbl.sorbs.net whois.rfc-ignorant.org wingate.opm.blitzed.org wormrbl.imp.ch
 wpbl.dnsbl.net.au xbl.spamhaus.org
 zen.spamhaus.org zombie.dnsbl.sorbs.net'
```

Рисунок 3.2 – Внедрение DNSBL адресов в разрабатываемое программное обеспечение

Переменная DNSBL_LIST содержит в себе адреса обращения к черным спискам DNSBL. Дальше процесс настройки параметров запуска ПО, где:
-q - Ничего не выводить
-l - Вывести список dnsbl-серверов
-d - Указатель, что надо проверить по указанному DNSBL
-f - Указатель, что используется файл
-i - Сграбить список DNSBL.

На рисунке 3.3 наглядно продемонстрирован процесс задачи параметров для параметров запуска разрабатываемого программного обеспечения.

```
# Параметры по умолчанию
IS_QUIET=0              # -q    - Ничего не выводить
IS_DNSBL_PRINT=0        # -l    - Вывести список dnsbl-серверов
IS_ONE_DNSBL=0          # -d    - Указатель, что надо проверить по указанному DNSBL
IS_FILE=0               # -f    - Указатель, что используется файл
IS_INET=0               # -i    - Сграбить список DNSBL с сайта http://myiptest.com/
IP_LIST=''              #       - Тут будут все IP'шники
IS_TIME_OUT_ERROR=0     #SS -t  - Считать тайм ауты подключения к DNSBL ошибками и выводить в поток ошибок?
DNS_SERVER='208.67.222.222'     # -s    - DNS Server (OpenDNS.com), через который будет проверяться IP по DNSBL
RESOLVCONF='/etc/resolv.conf'
```

Рисунок 3.3 – Задание параметров запуска, которые будут использоваться в разрабатываемом программном обеспечении

На рисунке 3.4 произведена настройка режима вывода на консоль прогресса выполнения скрипта.

```
#  name     Конвертирует строку в текущую локаль из UTF-8 и выводит строку на стандартный поток либо поток ошибок
#  $1       Строка, котору надо вывести (надо заключать в кавычки, если есть пробельные символы)
#  $2       Куда выводить (stduot (1)  или stderr (2))
#  output   Возвращает указанную строку ($1) выводя ее на stdout или stderr
#  return   0
_output() {
    # Если обычный режим, то выводим инфу на консоль, иначе просто все игнорируем
    if [ $IS_QUIET -eq 0 ] ; then
        if [ $1 -eq 2 ] ; then
            echo "$2" | $ICONV 1>&2
        else
            echo "$2" | $ICONV
        fi
    fi
}

#  name     Версия, копирайт, лицензия, контакты
#  output   --//--
#  return   0
_version() {
    _output 1 "$SCRIPT_NAME, version $VERSION"
    _output 1 "Copyright $AUTHOR"
    _output 1 "License $LICENSE"
    _output 1 "Contacts: $EMAIL | $VK | $INSTAGRAM"
}

#  name     "Правила" использования скрипта
#  output   --//--
#  return   0
_usage() {
    _output 1 "$SCRIPT_NAME [-q] [-l] [-d dnsbl | -f filename | -i] [-s dns_server] [-t] [-w] [--] [ip or domain-name ...]"
}
```

Рисунок 3.4 – Процесс проверки работоспособности заданных параметров запуска

На рисунке 3.5 продемонстрирован процесс настройки вывода на экран всех существующих параметров запуска при запуске скрипта с параметром запуска –help.

```
92  #  name     Справка по использованию параметров скрипта
93  #  output   --//--
94  #  return   0
95  _help() {
96      _usage
97      _output 1 ""
98      _output 1 "   -q        - Ничего не выводить"
99      _output 1 "   -l        - Показать список DNSBL-серверов"
100     _output 1 "   -d        - Проверить по указанному DNSBL"
101     _output 1 "   -f        - Взять список DNSBL-серверов из файла (по одному на строчку)"
102     _output 1 "   -i        - Спрабить список DNSBL с сайта http://myiptest.com/"
103     _output 1 "   -s        - Указать DNS-сервер, через который будет идти проверка"
104     _output 1 "   -t        - Считать Time Out'ы ошибкой"
105     _output 1 "   --        - Считается, что дальше идут только ip и/или доманные имена"
106     _output 1 "   [ip or domain-name ...]   - ip и/или доменный имена, которые нужно проверить"
107     _output 1 "   --version - Версия ПО"
108 }
109
110 #  name     Сообщение о неизвестном/неверном параметре переданном скрипту
111 #  output   --//--
112 #  return   0
113 _bad_param() {
114     _output 2 "Неизвестный параметр: $1"
115 }
```

Рисунок 3.5 – Процесс вывода на экран списка параметров запуска по запросу пользователя

На данном этапе разработки идет окончательная настройка параметров запуска ПО, вывод на экран текстовых подсказок, также добавились параметры запуска –version, и при запуске с параметром –h пользователю на экран выводится меню со всеми возможными параметрами запуска.

На рисунке 3.6 происходит проверка введенного IP адреса на валидность, в случае, если введенное значение не является IP адресом, то программное обеспечение закончит цикл и прекратит действия. Если же введенный IP адрес является действительно IP адресом, то программное обеспечение начнет выполнять проверку.

```
117 #  name     Проверяет, является ли переданый парамерт ip-адресом и, если да, заносит его в переменную $IP_LIST
118 #  $1       ip-адрес
119 #  output   none
120 #  return   0
121 _check_and_build_ip_list() {
122     CBL=$(echo "$1" | $SED -n '/[^0-9\.]\+/p')
123     # если пусто, значит считаем что у нас "IP"
124     if [ -z "$CBL" ] ; then
125         IP_LIST="$IP_LIST $1"
126     # иначе "доменное имя"
127     else
128         IP_LIST="$IP_LIST $($HOST $1 | $SED -n '/has\ address/s/^.\+\ \([0-9\.]\+\).*$/\1/gp')"
129     fi
130 }
```

Рисунок 3.6 – Процесс проверки введенного IP адреса и вывода на экран соответствующего сообщения

На рисунке 3.7 выполняется процесс парсинга параметров запуска программного обеспечения.

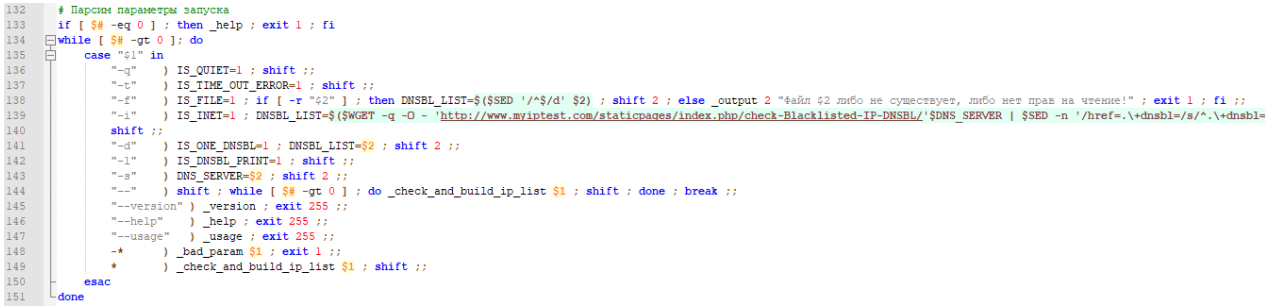

Рисунок 3.7 – Процесс парсинга информации по введенному IP с DNSBL

На рисунке 3.8 продемонстрирован процесс выполния условия, в зависимости от полученного результата выводится и необходимое сообщение на экран.

```
#  name     Выводит сообщение "Не совместимые параметры!"
#  output   --//--
#  return   0
_bad_params_combine() {
    _output 2 "Несовместимые параметры!"
    exit 1
}

# Проверка на (не)совместимость параметров
if [ $IS_DNSBL_PRINT -gt 0 -a $IS_QUIET -gt 0 ] ; then _bad_params_combine ; fi
if [ $IS_FILE -gt 0 -a $IS_INET -gt 0 ] ; then _bad_params_combine ; fi
if [ $IS_FILE -gt 0 -a $IS_ONE_DNSBL -gt 0 ] ; then _bad_params_combine ; fi
if [ $IS_ONE_DNSBL -gt 0 -a $IS_INET -gt 0 ] ; then _bad_params_combine ; fi

# Вывод списка DNSBL серверов, если на то был "запрос"
if [ $IS_DNSBL_PRINT -gt 0 ] ; then echo "$DNSBL_LIST" | $SED 's/\ /\n/g' | $SED '/^$/d' ; exit 0 ; fi

#  name     "Переворачивает" переданный ip-адрес и возвращает его
#  $1       ip-адрес
#  output   "Перевернутый" ip-адрес
#  return   0
_reverse_ip() {
    echo $1 | $SED -e 's/$/\n/;:z;s/\([0-9]\{1,3\}\)\([^\n]*\n\)/\2\1./;tz;s/.*\n//;s/.$//'
}
```

Рисунок 3.8 – Процесс общения пользователя с разрабатываемым программным обеспечением

На рисунке 3.9 получаем от DNSBL ответы, и в зависимости от того, какой ответ пришел, выводим пользователю сообщение с результатом его запроса на экран.

