МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__4__» _04_____ 2022 г.

АНАЛИЗ ЗАЩИЩЕННОСТИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Методические указания для выполнения лабораторных и практических работ
студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00,
12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Анализ защищенности вычислительных систем: методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 16 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ 1290.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Целью практической является помочь студенту провести сканирование системы для выявления уязвимостей.

ЗАДАНИЕ

Необходимо:

1. Выполнить сканирование сети
2. Научиться контролировать сеть
3. Совершить поиск и обнаружить уязвимые порты
4. Идентифицировать уязвимости вычислительной системы обеспечения безопасности
5. Скорректировать систему обеспечения безопасности

СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист
2. Краткая теория
3. Описание выполнения практической работы со скриншотами
4. Полный отчет о найденных уязвимостях и описание действий, предпринятых по их ликвидации
5. Вывод

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Защищенность является одним из важнейших показателей эффективности

функционирования ВС, наряду с такими показателями как надежность, отказоустойчивость, производительность и т. п.

Под защищенностью ВС будем понимать степень адекватности реализованных в ней механизмов защиты информации существующим в данной среде функционирования рискам, связанным с осуществлением угроз безопасности информации. Под угрозами безопасности информации традиционно понимается возможность нарушения таких свойств информации, как конфиденциальность, целостность и доступность.

На практике всегда существует большое количество не поддающихся точной оценке возможных путей осуществления угроз безопасности в отношении ресурсов ВС. В идеале каждый путь осуществления угрозы должен быть перекрыт соответствующим механизмом защиты. Данное условие является первым фактором, определяющим защищенность ВС. Вторым фактором является прочность существующих механизмов защиты, характеризующаяся степенью сопротивляемости этих механизмов попыткам их обхода, либо преодоления. Третьим фактором является величина ущерба, наносимого владельцу ВС в случае успешного осуществления угроз безопасности.

На практике получение точных значений приведенных характеристик затруднено, т. к. понятия угрозы, ущерба и сопротивляемости механизма защиты трудноформализуемы. Например, оценку ущерба в результате НСД к информации политического и военного характера точно определить вообще невозможно, а определение вероятности осуществления угрозы не может базироваться на статистическом анализе. Оценка степени сопротивляемости механизмов защиты всегда является субъективной.

Описанный в настоящей работе подход позволяет получать качественные оценки уровня защищенности ВС путем сопоставления свойств и параметров ВС с многократно опробованными на практике и стандартизированными критериями оценки защищенности.

Для того, чтобы математически точно определить этот показатель, рассмотрим формальную модель системы защиты ВС.

Основой формального описания систем защиты традиционно считается модель системы защиты с полным перекрытием, в которой рассматривается

взаимодействие "области угроз", "защищаемой области" (ресурсов ВС) и "системы защиты" (механизмов безопасности ВС).

К настоящему времени, формальные подходы к решению задачи оценки защищенности ВС из-за трудностей, связанных с формализацией, широкого практического распространения не получили. Значительно более действенным является использование неформальных классификационных подходов к анализу защищенности ВС. Вместо стоимостных оценок при построении и анализе неформальных моделей защиты предполагается в качестве характеристик использовать категорирование объектов: нарушителей (по целям, квалификации и доступным вычислительным ресурсам), информации (по уровням критичности и конфиденциальности), средства защиты (по функциональности и гарантированности реализуемых возможностей) и т. п. Такой подход не позволяет получать точные значения показателей защищенности, однако дает возможность классифицировать ВС по уровню защищенности и сравнивать их между собой. В качестве примера классификационных методик, получивших широкое распространение, можно привести многочисленные критерии оценки безопасности ИТ, принятые во многих странах в качестве национальных стандартов, устанавливающие классы и уровни (показатели) защищенности. Результатом развития национальных стандартов в этой области является, принятый в 1999 г. международный стандарт ISO 15408 («Общие критерии оценки безопасности информационных технологий»), обобщающий существующий мировой опыт в этой области.

Для первичного анализа защищенности вычислительной системы необходимо провести сканирование сети.

ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Средства для выполнения практической работы необходимо:

1. GFI Languard.
2. Загрузить можно по ссылке https://gfi-software.ru/download/gfi-languard/
3. Скриншоты могут отличаться.

4. Windows 2012 Server как хост.
5. Windows Server 2008 - виртуальная машина.
6. Microsoft .NET Framework 2.0.
7. Права администратора GFI LANguard Network Security Scanner.
8. Регистрация на сайте https://gfi-software.ru/download/gfi-languard/ для получения ключа.
9. Код активации, который придет на почту после регистрации.

Как администратор, Вы часто должны иметь дело отдельно с проблемами, связанными с управлением исправлениями и сетевым контролем. Ваша обязанность обеспечить анализ рисков и поддержать безопасное и совместимое сетевое состояние.

Сканирования безопасности или аудиты позволяют Вам идентифицировать и оценить возможные риски в сети. Контролирующие операции подразумевают любой тип проверки выполненного во время аудита сетевой безопасности. Они включают открытые проверки порта, получение недостающих патчей Microsoft, информацию о службе и пользователе и т.д.

Следуя шагам мастера установки, установите GFI LANguard network scanner на хост машину.

1. Нажмите Пуск.

Рисунок 1 - Рабочий стол

2. Запустите GFI LANguard.

Рисунок 2 - Windows Server 2012 – Приложения

3. Главное окно GFI LanGuard 2012.

Рисунок 3 - The GFI LANguard

4. Нажмите запустить сканирование GFI LANguard.

Рисунок 4 - The GFI LANguard

5. Откроется новое окно сканирования.

- В Scan Target, выберите localhost.
- В Profile option, выберите Full Scan.
- В Credentials option, выберите currently logged on user.

6. Нажмите Scan.