```
_get_and_parse_dnsbl_return() {
    for DNSBL in $DNSBL_LIST ; do
        GPD=$($HOST -t A $1.$DNSBL $DNS_SERVER)
        RES_A=$(echo "$GPD" | $SED -n '/not\ found/p')
        # Нас нет в DNSBL :)
        if [ -n "$RES_A" ] ; then
            _output 1 "[ OK ]         - $DNSBL      - $(echo $RES_A | $SED 's/^.*\(not\ found.\+$\)/\1/')"
        else
            RES_A=$(echo "$GPD" | $SED -n '/\ has\ address\ /s/^.*address\ \([0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\).*$/\1/p')
            # Черт... У нас проблемы :(
            if [ -n "$RES_A" ] ; then
                # Узнаем подробности
                RES_TXT=$($HOST -t TXT $1.$DNSBL $DNS_SERVER | $SED -n '/text/s/^.*\"\(.\+\)\".*$/\1/p')
                _output 2 "[ INFECTED!!!! ] - $DNSBL        - ($RES_A) $RES_TXT"
                RETURN_CODE=1
            else
                # Или может мы просто не можем достучаться до DNSBL-сервера?
                RES_A=$(echo "$GPD" | $SED -n '/connection timed out/p')
                if [ -n "$RES_A" ] ; then
                    if [ $IS_TIME_OUT_ERROR -gt 0 ] ; then
                        _output 2 "[ ERROR ]      - $DNSBL      - connection time out"
                        RETURN_CODE=1
                    else
                        _output 1 "[ ERROR ]      - $DNSBL      - connection time out"
                    fi
                else
                    _output 2 'Видимо структура сайта изменилась, напиши об этой ошибке мне на почту: df.idris@gmail.com'
                    RETURN_CODE=1
                fi
            fi
        fi
        done
        return $RETURN_CODE
}
```

Рисунок 3.9 – Заключительный этап разработки программного обеспечения

На данном этапе разработки программного обеспечения (рисунок 3.10) были созданы условия вывода результата на экран, при получении ответа с сайта, который содержит в себе информацию о том, что введенный IP-адрес находится в черном списке – выводится соответствующее сообщение. Если не программа получает ответ, который содержит в себе информацию о том, что введенный IP не находится в черном списке, программа выдает уже другой соответствующий ответ. Если же структура DNSBL сайта была изменена – выдается соответствующее сообщение о том, что определенный сервис по проверке более не поддерживается.

```
# IP указан?
if [ -z "$IP_LIST" ] ; then _output 2 "Не указаны адреса для проверки!" ; exit 1 ; fi

# Если есть в системе resolv.conf...
if [ -r $RESOLVCONF ] ; then
    # ...выдираем из него IP'шники DNS-серверов...
    NS=$($SED -n '/nameserver/s/nameserver[\ \t]\+\([0-9\.]\+\)/\1/p' $RESOLVCONF)
    # ...и проверяем их на доступность...
    for NS_CURRENT in $NS ; do
        # ...если хоть один доступен...
        $PING $NS_CURRENT >/dev/null 2>&1
        if [ $? -eq 0 ] ; then
            # ...то будем юзать его
            DNS_SERVER=$NS_CURRENT
            break
        fi
    done
fi  # В противном случае будет использоваться DNS-сервер по дефолту (см. переменную $DNS_SERVER в разделе "Параметры по умолчанию")

for IP in $IP_LIST ; do
    # Получаем "перевернутый" IP
    R_IP=$(_reverse_ip $IP)
    _output 1 "----------------------------------------$IP----------"
            # Проверяем наши IP по спам-базам и выводим результат на консоль
            _get_and_parse_dnsbl_return $R_IP
            EXIT_CODE=$?
done
exit $EXIT_CODE
```

Рисунок 3.10 – Разработка циклов для вывода результата сканирования на экран

Завершающие этап программы, закрытие всех циклов, что были необходимы при работе с программным обеспечением и вывод на экран полученных результатов

3.1 Произведение тестовой атаки на сервер

Перед тестированием разработанного программного обеспечения необходимо произвести тестовую атаку на какой-либо элемент сетевой инфраструктуры.

Для начала установим, что такое сетевая атака. Сетевая атака - действие, целью которого является захват контроля (повышение прав) над удалённой/локальной вычислительной системой, либо её дестабилизация, либо отказ в обслуживании, а также получение данных пользователей пользующихся этой удалённой/локальной вычислительной системой. На данный момент выделяют следующие атаки: mailbombing, переполнение буфера, использование специализированных программ (вирусов, снифферов, троянских коней, почтовых червей, rootkit-ов и т.д.), сетевая разведка, IP-спуфинг, man-in-the-middle, инъекция (SQL-инъекция, PHP-инъекция, межсайтовый скриптинг или XSS-атака, XPath-инъекция), отказ в обслуживании (DoS- и DDoS- атаки), phishing-атаки.

При проведении тестовой атаки было использовано программное обеспечение от стороннего разработчика. На рисунке 3.1 показан процесс добавления нового хоста в целях проведения сканирования на уязвимости.

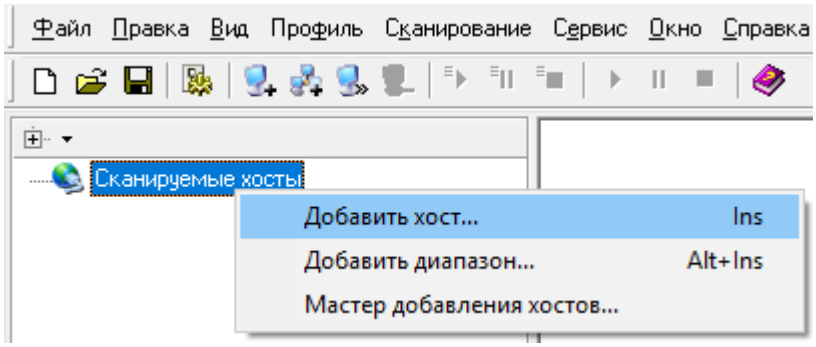

Рисунок 3.1 – Добавление нового хоста для проведения сканирования.

На сегодняшний день практически все крупные локальные сети имеют доступ к открытой сети интернет. В некоторых исключения, в некоторых локальных сетях может и не быть настроенного сервера, к которому бы был предоставлен доступ извне и для доступа к сети интернет используется тип соединения NAT, другими словами, одна рабочая станция обеспечивает интернетом все остальные подключенные по локальной сети рабочие станции. В других типах сетей могут работать несколько серверов, к которым в свою очередь был предоставлен доступ извне, в частности это делается ради удобства в использовании и настройке сервера, т.к. он владелец машины может находиться в одной стране, а сам сервер в другой. В любом случае, всегда есть одно устройство, доступ к которому предоставлен извне, и

наличие такого компьютера в сети ставит безопасность локальной сети под угрозу. На рисунке 3.2 программное обеспечение запрашивает ввод IP адрес устройства, которое необходимо просканировать.

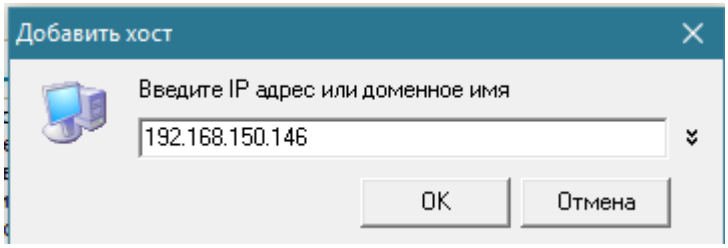

Рисунок 3.2 – Добавление нового хоста для старта сканирования

XSpider имеет возможность в автоматическом режиме сканировать порты устройства, если у устройства имеется открытый порт, то сканер уязвимостей подбирает определенный тип атаки, выводит на экран тип уязвимости, которая содержит в себе рабочая станция, и после пользователь уже на свое усмотрение решает, пользоваться ли найденной уязвимостью в системе, или же нет. Также XSpider имеет возможность абсолютно полностью в автоматическом режиме проверять выделенные диапазоны IP адресов по расписанию. Разработчиком данного программного продукта является компания Positive Technologies. Разработчики регулярно обновляют базу уязвимостей, вследствие чего данный программный продукт довольно часто получает обновления, и стремительно продолжает развиваться, хотя уже как несколько лет назад успешно закрепился на рынке, и не сдает своих позиций до сих пор. На рисунке 3.3 производится заключительный этап начала сканирования.

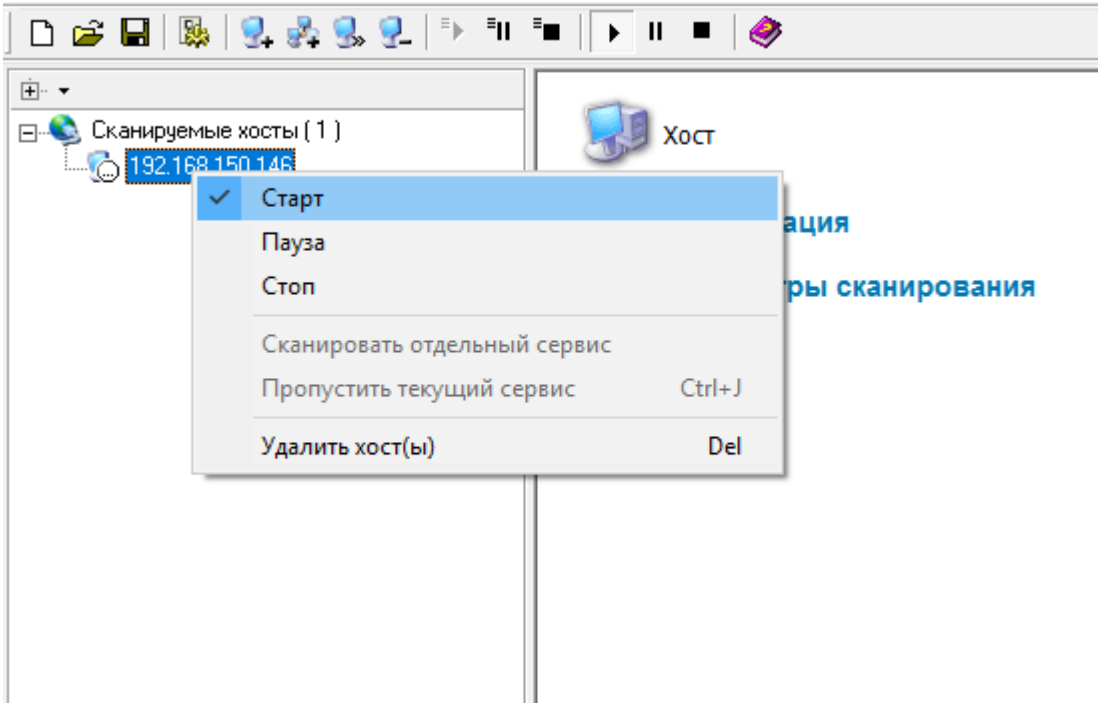

Рисунок 3.3 – Запуск сканирования

В качестве системы обнаружения/предотвращения вторжений был выбрал свободно распространяющееся программное обеспечение PSAD (Port Scan Attack Detector), в народе системы обнаружения/предотвращения вторжений называют IPS/IDS (IPS – Intrusion Prevention System, IDS – Intrusion Detection System). Выбранный программный продукт PSAD можно настроить и как систему обнаружения вторжения, и как систему предотвращения вторжений, все зависит лишь от желаний пользователя и от сугубо личных нужд. Такие системы сейчас пользуются особой популярностью, т.к. позволяют системным администраторам своевременно реагировать на инциденты информационной безопасности и предотвращать возможные проникновения в объекты сетевой инфраструктуры. Инструмент PSAD работает на основе встроенных правил и мониторинга журнала фаервола с целью обнаружить различного рода попытки вторжения. Процесс установки PSAD на операционную систему Linux Ubuntu 16.04 LTS

Программное обеспечение PSAD хранится в стандартных репозиториях Linux Ubuntu и установку необходимо провести выполнением вот этих команд:

```
Sudo apt-get update
Sudo apt-get install psad
```

В процессе установки выбираем Internet Site и доменное имя сервера, на которое будут отправлять сообщения с уведомлениями о вторжении.

Далее будет производиться настройка IPTables, система PSAD обнаруживает подозрительную активность на сервере путем мониторинга журнала событий в Firewall. Конфигурация Firewall проводится этими командами:

```
sudo iptables -A INPUT -j LOG
sudo iptables -A FORWARD -j LOG
```

Теперь необходимо настроить правило, которое бы прогоняло трафик через наш выделенный сервер, воспользуемся следующей командой:

```
sudo iptables -A INPUT -i lo -j ACCEPT
```

Теперь вносим правило, которое дает разрешение на считывание трафика выделенному приложению, используем следующую команду:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j
ACCEPT
```

Затем нужно добавить сервисы, которые должны остаться открытыми. Для того, чтобы добавить поддержку SSH, используем следующую команду:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

В нашем случае веб-сервер уже запущен, для того, чтобы разрешить трафик по 80 порту используем следующую команду:

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Если 80 порт уже занят каким-то другим сервисом, то аналогично использует такой же синтаксис, но выбираем другой порт:

```
sudo iptables -A INPUT -p protocol --dport номер_порта -j ACCEPT
```

После того, как внесли все правила, необходимо перезагрузить сервис.

Но прежде чем это сделать, необходимо применить изменения, сделанные в IPTables, используем следующую команду:

```
sudo iptables -A INPUT -j LOG
```

Также необходимо внести точно такое же правило для цепочки FORWARD:

```
sudo iptables -A FORWARD -j LOG
```

Теперь, чтобы система обнаружения вторжений отбрасывала трафик, который не подходит под установленные нами правила, необходимо применить команду ниже, это делается для того, чтобы журнал событий не забивался сторонними соединениями:

```
sudo iptables -A INPUT -j DROP
```

Существует также иная команда, которая игнорирует соединения, не подходящие под правила:

```
sudo iptables -P INPUT DROP
```

По сути разницы между этими правилами нет, результат будет одним и тем же.

Однако в случае, если вместо внесения, правила которое сбрасывает трафик, в конце цепочки была настроена политика DROP, необходимо не забыть заменить ее на ACCEPT, используем эту команду:

```
sudo iptables -P INPUT ACCEPT
sudo iptables -F
```

Если проигнорировать данную рекомендацию, то фаервол будет блокировать все входящие соединения.

По умолчанию IPTables не применяет правила, перед тем, как применить их вручную, необходимо протестировать работу под применением этих правил, для того, чтобы снизить риск поломки и возникновения конфликтов правил, необходимо использовать эти команды:

```
sudo apt-get install iptables-persistent
sudo service iptables-persistent start
```

В случае использования некоторых серверных приложений (таких как Web-сервер или FTP-сервер) атаки DoS могут заключаться в том, чтобы занять все соединения, доступные для этих приложений и держать их в занятом состоянии, не допуская обслуживания обычных пользователей. В ходе атак DoS могут использоваться обычные Интернет-протоколы, такие как

TCP и ICMP (Internet Control Message Protocol). Большинство атак DoS опирается не на программные ошибки или бреши в системе безопасности, а на общие слабости системной архитектуры. Некоторые атаки сводят к нулю производительность сети, переполняя ее нежелательными и ненужными пакетами или сообщая ложную информацию о текущем состоянии сетевых ресурсов. Этот тип атак трудно предотвратить, так как для этого требуется координация действий с провайдером. Если трафик, предназначенный для переполнения вашей сети, не остановить у провайдера, то на входе в сеть вы это сделать уже не сможете, потому что вся полоса пропускания будет занята. Когда атака этого типа проводится одновременно через множество устройств, мы говорим о распределенной атаке DoS (DDoS - distributed DoS).

Атака будет произведена на один из элементов сети, рабочую станцию с предварительно настроенной на ней PSAD-системой (система выявления потенциальных вторжений) с целью увидеть событие, проанализировать действие подключенного IP к нашей сети, произвести сбор информации о его действиях, и после проверить его IP в разработанном программном обеспечении, с целью расширить имеющуюся доказательную базу, и принять решение по возникшему инциденту. Целенаправленно по отношению к данной выделенной рабочей станции будут произведены действия, которые могут быть охарактеризованы как подозрительные, иными словами, действия, в результате которых злоумышленник имеет возможность получить подробную информацию по нашей рабочей станции, выявить ее уязвимости, в дальнейшем произвести продажу собранных данных, либо же самостоятельно воспользоваться полученными

данными, и произвести кражу информации с данной рабочей станции. Будет использован свободно распространяемый в сети интернет сканер уязвимостей XSpider. IP адрес рабочей станции 192.168.150.146. IP адрес рабочей станции, с которой я буду производить подозрительные действия: 192.168.150.1

На рисунке 3.11 система фиксации аномалий реагирует на действия с устройства с IP 192.168.150.1, и классифицирует их как потенциально опасные, на рисунке видно, что в процессе были зафиксированы возможные попытки использования различных BackDoor-ов и т.д.

Рисунок 3.11 – Процесс фиксации подозрительных действий с IP 192.168.150.1

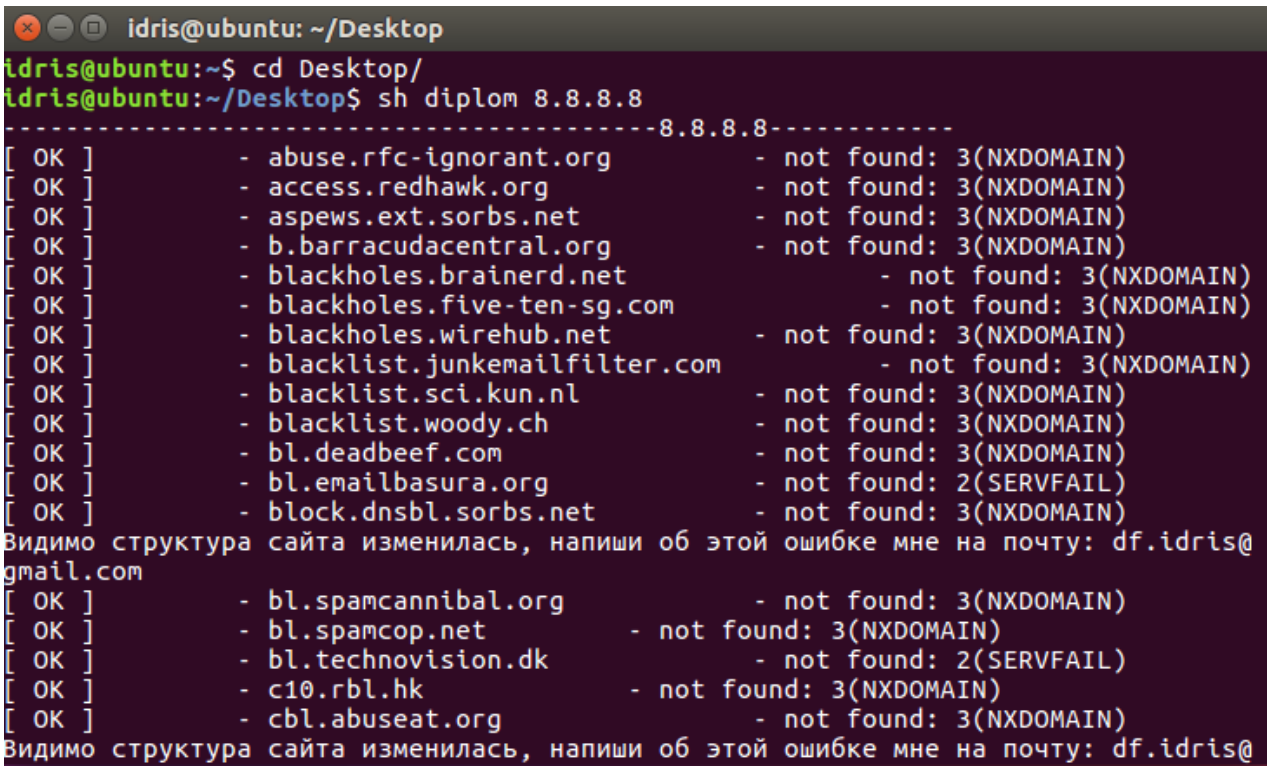

Рисунок 3.12 - Демонстрация работы разработанного программного обеспечения

На рисунке 3.12 продемонстрировано тестовая проверка IP адреса Google.com на наличие в domain name system black list (DNSBL).