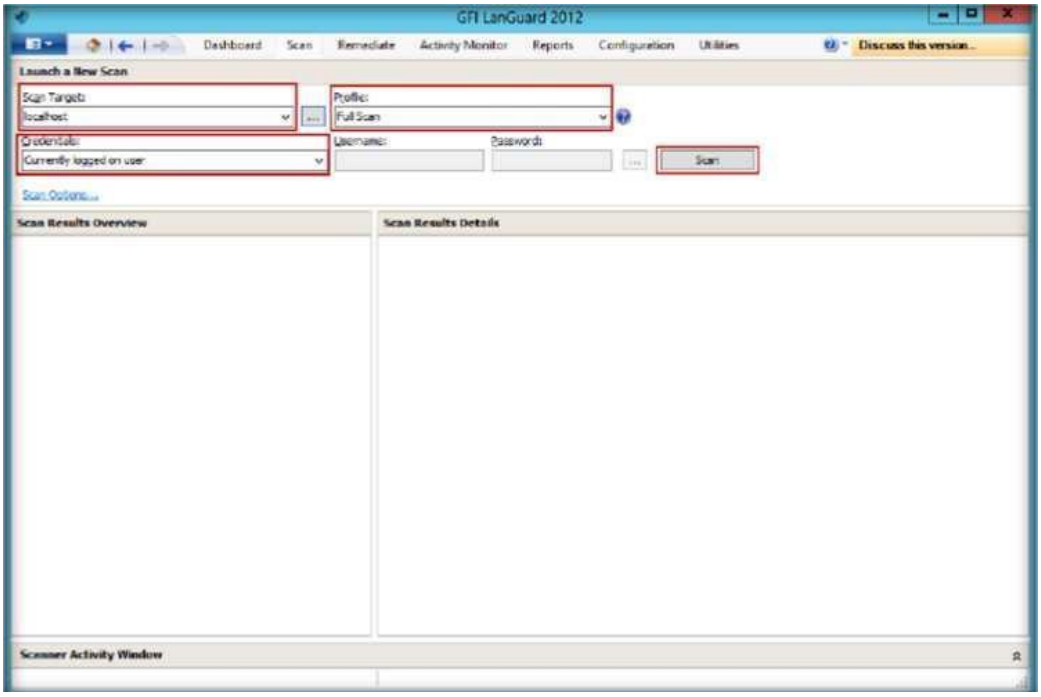


Рисунок 5 - Настройки

7. Сканирование запустилось и оно займёт какое то время, которое указанно ниже.

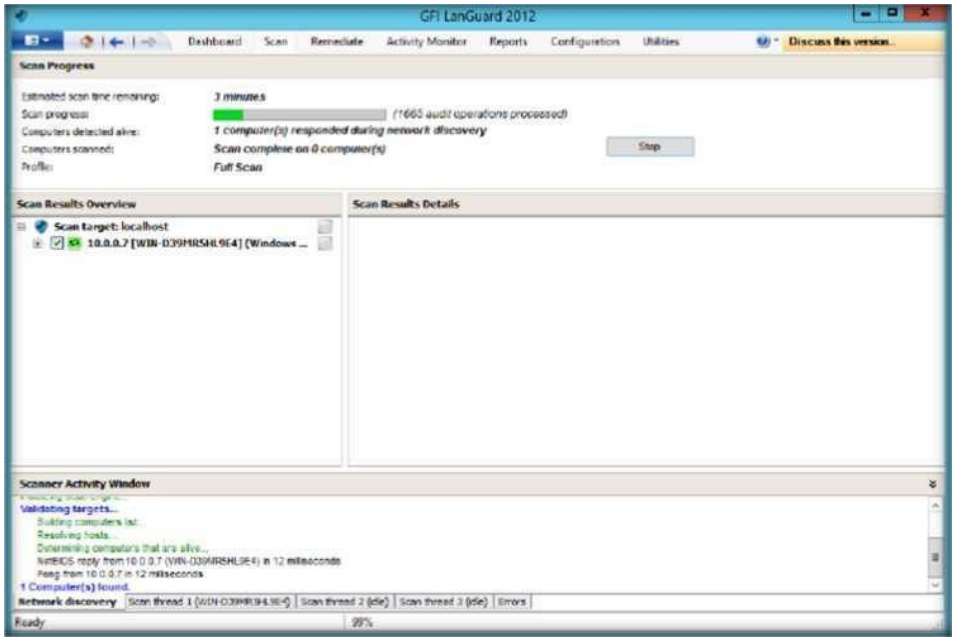


Рисунок 6 - The GFI LanGuard

8. После завершения сканирование, вы увидите результат.

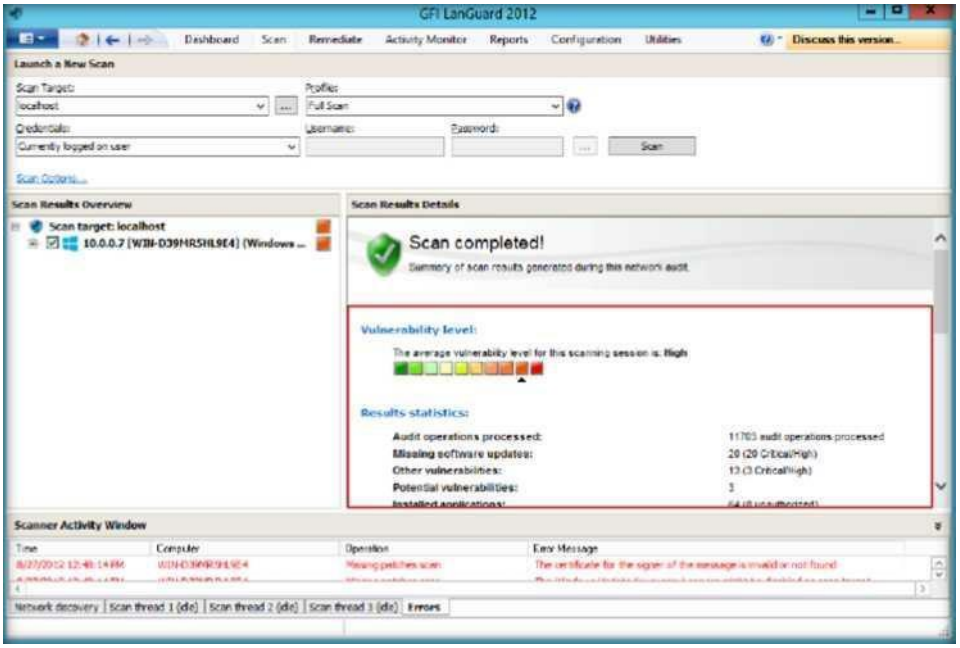


Рисунок 7 - The GFI LanGuard

9. Чтобы просмотреть результат сканирования щелкните по IP адресу.
10. Нажмите Vulnerability Assessment, это покажет оценку уязвимостей и