Вывод

В процессе выполнения работы была реализована тестовая атака на рабочую станцию с IP 192.168.150.146 под управлением операционной системы Linux Ubuntu 16.04 LTS с предварительно настроенной на ней системой обнаружения вторжений и выявлении подозрительной активности извне, в результате тестовой атаки рабочей станции с IP 192.168.150.146 была зафиксирована попытка проникновения в операционную систему, попытка увеличения прав доступа до уровня администратора, попытка воспользоваться BackDoor-уязвимостью. Все эти атаки были пропущены фаерволом изначально, но позднее были отброшены системой обнаружения/предотвращения вторжений. Был получен IP адрес, с которого были произведены упомянутые выше типы атак: 192.168.150.1, данный IP адрес позднее был проверен при помощи разработанного программного обеспечения на наличие в DNSBL, данный IP адрес был идентифицирован открытыми черными списками как распространяющий вредоносное программное обеспечение, рассылающий спам, в результате проделанных действий сбор доказательной базы по данному инциденту был окончен, было принято заблокировать данный IP на FireWall.

4 Безопасность жизнедеятельности

Дано помещение 16 на 10, 5 метров в высоту, предположительно офисное помещение, выделенное для отдела информационной безопасности, в нем работают 10 человек, 7 мужчин и 3 женщины, есть одна кофеварка и один принтер. В данном помещении находятся также 10 ноутбуков, что не издают никаково шума, а также одно большое окно. Но в данном помещении нет кондиционера, необходимо произвести рассчет тепловых нагрузок в указанном по исходным данным помещении, внешние и внутренние.

Произвести рассчет необходимого объема воздуха для того, чтобы подать его в помещение, и по полученному значению объема воздуха подобрать подходящую модеть кондиционера. После выполнения процесса подбора подходящего кондионера привести основные характеристики для данной модели кондиционера.

1) Произвести расчет тепловых нагрузок, оказываемых на помещение: внешних и внутренних;
2) Произвести расчет объема воздуха, необходимого для подачи в помещение;
3) По полученному значению объема воздуха подобрать подходящую модель кондиционера;
4) Привести основные характеристики для данной выбранной модели кондиционера.
5) Привести схему расположения кондиционера в помещении.

Исходные данные**:**

Город: Алматы;
Параметры помещения (Д х Ш х В), м: 16 х 10 х 5;
Данные по оборудованию: кол-во 5 шт.;
Мощность Роб, кВт/ч = 0,5;
КПД $\eta = 0,75$;
Данные по ист. света: мощ. N ос. уст., Вт/м^2 = 60;
Вид ист. св.: люминиц. лампы;
Число сотрудников, из них: мужчины = 7, женщины = 3;
Окна: кол-во 4;
Площадь 1 окна, м^2 = 70;
Расположение: СЗ;
Вид: остекление в один-х метал. переплет, загрязнение умеренное;
Расчетное время суток, ч.: 12-13;
Температура в помещении, ^{0}С: летом 24, зимой 20;
Вид положения работы: тяжелая работа.

4.1 Расчёт тепловых нагрузок в помещении: внутренние и наружные

В помещении разного назначения действуют чаще всего именно тепловые нагрузки, которые возникают с наружной части помещения (наружные); а также тепловые нагрузки, которые возникают в основнос внутри помещений (внутренние).

4.1.1 Наружные тепловые нагрузки

Данные виды нагрузок представлены следующими составляющими:

– теплопоступления или теплопотери, в результате которых возникают разности температур снаружи и внутри здания, сквозь стены, несущие, опоры, дверные проемы, оконные проемы.

– Чаще всего возникающие разности температур с наружной части здания и внутренней его части летом являются положительными, в результате всего этого процесса имеет место приток тепла снаружи во внутрь помещения; и наоборот – зимой эта разность отрицательна и направление потока тепла меняется;

– теплопоступления от солнечного излучения через застекленные площади; данная нагрузка проявляется в форме ощущаемого тепла;

– теплопоступления от инфильтрации.

В зависимости от времени года и времени суток наружные тепловые нагрузки могут быть положительными. Теплопоступления и теплопотери в результате разности температур определяются по формуле 1.1:

$$Q_{огр}=V_{пом}*X_0*(t_{Нрасч}-t_{Врасч}),\ Вт \qquad (1.1)$$

где $V_{пом.}$ – объем помещения, $м^3$:

$V_{пом}= 16\cdot10\cdot5=800\ м^3$;

X_o – удельная тепловая характеристика, $Вт/м^3\ ^0C$:

$X_0=0.42\ Вт/м^3\ ^0C$;

$t_{Нрасч}$ – наружная температура (параметр А). Для холодного периода – средняя температура самого холодного месяца в 13 часов, для теплого периода – средней температуре самого жаркого месяца в 13 часов.

$t_{Врасч}$ – внутренняя температура, выбирается с учетом комфортных условий или технологических требований, предъявляемых к производственным процессам.

Для теплого времени года:

$t_{Нрасч} = 29{,}4\ ^0C$

$t_{Врасч} = 26\ ^0C$

$Q_{огр.}= 800\cdot0{,}42\cdot3{,}4=1142{,}4\ Вт$

Для холодного времени года

$t_{Нрасч} = -9\ ^0C$

$t_{Врасч} = 19\ ^0C$

$Q_{огр.}= 800\cdot0{,}42\cdot(-28) = |9408|\ Вт$

Избыток теплоты излучения солнечных лучей, в зависимости от типа стекла почти до 90% поглощается средой помещения, оставшаяся часть, естественно, отражается. Максимальная тепловая нагрузка достигается при максимальном уровне излучения, которое имеет прямую и рассеянную составляющие. Интенсивность излучения зависит от ширины местности, времени года и времени суток.

Поступление тепла от солнечного излучения через остекление определяется по формуле 1.2:

$$Q_р=(q^I F_0{}^{II} + q^{II} F_0{}^{II}) * \beta_{с.з.} \quad (1.2)$$

q^I, q^{II} – тепловые потоки от прямой и рассеянной солнечной радиации, Вт/м^2 ;
$F^I_о$, $F^{II}_о$ – площади светового проема, облучаемые и необлучаемые прямой солнечной радиацией, м^3 ;
βс. з. – коэффициент теплопропускания
βс. з. = 0.15

При отсутствии наружных затеняющих козырьков, ребер и т. д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение

$$F^I_о = F_о ; F^{II}_о = 0 \quad (1.3):$$

$$Q_р=q^I F_0 * \beta_{с.з.} = (q_{вп}+q_{вр}) * K_1{}^c * K_2 * \beta_{с.з.} * n * S_0, \text{ Вт} \quad (1.4)$$

$Q_{вп}$; $q_{вр}$ – тепловые потоки от прямой рассеянной радиации, Вт/м^2.
Lля широты в 440 СШ до полудня в 11-12 ч. при расположении З:

$Q_{вп}$ = 73 Вт/м^2 ; $q_{вр}$ = 77 Вт/м^2 ;

$F_о = n*S_о = 4 \cdot 72 = 288$ м^2 – площадь светового проема (n – число окон; $S_о$ – площадь 1 окна);

K_1 – коэффициент затемнения остекления переплетами (K^C_1 – для облученных проемов):

$K^C_1 = 0.72$;

K_2 – коэффициент загрязнения остекления:

$K_2 = 0.9$.

Тогда:

$Q_р=(73+77)*0,72*0,9*0,15*4,5=65,61$ Вт.

По таблице 5 [1] для широты в 440 СШ до полудня в 11-12 ч. при расположении В:

$Q_{вп}$ = 214 Вт/м^2; $q_{вр}$ = 79 Вт/м^2 ;

$F_о = nS_о = 4 \cdot 72 = 288$ м^2 – площадь светового проема (n – число окон; $S_о$ – площадь 1 окна);

Тогда:

$Q_р=(214+79)*0,72*0,9*0,15*4,5=128,2$ Вт.

Тогда общее теплопоступление солнечного излучения с обеих окон равно:

$Q_р$=65,61+128,2=193,81 Вт.

4.1.2 Внутренние тепловые нагрузки

Внутренние нагрузки в жилых, офисных или относящихся к сфере обслуживания помещениях слагаются в основном из тепла:

– выделяемого людьми;

– выделяемого лампами и осветительными, электробытовыми приборами;

– выделяемого компьютерами, печатающими устройствами фотокопировальными машинами пр.;

В производственных и технологических помещениях различного назначения дополнительными источниками тепловыделений могут быть: нагретое производственное оборудование, горячие материалы, в том числе жидкости и различного рода полуфабрикаты, продукты сгорания и химических реакций.

Теплопоступления от людей зависит от интенсивности выполняемой работы и параметров окружающего воздуха. Тепло, выделяемое человеком, складывается из ощутимого (явного), то есть передаваемого в воздух помещения путем конвекции и лучеиспусканий, и скрытого тепла, затрачиваемого на испарение влаги с поверхности кожи и из легких.

По таблице 8.1 летом при 24 ^{0}С один мужчина выделяет явного тепла 61 Вт, а общего – 102 Вт. Женщина выделяет 85% от нормы тепловыделений взрослого мужчины. Тогда выделение явного тепла в помещении составит:

$Q_л^я$ = 61*7+61*3*0,85=582,55 Вт.

А выделение общего тепла:

$Q_л^о$ = 102*7+102*3*0,85 = 770,1 Вт.

По таблице 8.1 зимой при 20 ^{0} С один мужчина выделяет явного тепла 82 Вт, а общего – 103 Вт. Женщина выделяет 85% от нормы тепловыделений взрослого мужчины. Тогда выделение явного тепла в помещении составит:

$Q_л^я$ = 82*7+82*3*0,85 = 783,1 Вт.

А выделение общего тепла:

$Q_л^о$=103*7+103*3*0,85 = 1333,85 Вт.

Теплопоступление от осветительных приборов, оргтехники и оборудования рассчитывается следующим образом. Теплопоступление от ламп определяется по формуле (1.4):

$$Q_{осв}= \eta * N_{осв} *F_{пол} \quad (1.4)$$

η – коэффициент перехода электрической энергии в тепловую (для люминесцентных ламп η=0.5-0.6);

$N_{осв}$ – установленная мощность ламп (N=60 Вт/м^2);

$F_{пол}$ – площадь пола:

$F_{пол}$=16·10=160

Тогда:

$Q_{осв}$=0,5·60·160= 4800 Вт

Тепло, выделяемое производственным оборудованием, определяется по формуле (1.5):

$$Q_{об} = N_{уст} * K \quad (1.5)$$

$Q_{об}$=1,8*10^3*11*0,95=18810 Вт.

Теплопритоки, возникающие за счет находящейся оргтехники, – это 30% мощности оборудования:

$Q_{орг}$ = 1,8*10^3*11*0,3=5940 Вт.

4.2 Расчёт количества воздуха, необходимого для подачи в помещение

На основании выполненных расчетов составим баланс теплопоступлений в помещении:

Лето: $Q_{изб}$=193,81+783,1+4800+18810+5940+1142,4= 31669,31 Вт

Зима**:** $Q_{изб}$=193,81+783,1+4800+18810+5940+9408=39934,31 Вт

Так как тепловой баланс для лета больше зимнего теплового баланса, то рассчитаем теплонапряженность воздуха по формуле:

$$Q_н=\frac{Q_{изб.лето} \cdot 860}{V_{пом}} \quad (1.6)$$

$$Q_н=\frac{39,934 \cdot 860}{800} = 42,938 \text{ ккал/м3}$$

При $Q_Н$>20ккал/м3 , Δt= 8 °С.

Определение количества воздуха, необходимое для поступления в помещение:

$$L=\frac{Q_{изб} \cdot 860}{C*\Delta t*\gamma} \quad (1.7)$$

$$L= \frac{39934 \cdot 860}{0,24 \cdot 8 \ * 1,206*10\text{^}4} = 355 \text{ м3/час}, \text{ где}$$

C=0,24 ккал/(кг°С) – теплоемкость воздуха,

γ=1,206 кг/$м^3$ – удельная масса приточного воздуха.

Определение кратности воздухообмена:

$$N=\frac{355}{800} = 0,44 \text{ час}^{-1}$$

4.3 Основные характеристики и схема расположения выбранного кондиционера

Исходя из полученных данных, выберем кондиционер сплит-системы настенного типа.

Таблица 4.1 – Основные технические характеристики настенного кондиционера серии TOSHIBA RAV-SM1102CT-E

Эл. питаниеВ /Гн	Произв. по холоду, Вт	Потр. эл мощн, Вт	Потребл ток, А	Произв. по теплу, кВт	Размер (внешн. блок) мм	Расход воздуха, м3 /ч	Размер (внутр. блок) мм
220/50/1	14,07	7300	10,2	7330	L 845 H 700 B 320	8000	L 735 H 620 B 310

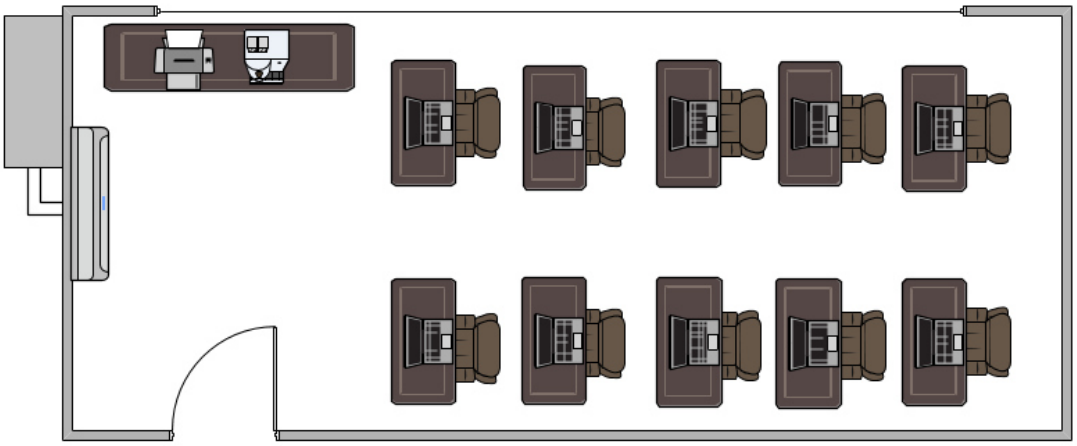

Рисунок 4.1 – Схема расположения кондиционеров в производственном помещении

5 Технико-экономическое обоснование

Данный дипломный проект подразумевает проектирование системы защиты информации с применением кодирования ключа. Сложность и актуальность мой работы заключается в том, что программа содержит в себе сразу 3 алгоритма шифрования. Один для генерации секретного ключа, второй для кодирования, и третий, с помощью ключа шифрует передаваемое сообщение. Этот продукт может применяться в условиях малого и среднего бизнеса, а также в частном порядке.

В данной части дипломной работы будут рассчитаны все расходы на разработку, требуемых материалов и устройств, затраты на программные обеспечения, затраты на электроэнергию и рабочий труд, а также амортизацию основных компонентов.

5.1 Расчет трудоемкости разработки программного продукта

Приведем перечень основных этапов и работ, которые нужно выполнить для определения трудоемкости разработки программного обеспечения. Трудоемкость работы определялась согласно нормам времени на проведение расчетов, анализа и исследований. Форма разделения работ по этапам с указанием трудоемкости их выполнения приведена в таблице 5.1.

Таблица 5.1 – Распределение работ по этапам и оценка их трудоемкости

Этапы разработки ПП	Виды работ	Трудоемкость разработки, чел. х ч.
1 этап	Постановка задач	18
2 этап	Разработка и утверждение технического задания на разработку ПП	23
3 этап	Знакомство с существующим программным для сбора данных	14
4 этап	Поиск и ознакомление с литературой по теме современных алгоритмов сбора и их применения	28
5 этап	Составления аналитического обзора существующих приложений сбора данных	26
6 этап	Реализация проекта	33
7 этап	Отладка программного обеспечения	26
8 этап	Оформление отчета и составление выводов о проделанной работе	28
9 этап	Тестирование проекта	23
ИТОГО		219

Продолжительность рабочего дня равна 8 часам. То есть количество дней, затраченных на осуществление – 27 рабочих дней.

Расчет затрат на разработку программного продукта

Определение затрат на разработку программного продукта производится на основе существующей сметы, которая включает следующие статьи:

– материальные затраты;
– затраты на оплату труда;
– социальный налог;
– амортизация основных фондов;

Статья «Материальные затраты» состоит из основных и вспомогательных материалов, электрической энергии, которые необходимы для разработки программного продукта. Расчет затрат на материальные ресурсы производится по форме, приведенной в таблице 5.2

Таблица 5.2 – Затраты на материальные ресурсы

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Офисная бумага IQ Ultra, A4	SvetoCopy	Пачка	2	1250	2 500,00
Тетрадь А4 (96 листов)	Maui and Sons	Штук	1	510	510,00
Блокнот (96 листов)	OFFICE	Штук	1	944	944,00
Ручки	Universal Liner	Штук	5	120	600,00
Карандаши	Maped	Штук	5	65	325,00
Компьютерная мышь (беспроводная)	Logitech M335	Штук	1	10990	10 990,00
Итого					15 869,00

В ноутбуке Acer E5-576G предусмотрены встроенная операционная система и дополнительное программное обеспечение, поэтому затраты на покупку новой операционной системы Windows 10 и лицензионную MS Office производиться не будут.

Таблица 5.3 – Затраты на ОС и ПО, необходимые для проекта

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Ноутбук	Acer E5-576G	Шт.	1	140 000	140 000,00
Принтер	Epson L322	Шт.	1	56420	56 420,00
Модем	TP-Link TD-W9970	Шт.	1	7 590	7 590,00
Итого					204 010,00

Общая сумма затрат на материальные ресурсы ($З_м$) определяется по формуле:

$$З_м = \sum P_i \times Ц_i \quad (5.1)$$

где P_i– расход i-го вида материального ресурса, натуральные единицы;
$Ц_i$– цена за единицу i-го вида материального ресурса, тг;
i – вид материального ресурса;
n – количество видов материальных ресурсов.

$З_м = 15\ 869 + 204\ 010 = 219\ 879{,}00$ (тг)

Материальные затраты составят 219 879,00 тенге.