аудит По и

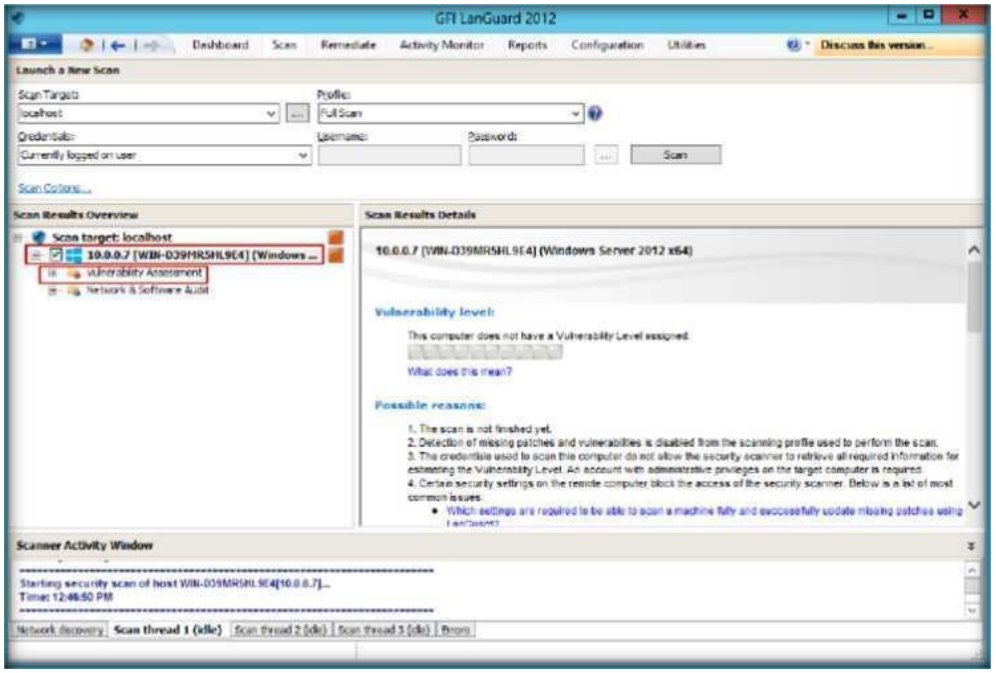


Рисунок 8 - Vulnerability Assessment

11. Это покажет все индикаторы Vulnerability Assessment по категориям.

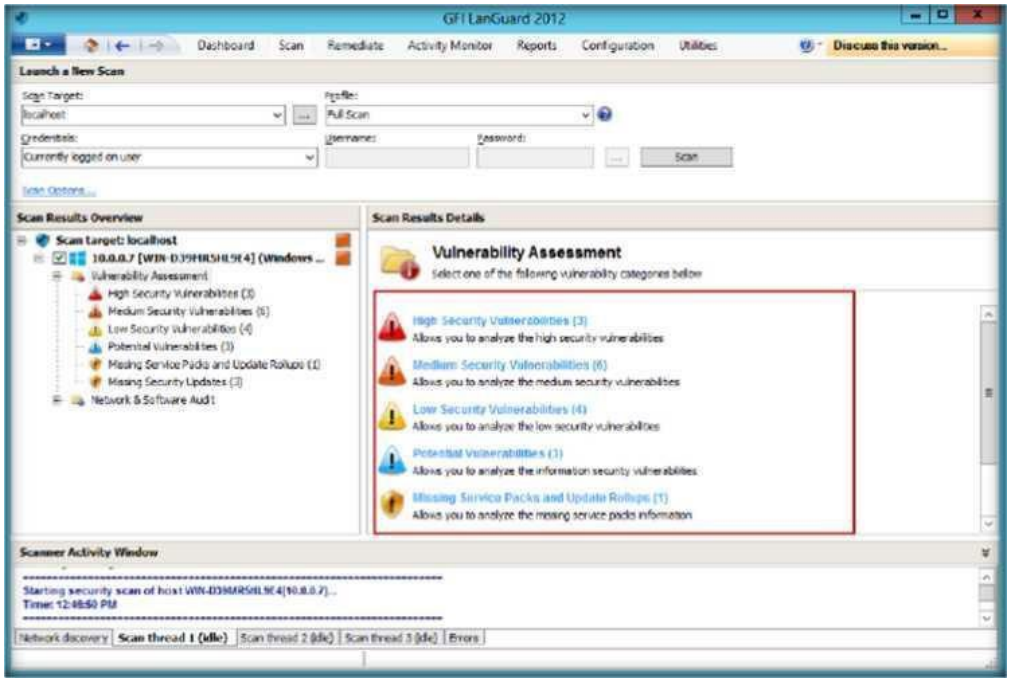


Рисунок 9 - Категории Vulnerability Assessment

12. Нажмите Network & Software Audit и потом System Patching Status, который покажет все системные исправления.