5.3 Расчет затрат на электроэнергию

Важно рассчитать затраты на электроэнергию, потому что в процессе работы используется электрооборудование. Время работы оборудования для разработки программного продукта берется равным 219 часов для ноутбуков и модема, данное количество часов было рассчитано в таблице 6.1.

$$Э = З_{эл.эн.обор} + З_{доп.нуж} \quad (5.2)$$

где $З_{эл.эн.обор}$ – затраты на электроэнергию оборудования;
$З_{доп.нуж.}$ – затраты электроэнергии на дополнительные нужды.
Расходы электроэнергии на оборудование рассчитывается по формуле:

$$З_{эл.эн.обор} = \sum W \times К_{исп} \times S \times T \quad (5.3)$$

гдеW – потребляемая мощность, Вт;
$К_{исп}$ – коэффициент использования ($К_{исп} = 0{,}7..0{,}9$);
T – время работы;
S – тариф (1кВт/ч = 18,32тг).

Сводные результаты расчета затрат на электроэнергию представлены в таблице 5.4.

Таблица 5.4 – Затраты на электроэнергию

Наименование приборов	Паспортная мощность, кВт	Коэффициент мощности	Время работы оборудования, ч	Цена ЭЭ тг/ кВтч	Сумма, тг
Ноутбук	0,6	0,7	219	18,32	1685,07
Модем	0,08	0,9	100	18,32	131,90
Принтер	0,5	0,9	12	18,32	98,92
Кондиционер	0,8	0,9	200	18,32	2638,08
Освещение	0,3	0,7	219	18,32	842,53
Итого					5 396,50

$З_{эл.эн.обор} = 1685{,}07 + 131{,}90 + 98{,}92 + 2638{,}08 + 842{,}53 = 5396{,}50$(тенге)

Затраты на дополнительные потребности берутся по укрупненному показателю в размере 5% от затрат на оборудование:

$$З_{доп.нуж} = 5\% \times З_{эл.эн.обор} \quad (5.4)$$

Затраты на дополнительные потребности рассчитаны по формуле (5.4):

$З_{доп.нуж} = 0,05 \times 5\ 396,50 = 269,82$ (тенге)

Таким образом суммарные затраты на электроэнергию составляют:

Э = 5396,50 + 269,82 = 5 666,32 (тенге)

5.4 Расчет затрат на оплату труда

Над разработкой проекта работают два сотрудника:

- технический руководитель проекта – он изучает предметную область, проводит анализ требований к системе, занимается внедрением и поддержкой;
- разработчик – создание и реализует модель, занимается тестировкой и отладкой продукта;

Общая сумма затрат на оплату труда ($З_{тр}$) определяется по формуле:

$$З_{тр}=\sum ЧС_i \times Т_i \quad (5.5)$$

где $ЧС_i$– часовая ставка i-го работника, тг;

Ti – трудоемкость разработки модели, чел.×ч;

i – категория работника;

n – количество работников, занятых разработкой ПП.

На этапах разработки, участники разработки задействованы неравноценно, для этого необходимо рассчитать часовую ставку работника, а затем общий размер заработной платы.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i=\frac{ЗПi}{ФРВi} \quad (5.6)$$

где Зпi – месячная заработная плата i-го работника, тг;

ФРВi- месячный фонд рабочего времени i-го работника

Месячная заработная плата сотрудников:

технический руководитель проекта – 180 000 тг;

Разработчик – 120 000 тг.

$ЧС_i$= 180 000 / (22 × 8) = 1 022,72 тг/ч

$ЧС_i$= 120 000 / (22 × 8) = 681,81 тг/ч

Часовая ставка технического руководителя проекта составляет 1 022,72 (тг/ч), трудоемкость разработки – 90 ч. Часовая ставка разработчика составляет 681,81 (тг/ч), трудоемкость разработки – 219 ч.

Рассчитаем общую сумму затрат на оплату труда по формуле (5.5):

$З_{тр}$ = 1 022,72 × 90 + 681,81 × 219 = 241 361,19 (тенге)

Сводные результаты расчета затрат на оплату труда показаны в таблице 5.5.

Таблица 5.5 – Расчёт основной заработной платы разработчиков.

Категория работника	Квалификация	Трудоемкость разработки ПП, час.	Часовая ставка, тг/ч	Сумма, тг.
Технический руководитель	Профессор	90	1 022,72	92 044,8
Разработчик	Студент	219	681,81	149 316,39
Итого				241 361,19

5.5 Расчет затрат по социальному налогу

Социальный налог – согласно Налоговому кодексу Республики Казахстан составляет 9,5 % от ФОТ (фонда оплаты труда). Следует отметить, что пенсионные отчисления не облагаются социальным налогом.

$$С_н = (ФОТ- ПО)\times 0,095 \quad (5.7)$$

где ПО - отчисления в пенсионный фонд, 10% от ФОТ.

Социальный налог рассчитываем по формуле (5.7):

ПО = 241 361,19 × 0,1 = 24 136,2 тенге;
$С_н$ = (241 361,19 – 24 136,2) × 0,095 = 20 636,37 тенге

Сводные результаты расчета затрат представлены в таблице 5.6.

Таблица 5.6 - Начисление социального налога

Категория работника	Количество человек	Заработная плата, тг	Пенсионные отчисления, тг	Социальный налог, тг
Научный руководитель	1	92 044,8	9 204,48	7 869,75
Разработчик	1	149 316,39	15 272,5	12 766,55
Итого				20 636,3

5.6 Амортизация основных фондов и прочие затраты

Годовые нормы амортизации ОФ принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования ОФ. Амортизация основных фондов определяется:

$$А_г = \frac{С_{об} \times Н_а}{100} \quad (5.8)$$

где, $С_{об}$ – стоимость оборудования;

$Н_а$ – норма амортизации (норма амортизация = 20);

По формуле 5.8 рассчитаем сумму амортизационных отчислений за год для ноутбука:

$$А_г = \frac{140\,000 \times 20}{100} = 28\,000{,}00 \text{ тг}$$

Рассчитаем сумму амортизации за время разработки:

$$А_р = \frac{28\,000 \times 28}{365} = 2\,147{,}95 \text{ тг}$$

Аналогичным способом рассчитаем сумму амортизации для остального оборудования.

Результаты расчетов приведены в таблице 5.7

Таблица 5.7 - Амортизация основных фондов

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Сумма амортизации за год, тг	Сумма амортизации за время разработки, тг
Ноутбук	140 000	20	28 000	2 147,95
Принтер	56 420	20	11 284	865, 62
Модем	7 590	15	1 138,5	38,99
ИТОГО амортизация основных средств			40 422,5	3 052,56

Смета затрат на разработку программного продукта

На основании полученных данных по отдельным статьям составляется смета затрат на разработку программного продукта по форме, приведенной в таблице.

Таблица 5.8 – Смета затрат на разработку программного продукта

Статьи затрат	Сумма, тг
Затраты на оборудование и материальные расходы	219 879,00
Затраты на оплату труда	241 361,19
Социальные налоги	20 636,30
Затраты на электроэнергию	5 666,32

Амортизация основных фондов	3 052,56
Прочие расходы (интернет)	22 404,00
Итого по смете	512 999,37

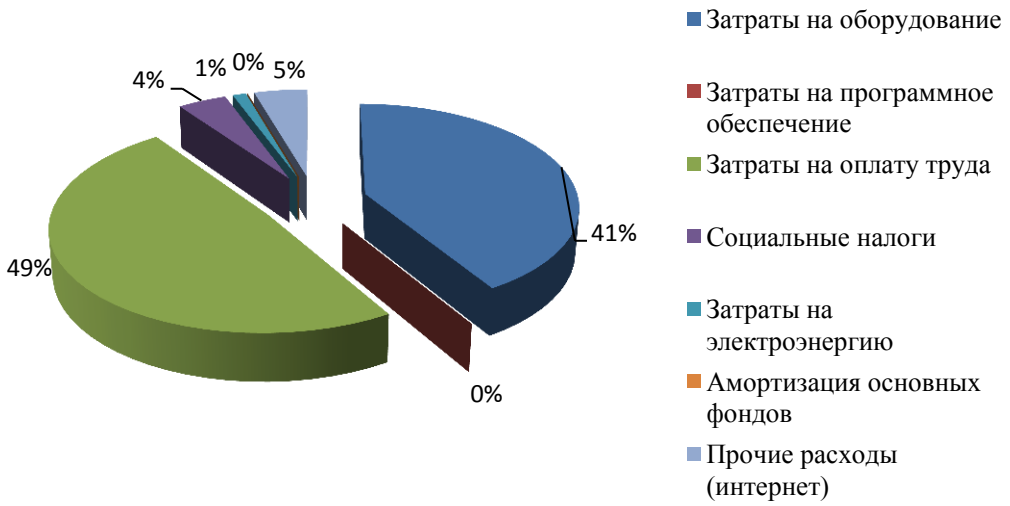

Рисунок 5.1 - Диаграмма структуры эксплуатационных затрат

5.7 Определение возможной (договорной) цены программного продукта

Величина возможной (договорной) цены программного продукта устанавливается на основе эффективности, качества и сроков её выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена ЦД для прикладных программных продуктов рассчитывается по формуле:

$$Ц_Д = З_{НИР}\left(1 + \frac{Р}{100}\right) \quad (5.9)$$

где $З_{НИР}$- затраты на разработку ПП, тг;
Р – средний уровень рентабельности ПП. (20%).

$Ц_Д$ = 512 999,37 × (1 + 20/100) = 615 599,24 тенге

Р = 512 999,37 × 0,2 = 102 599,87 тенге

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка (НДС) устанавливается законодательно. Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_Д + Ц_Д \times НДС \quad (5.10)$$

512 999,37 + 512 999,37 × 0,12 = 689 471,14 тенге

Рассчитанную возможную цену ПП можно округлить до 669 500 тенге.

Вывод

В данной главе были произведены расчеты экономических затрат на приобретение необходимого оборудования и программного обеспечения для разработки программы сбора доказательной базы, а также расчет на оплату труда. Полностью рассчитаны затрат на покупку оборудования; расчет трудоемкости разработки программного продукта; социальный налог и пенсионные отчисления, расходы на электроэнергию и амортизационные отчисления.

Затраты на разработку ПО составили 512 999,37 тг. Договорная цена составила = 689 471,14 тг. Итоговая прибыль составила 102 599,87 тг.

Список литературы

1 Устинов, Г.Н. Уязвимость и информационная безопасность телекоммуникационных технологий/ Г.Н. Устинов М.: Радио и связь, 2016. - 342 с.

2 И. Ф. Мазалов, К. Г. Мустафин, Е. М. Тыщенко, М. А. Сералиева Методические указания по выполнению РГР для студентов специальности 5В0731100-БЖ. – Алматы: АУЭС, 2015. – 38 с.

3 Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 – Информационные системы – Алматы: АУЭС, 2013.

4 Аманжолова К. Б., Алибаева С. А. Экономика предприятии телекоммуникации: Учебное пособие. – Алматы: АИЭС, 2003.

5 Информационная безопасность и защиты информации. Учебное пособие. Ростов-на-Дону, 2015.

6 Официальный сайт VMWare, URL: https://www.vmware.com (дата обращения 15.03.2019).

7 Официальный сайт *Ubuntu,* URL: https://www.ubuntu.com (дата обращения 15.03.2019).

8 Инструкция по настройке PSAD, статьи по настройке качественной системы информационной безопасности, URL: https://www.8host.com (дата обращения 15.03.2019)

9 Официальный источник, разработчик и дистрибьютор XSpider, URL: https://www.ptsecurity.com/ru-ru/products/xspider/ (дата обращения: 15.03.2019)

10 Боршевников А. Е. Сетевые атаки. Виды. Способы борьбы // Современные тенденции технических наук: материалы Междунар. науч. конф. (г. Уфа, октябрь 2014 г.). – Уфа: Лето, 2014. – С. 8-13. – URL https://moluch.ru/conf/tech/archive/5/1115/ (дата обращения 29.05.2019).