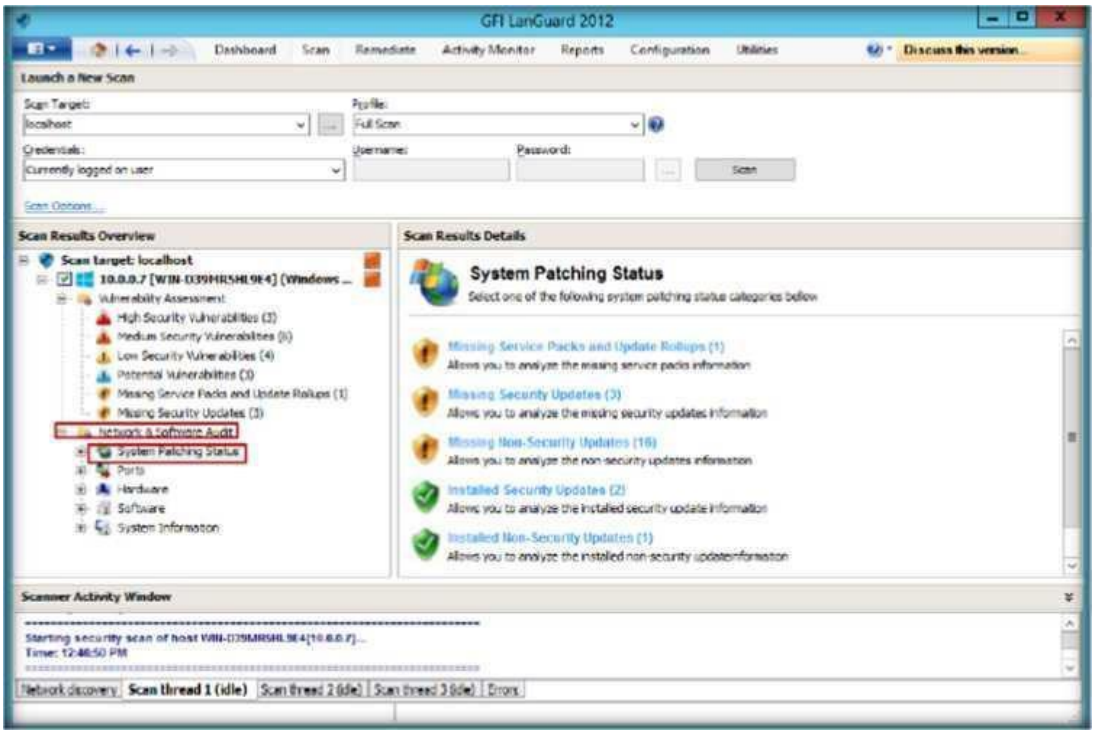


Рисунок 10 - Состояние обновлений

13. Нажмите Ports, затем Open TCP Ports.

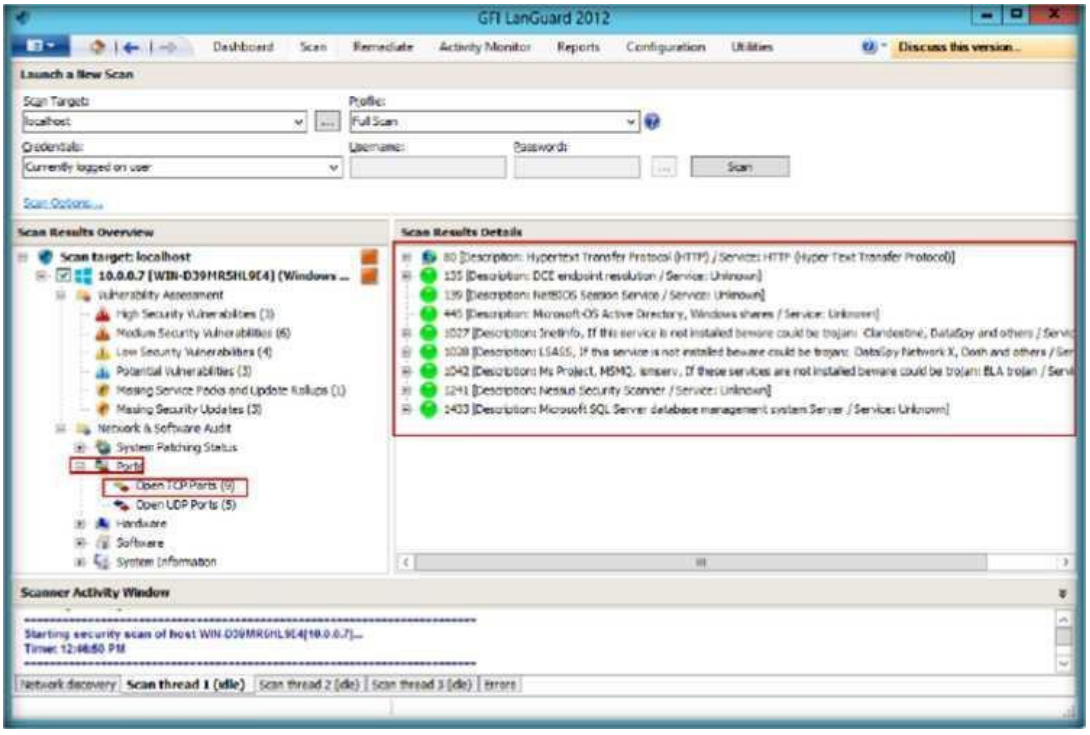


Рисунок 11 - TCP/UDP Порты

14. Нажмите System Information, которая покажет всю системную информацию.

15. Нажмите Password Policy.

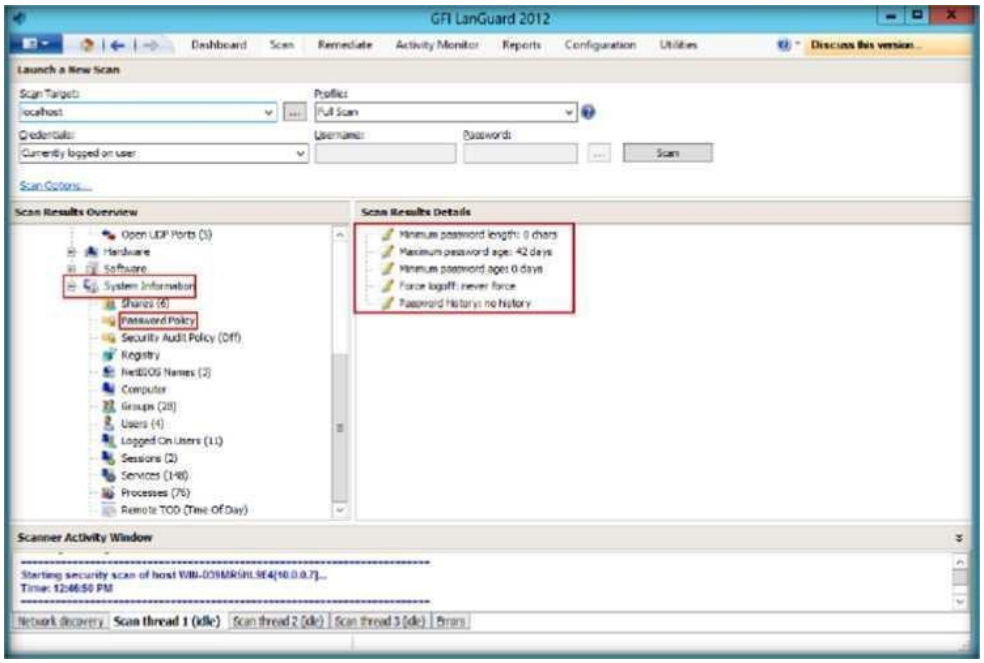


Рисунок 12 - Password Policy

16. Нажмите Groups, которая покажет все группы в системе.

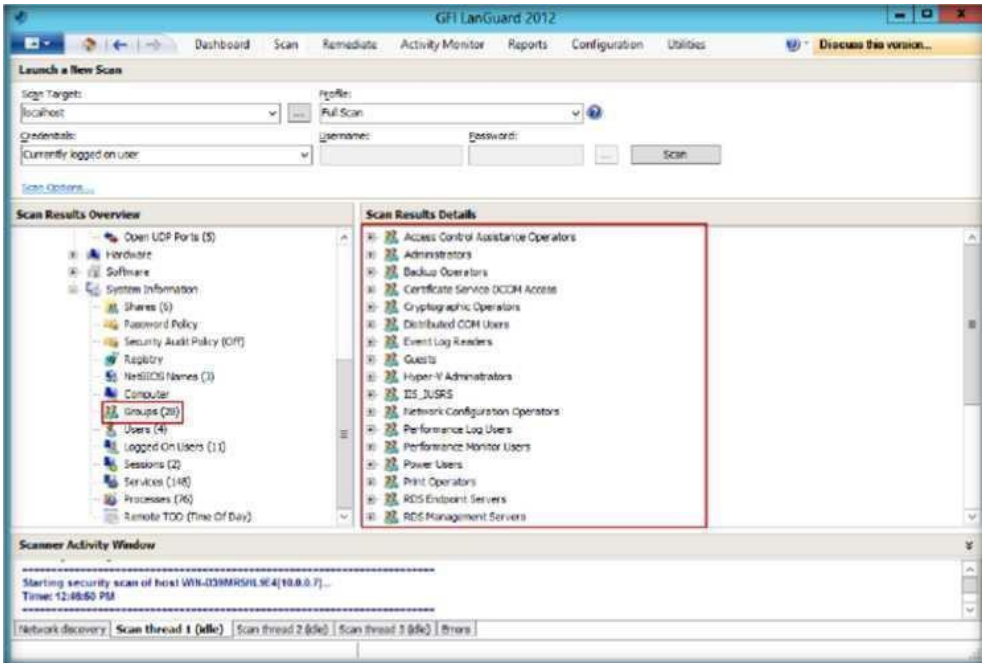


Рисунок 13 - Информация о группах Groups

17. Нажмите Dashboard tab, которая покажет всю информацию сети.

Рисунок 14 - Отчет состояния сети

Запишите все результаты, угрозы и уязвимости, обнаруженные во время сканирования проведенного в практической работе. Все результаты необходимо оформить в следующую таблицу:

Средства	**Собранная информация**
The GFI LANguard	Уровень уязвимостей
	Оценка уязвимостей
	Состояние системного исправления
	Детали сканирования открытых портов TCP
	Детали результатов сканирования для политики паролей
	Вся сеть: Уровень уязвимостей Датчики безопасности Уязвимые компьютеры Состояние агента Тенденция уязвимостей в течении времени

	Распределение уязвимостей

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Проанализируйте как GFI LANguard защищает против червей.
2. При каких обстоятельствах GFI LANguard вывод диалог во время патча.
3. Вы можете изменить сообщение, выведенное на экран, когда GFI LANguard выполняет административные задачи? Если да, то как?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Грибунин В. Г. Комплексная система защиты информации на предприятии [Текст] : учебное пособие / В. Г. Грибунин, В. В. Чудовский. – М.: Академия, 2009. - 416 с.
2. Заика А. Компьютерная безопасность [Электронный ресурс] / А. Заика. - М. : РИПОЛ классик, 2013. - 160 с. // Режим доступа – http://biblioclub.ru/index.php?page=book&id=227317
3. Безбогов А. А., Яковлев А. В., Шамкин В. Н. Методы и средства защиты компьютерной информации [электронный ресурс]: Учебное пособие. – Тамбов: Издательство ТГТУ, 2006.- 196 с. /Электронная библиотека «Единое окно доступа к образовательным ресурсам» – http://window.edu.ru
4. Шаньгин В. Ф. Защита компьютерной информации. Эффективные методы и средства / Шаньгин В. Ф. – М. : ДМК Пресс, 2010.–544 с.

5. Семкин С. Н., Семкин А.Н. Основы правового обеспечения защиты информации. Учебное пособие для вузов. – М.: Горячая линия- Телеком, 2008. - 238 с.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_4_» ____04____ 2022 г.

Шифры полиалфавитной замены

Методические указания для выполнения лабораторных / практических работ
студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00,
12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Шифры полиалфавитной замены: методические указания для выполнения лабораторных / практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 9 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ 1291.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Получение навыков создания зашифрованного сообщения при помощи алгоритма шифрования Виженера

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Шифр Виженера (фр. Chiffre de Vigenère) — метод полиалфавитного шифрования буквенного текста с использованием ключевого слова.

Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джован Баттиста Беллазо (итал. Giovan Battista Bellaso) в книге La cifra del. Sig. Giovan Battista Bellaso в 1553 году, однако в XIX веке получил имя Блеза Виженера, французкого дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.

История

Первое точное документированное описание многоалфавитного шифра было сформулировано Леоном Баттиста Альберти в 1467 году, для переключения между алфавитами использовался металлический шифровальный диск. Система Альберти переключает алфавиты после нескольких зашифрованных слов. Позднее, в 1518 году, Иоганн Трисемус в своей работе «Полиграфия» изобрел tabula recta — центральный компонент шифра Виженера.