Приложение А

```
#!/bin/sh
#
#
SCRIPT_NAME='Дипломная работа'
VERSION='0.3.5'
DESCRIPTION='Дипломная работа, проверка IP в базах на вредонос.
Выполнил Дифу Идрис.'
AUTHOR='Difu Idris AUPET'
EMAIL='df.idris@gmail.com'
VK='vk.com/waless'
INSTAGRAM='http://instagram.com/df.idris'
LICENSE='Ubuntu.com'
SED='/bin/sed'
WGET='/usr/bin/wget'
HOST='/usr/bin/host'
ICONV='/usr/bin/iconv -t UTF-8 '
PING='/bin/ping -c 1 '
DNSBL_LIST='abuse.rfc-ignorant.org access.redhawk.org
aspews.ext.sorbs.net b.barracudacentral.org
blackholes.brainerd.net blackholes.five-ten-sg.com
blackholes.wirehub.net
blacklist.junkemailfilter.com
blacklist.sci.kun.nl blacklist.woody.ch bl.deadbeef.com
bl.emailbasura.org block.dnsbl.sorbs.net
bl.redhatgate.com bl.spamcannibal.org
bl.spamcop.net bl.technovision.dk c10.rbl.hk cbl.abuseat.org
cbl.anti-spam.org.cn cblless.anti-spam.org.cn
cblplus.anti-spam.org.cn
combined.njabl.org db.wpbl.info
dialups.mail-abuse.org dialups.visi.com
dnsbl-0.uceprotect.net dnsbl-1.uceprotect.net dnsbl-2.uceprotect.net
dnsbl-3.uceprotect.net dnsbl.ahbl.org
dnsbl.cyberlogic.net
dnsbl.jammconsulting.com dnsbl.kempt.net
dnsbl.njabl.org dnsbl.sorbs.net duinv.aupads.org
dul.dnsbl.sorbs.net
dul.ru fl.chickenboner.biz hil.habeas.com
hostkarma.junkemailfilter.com
http.dnsbl.sorbs.net http.opm.blitzed.org
images.rbl.msrbl.net ips.backscatterer.org
ircbl.ahbl.org ix.dnsbl.manitu.net
korea.services.net l2.bbfh.ext.sorbs.net list.dnswl.org
mail-abuse.blacklist.jippg.org map.spam-rbl.com
misc.dnsbl.sorbs.net msgid.bl.gweep.ca
 multi.surbl.org multi.uribl.com
no-more-funn.moensted.dk ohps.dnsbl.net.au omrs.dnsbl.net.au
orid.dnsbl.net.au orvedb.aupads.org osps.dnsbl.net.au
```

```
osrs.dnsbl.net.au owfs.dnsbl.net.au owps.dnsbl.net.au
pbl.spamhaus.org
phishing.rbl.msrbl.net probes.dnsbl.net.au
proxy.bl.gweep.ca psbl.surriel.com query.bondedsender.org
rbl-plus.mail-abuse.org rbl.snark.net rdts.dnsbl.net.au
relays.bl.gweep.ca
relays.bl.kundenserver.de relays.mail-abuse.org
relays.nether.net ricn.dnsbl.net.au rmst.dnsbl.net.au
rot.blackhole.cantv.net rsbl.aupads.org satos.rbl.cluecentral.net
sbl.csma.biz
sbl.spamhaus.org sbl-xbl.spamhaus.org
smtp.dnsbl.sorbs.net socks.dnsbl.sorbs.net socks.opm.blitzed.org
sorbs.dnsbl.net.au spam.dnsbl.sorbs.net spamguard.leadmon.net
spam.olsentech.net spamrbl.imp.ch
spamsites.dnsbl.net.au spamsources.dnsbl.info
spamsources.fabel.dk spam.wytnij.to t1.bl.dnsbl.net.au t1.dnsbl.net.au
ubl.unsubscore.com
ucepn.dnsbl.net.au virbl.bit.nl
virbl.dnsbl.bit.nl virus.rbl.jp virus.rbl.msrbl.net web.dnsbl.sorbs.net
whois.rfc-ignorant.org wingate.opm.blitzed.org wormrbl.imp.ch
wpbl.dnsbl.net.au xbl.spamhaus.org
zen.spamhaus.org zombie.dnsbl.sorbs.net'
# Параметры по умолчанию
IS_QUIET=0
# -q  - Ничего не выводить
IS_DNSBL_PRINT=0
# -l  - Вывести список dnsbl-серверов
IS_ONE_DNSBL=0
# -d  - Указатель, что надо проверить по указанному DNSBL
IS_FILE=0
# -f  - Указатель, что используется файл
IS_INET=0
# -i  - Сграбить список DNSBL с сайта http://myiptest.com/
IP_LIST=''
#     - Тут будут все IP'шники
IS_TIME_OUT_ERROR=0
#SS -t     - Считать тайм ауты подключения к DNSBL ошибками и выводить
в поток ошибок?
DNS_SERVER='208.67.222.222'
# -s  - DNS Server (OpenDNS.com), через который будет проверяться IP по
DNSBL
RESOLVCONF='/etc/resolv.conf'
#  name    Конвертирует строку в текущую локаль из UTF-8 и выводит
строку на стандартный поток либо поток ошибок
#  $1      Строка, котору надо вывести (надо заключать в кавычки, если
есть пробельные символы)
#  $2      Куда выводить (stduot (1)  или stderr (2))
#  output  Возвращает указанную строку ($1) выводя ее на stdout или
stderr
#  return  0
```

```
_output()
     {
     # Если обычный режим, то выводим инфу на консоль, иначе просто все
игнорируем
     if [ $IS_QUIET -eq 0 ] ; then
          if [ $1 -eq 2 ] ; then
               echo "$2" | $ICONV 1>&2
          else
               echo "$2" | $ICONV
          fi
     fi
}
#  name    Версия, копирайт, лицензия, контакты
#  output  --//--
#  return  0
_version() {
     _output 1 "$SCRIPT_NAME, version $VERSION"
     _output 1 "Copyright $AUTHOR"
     _output 1 "License $LICENSE"
     _output 1 "Contacts: $EMAIL | $VK | $INSTAGRAM"
}
#  name    "Правила" использования скрипта
#  output  --//--
#  return  0
_usage() {
     _output 1 "$SCRIPT_NAME [-q] [-l]
     [-d dnsbl | -f filename | -i] [-s dns_server]
     [-t] [-w] [--] [ip or domain-name ...]"
}
#  name    Справка по использованию параметров скрипта
#  output  --//--
#  return  0
_help() {
     _usage
     _output 1 ""
     _output 1 "   -q - Ничего не выводить"
     _output 1 "   -l      - Показать список DNSBL-серверов"
     _output 1 "   -d - Проверить по указанному DNSBL"
     _output 1 "   -f      - Взять список DNSBL-серверов из файла (по
одному на строчку)"
     _output 1 "   -i      - Сграбить список DNSBL с сайта
http://myiptest.com/"
     _output 1 "   -s      - Указать DNS-сервер, через который будет
идти проверка"
     _output 1 "   -t      - Считать Time Out'ы ошибкой"
     _output 1 "   --      - Считается, что дальше идут только ip и/или
доманные имена"
     _output 1 "   [ip or domain-name ...] - ip и/или доменный имена,
которые нужно проверить"
     _output 1 "   --version    - Версия ПО"
```

```
}
#  name          Сообщение о неизвестном/неверном параметре переданном
скрипту
#  output  --//--
#  return  0
_bad_param() {
     _output 2 "Неизвестный параметр: $1"
}
# name
Проверяет, является ли переданный параметр ip-адресом и, если да,
заносит его в переменную $IP_LIST
# $1       ip-адрес
# output   none
# return   0
_check_and_build_ip_list() {
     CBL=$(echo "$1" | $SED -n '/[^0-9\.]\+/p')
     #если пусто, значит считаем, что у нас "IP"
     if [ -z "$CBL" ] ; then
          IP_LIST="$IP_LIST $1"
     #иначе "доменное имя"
     else
          IP_LIST="$IP_LIST $($HOST $1 | $SED -n '/has\
address/s/^.\+\ \([0-9\.]\+\).*$/\1/gp')"
     Fi }
# Парсим параметры запуска
if [ $# -eq 0 ] ; then _help ; exit 1 ; fi
while [ $# -gt 0 ]; do
     case "$1" in
          "-q" ) IS_QUIET=1; shift ;;
          "-t" ) IS_TIME_OUT_ERROR=1; shift ;;
          "-f" ) IS_FILE=1; if [ -r "$2" ] ; then DNSBL_LIST=$($SED
'/^$/d' $2) ; shift 2 ; else _output 2 "Файл $2 либо не существует,
либо нет прав на чтение!" ; exit 1 ; fi ;;
          "-i" ) IS_INET=1 ; DNSBL_LIST=$($WGET -q -O -
'http://www.myiptest.com/staticpages/index.php/check-Blacklisted-IP-
DNSBL/'$DNS_SERVER | $SED -n '/href=.\+dnsbl=/s/^.\+dnsbl=\([a-zA-Z0-
9\.\-]\+\).*$/\1/gp');
          shift ;;
          "-d" ) IS_ONE_DNSBL=1 ; DNSBL_LIST=$2 ; shift 2 ;;
          "-l" ) IS_DNSBL_PRINT=1 ; shift ;;
          "-s" ) DNS_SERVER=$2 ; shift 2 ;;
          "--" ) shift ; while [ $# -gt 0 ] ; do
_check_and_build_ip_list $1 ; shift ; done ; break ;;
          "--version"     ) _version; exit 255 ;;
          "--help"   ) _help; exit 255 ;;
          "--usage"  ) _usage; exit 255 ;;
          -*         ) _bad_param $1; exit 1 ;;
          *          ) _check_and_build_ip_list $1; shift ;;
     esac
done
```

```
# Список DNSBL-серверов заполнен?
if [ -z "$DNSBL_LIST" -a $IS_INET -eq 1 ] ;
then _output 2 'Список DNSBL-серверов пуст! Видать не удалось
подкючиться к северу.'; fi
if [ -z "$DNSBL_LIST" -a $IS_FILE -eq 1 ] ;
then _output 2 'Список DNSBL-серверов пуст! Видать файл пустой.'; fi
if [ -z "$DNSBL_LIST" ] ;
then _output 2 'Список DNSBL-серверов пуст!'; fi
#  name    Выводит сообщение "Не совместимые параметры!"
#  output  --//--
#  return  0
_bad_params_combine() {
     _output 2 "Несовместимые параметры!"
     exit 1
}
# Проверка на (не)совместимость параметров
if [ $IS_DNSBL_PRINT -gt 0 -a $IS_QUIET -gt 0 ] ;
then _bad_params_combine ; fi
if [ $IS_FILE -gt 0 -a $IS_INET -gt 0 ] ;
then _bad_params_combine ; fi
if [ $IS_FILE -gt 0 -a $IS_ONE_DNSBL -gt 0 ] ;
then _bad_params_combine ; fi
if [ $IS_ONE_DNSBL -gt 0 -a $IS_INET -gt 0 ] ;
then _bad_params_combine ; fi
# Вывод списка DNSBL серверов, если на то был "запрос"
if [ $IS_DNSBL_PRINT -gt 0 ] ;
then echo "$DNSBL_LIST" | $SED 's/\ /\n/g' | $SED '/^$/d' ; exit 0 ; fi
#  name    "Переворачивает" переданный ip-адрес и возвращает его
#  $1      ip-адрес
#  output  "Перевернутый" ip-адрес
#  return  0
_reverse_ip() {
     echo $1 | $SED -e 's/$/\n/;:z;s/\([0-
9]\{1,3\}\)\([^\n]*\n\)/\2\1./;tz;s/.*\n//;s/.$//'
}
#  name    Получает ответ от DNSBL сервера, парсит его и выводим
соответствующую информацию
#  $1      "Перевернутый" ip-адрес для проверки
#  output  Сообщение о результате проверки
#  return  0 or 1
_get_and_parse_dnsbl_return() {
     for DNSBL in $DNSBL_LIST ; do
          GPD=$($HOST -t A $1.$DNSBL $DNS_SERVER)
          RES_A=$(echo "$GPD" | $SED -n '/not\ found/p')
          # Нас нет в DNSBL :)
          if [ -n "$RES_A" ] ; then
               _output 1 "[ OK ]
- $DNSBL         - $(echo $RES_A | $SED 's/^.*\(not\ found.\+$\)/\1/')"
          else
```

```
                RES_A=$(echo "$GPD" | $SED -n '/\ has\ address\
/s/^.*address\ \([0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-
9]\{1,3\}\).*$/\1/p')
                # У нас проблемы :(
                if [ -n "$RES_A" ] ; then
                     # Узнаем подробности
                     RES_TXT=$($HOST -t TXT $1.$DNSBL $DNS_SERVER |
$SED -n '/text/s/^.*\"\(.\+\)\".*$/\1/p')
                     _output 2 "[ INFECTED!!!! ] - $DNSBL       -
($RES_A) $RES_TXT"
                     RETURN_CODE=1 else
                     # Или может мы просто не можем достучаться до
DNSBL-сервера?
                     RES_A=$(echo "$GPD" | $SED -n '/connection timed
out/p')
                     if [ -n "$RES_A" ] ; then
                          if [ $IS_TIME_OUT_ERROR -gt 0 ] ; then
                               _output 2 "[ ERROR ]      - $DNSBL
     - connection time out"
                               RETURN_CODE=1
                          else
                               _output 1 "[ ERROR ]      - $DNSBL
     - connection time out"
                          fi
                     else
                          _output 2 'Видимо структура сайта
изменилась, напиши об этой ошибке мне на почту: df.idris@gmail.com'
                          RETURN_CODE=1
                     fi
                fi
           fi
           done
           return $RETURN_CODE }
# IP указан?
if [ -z "$IP_LIST" ] ;
then _output 2 "Не указаны адреса для проверки!" ; exit 1 ; fi
# Если есть в системе resolv.conf...
if [ -r $RESOLVCONF ] ; then
     # ...выдираем из него IP'шники DNS-серверов...
     NS=$($SED -n '/nameserver/s/nameserver[\ \t]\+\([0-9\.]\+\)/\1/p'
$RESOLVCONF)
     # ...и проверяем их на доступность...
     for NS_CURRENT in $NS ; do
          # ...если хоть один доступен...
          $PING $NS_CURRENT >/dev/null 2>&1
          if [ $? -eq 0 ] ; then
               # ...то будем юзать его
               DNS_SERVER=$NS_CURRENT
               break
          fi
```

```
     done
fi   # В противном случае будет использоваться DNS-сервер по дефолту
(см. переменную $DNS_SERVER в разделе "Параметры по умолчанию")
for IP in $IP_LIST ; do
     # Получаем "перевернутый" IP
     R_IP=$(_reverse_ip $IP)
     _output 1 "----------------------------------------$IP----------
--"
                # Проверяем наши IP по спам-базам и выводим результат
на консоль
                _get_and_parse_dnsbl_return $R_IP
                EXIT_CODE=$?
Done
exit $EXIT_CODE
# простая SHELL функция, которая выдает нам сообщение об ошибке
#  $0  : Имя скрипта
# >&2  : redirect/send the message to stderr
ERROR() {
  echo $0 ERROR: $1 >&2
  exit 2
}
# -- Проверка параметров
[ $# -ne 1 ] && ERROR 'Пожалуйста, укажите один IP адрес
# -- Проверка на правильность написания IP адреса'.'
# -- Процесс реверсинга IP адреса
# -- Если IP введен некорректно, реверсинг пройдет с ошибкой'
reverse=$(echo $1 |
  sed -ne "s~^\([0-9]\{1,3\}\)\.\([0-9]\{1,3\}\)\.\([0-
9]\{1,3\}\)\.\([0-9]\{1,3\}\)$~\4.\3.\2.\1~p")
if [ "x${reverse}" = "x" ] ; then
      ERROR  "IMHO '$1' Это неправильный IP адрес"
      exit 1
fi
# Применяем IP адресу необходимые параметры как аргумент
# Если IP адрес не прошел проверку выводится сообщение,
# Если реверсинг прошел успешно, то IP будет выглядеть так 44.33.22.11
# Процесс реверсинга:
#   [ "x44.33.22.11" = "x" ]
# Тут программа выдает сообщение об ошибке
# Пустой реверсинг выводит сообщение об ошибке в IP адресе
# Текст будет выглядеть примерно так:
#   [ "x" = "x" ]
# При выводе сообщения об ошибке программа автоматически прекратит
работу
# -- реверсинг совместно с DNS ( address -> name) DNS lookup
REVERSE_DNS=$(dig +short -x $1)
echo IP $1 NAME ${REVERSE_DNS:----}
# -- cycle through all the blacklists
for BL in ${BLISTS} ; do
    # Вывод на экран сообщения о том, что IP был найдет в DNSBL
```

```
    printf "%-60s" " ${reverse}.${BL}."
    # вывод ссылки на экран
    #echo "$(dig +short -t a ${reverse}.${BL}. |  tr '\n' ' ')"
    LISTED="$(dig +short -t a ${reverse}.${BL}.)"
    echo ${LISTED:----}
done
```