То, что сейчас известно под шифром Виженера, впервые описал Джованни Батиста Беллазо в своей книге La cifra del. Sig. Giovan Battista Bellaso. Он использовал идею tabula recta Трисемуса, но добавил ключ для переключения алфавитов шифра через каждую букву. Блез Виженер представил своѐ описание простого, но стойкого шифра перед комиссией Генриха III во Франции в 1586 году, и позднее изобретение шифра было присвоено именно ему. Давид Кан в своей книге «Взломщики кодов» отозвался об этом осуждающе, написав, что история «проигнорировала важный факт и назвала шифр именем Виженера, несмотря на то, что он ничего не сделал для его создания».

Шифр Виженера имел репутацию исключительно стойкого к «ручному» взлому. Известный писатель и математик Чарльз Лютвидж Доджсон (Льюис Кэрролл) назвал шифр Виженера не взламываемым в своей статье «Алфавитный шифр» англ. The Alphabet Cipher, опубликованной в детском журнале в 1868 году. В 1917 году Scientific American также отозвался о шифре Виженера, как о неподдающемся взлому. Это представление было опровергнуто после того, как Казиски полностью взломал шифр в XIX веке, хотя известны случаи взлома этого шифра некоторыми опытными криптоаналитиками ещѐ в XVI веке.

Шифр Виженера достаточно прост для использования в полевых условиях, особенно если применяются шифровальные диски. Например, «конфедераты» использовали медный шифровальный диск для шифра Виженера в ходе Гражданской войны. Послания Конфедерации были далеки от секретных, и их противники регулярно взламывали сообщения. Во время войны командование Конфедерации полагалось на три ключевых словосочетания: «Manchester Bluff», «Complete Victory» и — так как война подходила к концу — «Come Retribution».

Гилберт Вернам попытался улучшить взломанный шифр (он получил название шифр Вернама - Виженера в 1918 году), но, несмотря на его усовершенствования, шифр так и остался уязвимым к криптоанализу. Однако работа Вернама в конечном итоге всѐ же привела к получению шифра, который по-настоящему трудно взломать.

Описание

Квадрат Виженера, или таблица Виженера, также известная как tabula recta, может быть использована для шифрования и расшифрования.

В шифре Цезаря каждая буква алфавита сдвигается на несколько строк; например в шифре Цезаря при сдвиге +3, A стало бы B, B стало бы E и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов или квадрат (таблица) Виженера. Применительно к Русскому алфавиту таблица Виженера составляется из строк по 31 символов, причѐм

каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 31 различных шифров Цезаря. На разных этапах кодировки шифр Виженера использует различные алфавиты из этой таблицы. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. В приложении 1 представлена таблица Виженера.

3. ВЫПОЛНЕНИЕ РАБОТЫ

Предположим, что исходный текст имеет вид:

СООБЩЕНИЕ

Человек, посылающий сообщение, записывает ключевое слово («ОКНО») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

ОКНООКНОО

Первый символ исходного текста С зашифрован последовательностью О, которая является первым символом ключа. Первый символ Я шифрованного текста находится на пересечении строки С и столбца О в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; то есть второй символ шифрованного текста Ш получается на пересечении строки О и столбца К. Остальная часть исходного текста шифруется подобным способом.

Исходный текст: СООБЩЕНИЕ

Ключ: ОКНООКНОО

Зашифрованный текст: ЯШЪПЖПЬЦУ

Расшифровывание производится следующим образом: находим в таблице Виженера строку, соответствующую первому символу ключевого слова; в данной строке находим первый символ зашифрованного текста. Столбец, в котором находится данный символ, соответствует первому символу исходного текста. Следующие символы зашифрованного текста расшифровываются подобным образом.

Если буквы А-Я соответствуют числам 0-32, то шифрование Виженера можно записать в виде формулы:

$$C_i \equiv (P_i + K_i) \bmod 32$$

Расшифровка:

$$P_i \equiv (C_i - K_i + 31) \bmod 32$$

4. ВАРИАНТЫ ЗАДАНИЙ

№	Исходный текст
1	Шумит дубравушка к непогодушке
2	Утром вороны каркают к дождю
3	Сорока на хвосте принесла
4	Снег холодный, а от мороза укрывает
5	Сирень или берѐза, а всѐ дерево
6	Сегодня не тает, а завтра кто знает
7	Розы без шипов не бывает
8	Не высок лесок, а от ветра защищает
9	На всех и солнышко не светит
10	Красна ягодка, да на вкус горька
11	В осеннее ненастье семь погод на дворе
12	Ветром ветра не смеряешь
13	Пропущенный час годом не нагонишь
14	Счастливые часов не наблюдают
15	Друг неиспытанный, как орех не расколотый
16	Дружи с теми, кто лучше тебя самого
17	Крепкую дружбу и топором не разрубишь
18	Кто друг прямой, тот брат родной
19	лучше выслушать упрѐки друга, чем потерять его
20	Одна пчела много мѐду не принесѐт
21	С тем не ужиться, кто любит браниться
22	Старый друг лучше новых двух
23	На чужой сторонушке рад родной воробушке
24	Народы нашей страны дружбой сильны
25	Поднявший меч от меча и погибнет
26	При солнце тепло, при Родине добро
27	Старая слава новую любит
28	Любишь кататься - люби и саночки возить
29	Кто пахать не ленится, у того хлеб родится
30	На печи не храбрись, а в поле не трусь

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Панасенко, С. Алгоритмы шифрования [Текст] / С. Панасенко. Спб: БХВ-Петербург, 2009, 576 стр.

2) Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография [Текст] / А.Г. Ростовцев, Е.Б. Маховенко. М: АНО НПО "Профессионал", 2005, 480 стр.

3) Рябко, Б. Я., Фионов, А. Н. Основы современной криптографии для специалистов в информационных технологиях [Текст] / Б. Я. Рябко, А. Н. Фионов. М: Научный мир, 2004, 179 стр.

4) Смарт, Н. Криптография [Текст] / Н. Смарт. М: Техносфера, 2006, 528 стр.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__»_____2022 г.

Потоковые шифры. Скремблирование бинарного потока данных

Методические указания для выполнения лабораторных и практических работ
студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00,
12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Потоковые шифры. Скремблирование бинарного потока данных: методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 13 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ 1292

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Получение навыков преобразования цифрового потока без изменения скорости передачи с целью получения свойств, близких к свойствам случайной последовательности

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Скремблирование — это обратимое преобразование цифрового потока без изменения скорости передачи с целью получения свойств, близких к свойствам случайной последовательности. Исходное сообщение можно восстановить, применив обратный алгоритм. Применительно к телекоммуникационным системам скремблирование повышает надежность синхронизации устройств, подключенных к линии связи, и уменьшает уровень помех, излучаемых на соседние линии многожильного кабеля. Есть и иная область применения скремблеров — защита передаваемой информации от несанкционированного доступа.

Для синхронной передачи двоичный сигнал должен удовлетворять двум основным требованиям:

1. частота смены символов (1, 0) должна обеспечивать надежное выделение тактовой частоты непосредственно из принимаемого сигнала;

2. спектральная плотность мощности передаваемого сигнала должна быть, по возможности, постоянной и сосредоточенной в заданной области частот с целью снижения взаимного влияния каналов.

Одним из способов обработки двоичных посылок, удовлетворяющим данным требованиям является скремблирование (scramble – перемешивание).

После скремблирования появление «1» и «0» в выходной последовательности примерно равновероятно. Скремблирование также может использоваться для определенной защиты передаваемой информации, а также для идентификации абонентов.

Скремблирование широко применяется во многих видах систем связи для улучшения статистических свойств сигнала. Обычно скремблирование осуществляется на последнем этапе цифровой обработки непосредственно перед модуляцией (Рис. 1).

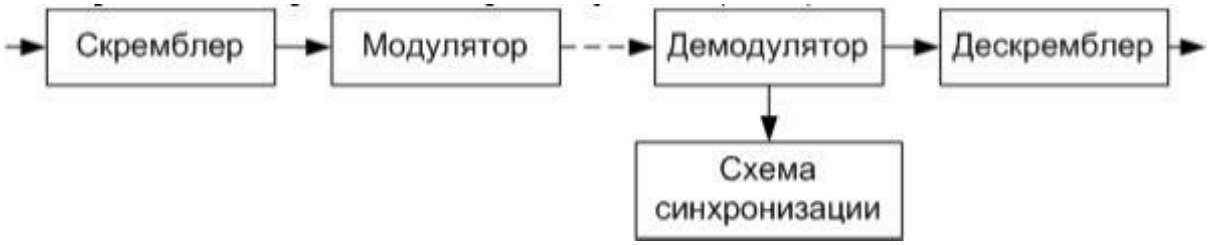


Рис. 1. Схема включения скремблера и дескремблера в канал связи

Скремблирование производится на передающей стороне с помощью устройства – скремблера, реализующего логическую операцию суммирования по модулю 2 исходного и преобразующего псевдослучайного двоичного сигнала. На приемной стороне осуществляется обратная операция – восстановление устройством, называемым дескремблером. Дескремблер выделяет из принятой последовательности исходную последовательность.

Основной частью скремблера является генератор псевдослучайной последовательности (ПСП) в виде линейного n-каскадного регистра с обратными связями, формирующий последовательность максимальной длины $2^n - 1$.

Различают два основных типа скремблеров и дескремблеров - самосинхронизирующиеся (СС) и с установкой (аддитивные). В литературе также можно встретить другие названия – скремблеры с неизолированным и изолированным от линии связи генераторами псевдослучайных последовательностей. Особенностью самосинхронизирующегося скремблера (СС скремблера) (Рис. 2) является то, что он управляется скремблированной последовательностью, т.е. той, которая передается в канал. Поэтому при данном виде скремблирования не требуется специальной установки состояний скремблера и дескремблера; скремблированная последовательность записывается в регистры сдвига скремблера и дескремблера, устанавливая их в идентичное состояние. При потере синхронизма между скремблером и дескремблером время

восстановления синхронизма не превышает числа тактов, равного числу ячеек регистра скремблера.

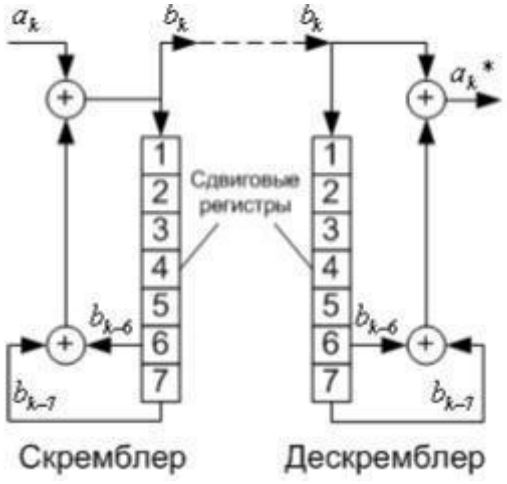


Рис. 2. Самосинхронизирующиеся скремблер и дескремблер.

На приемной стороне выделение исходной последовательности происходит путем сложения по модулю 2 принятой скремблированной последовательности с последовательностью на выходе сдвигового регистра. Например, для схемы Рис. 2 входная последовательность a_k с помощью скремблера в соответствии с соотношением $b_k = a_k \, xor \left(b_{k-6} \, xor b_{k-7}\right)$ преобразуется в посылаемую двоичную последовательность b_k. В приемнике из этой последовательности таким же регистром сдвига, как на приеме, формируется последовательность $a^* = b \, xor \left(b \quad xorb\right)$. Эта последовательность на выходе дескремблера идентична первоначальной последовательности a_k.

$k \qquad k \qquad k-6 \qquad k-7$

Как следует из принципа действия схемы, при одной ошибке в последовательности b_k ошибочными получаются также последующие шестой и седьмой символы (в данном примере). В общем случае влияние ошибочно принятого бита будет сказываться α раз, где α - число обратных связей. Таким

образом, СС скремблер-дескремблер обладает свойством размножения ошибок. Данный недостаток СС скремблера-дескремблера ограничивает число обратных связей в регистре сдвига; практически это число не превышает $\alpha = 2$.

Второй недостаток СС скремблера связан с возможностью появления на его выходе при определенных условиях так называемых «критических ситуаций», когда выходная последовательность приобретает периодический характер с периодом, меньшим длины ПСП. Чтобы предотвратить это, в скремблере и дескремблере предусматриваются специальные дополнительные схемы контроля, которые выявляют наличие периодичности элементов на входе и нарушают ее.

Недостатки, присущие СС скремблеру-дескремблеру, практически отсутствуют при аддитивном скремблировании (Рис. 3), однако, этот тип скремблеров-дескремблеров требует предварительной идентичной установки состояний регистров скремблера и дескремблера. В скремблере с установкой (АД-скремблере), как и в СС скремблере, производится суммирование входного сигнала и ПСП, но результирующий сигнал не поступает на вход регистра. В дескремблере скремблированный сигнал также не проходит через регистр сдвига, поэтому размножения ошибок не происходит.

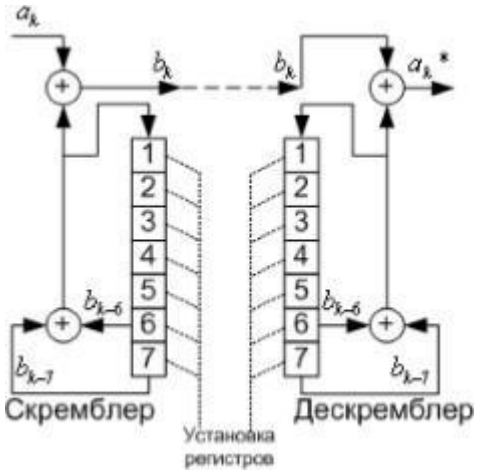


Рис. 3. Аддитивные скремблер и дескремблер.

Суммируемые в скремблере последовательности независимы, поэтому их период всегда равен наименьшему общему краткому величин периодов входной последовательности и ПСП и критическое состояние отсутствует. Отсутствие эффекта размножения ошибок и необходимости в специальной логике защиты от нежелательных ситуаций делают способ аддитивного скремблирования предпочтительнее, если не учитывать затрат на решение задачи фазирования скремблера и дескремблера. В качестве сигнала установки в ЦСП используют

сигнал цикловой синхронизации. Скремблирование так же влияет на энергетический спектр двоичного сигнала (Рис. 4).

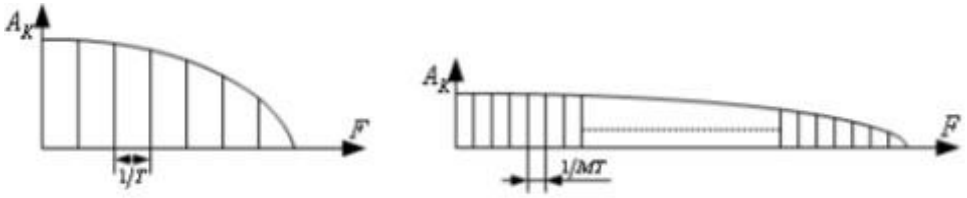
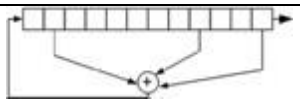
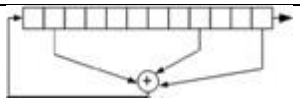
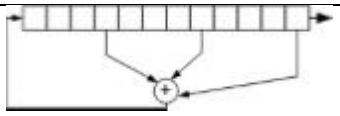
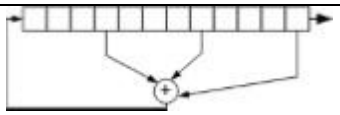
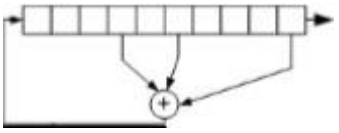
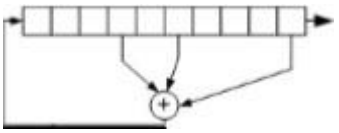
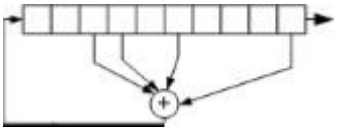
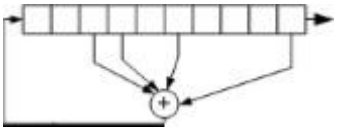
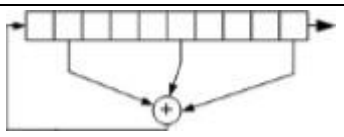
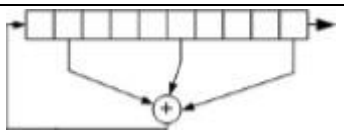


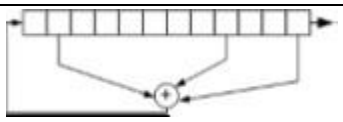
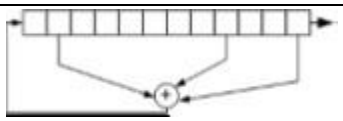
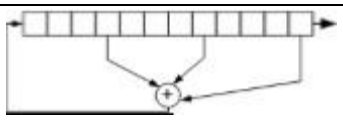
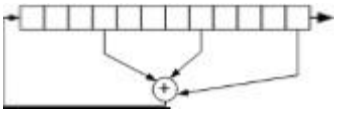
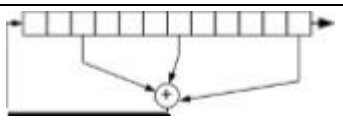
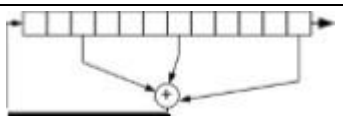
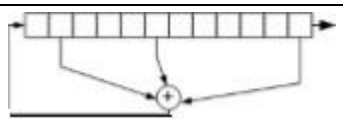
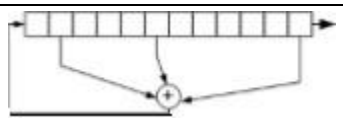
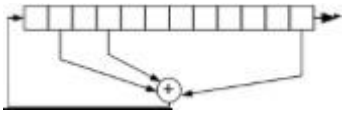
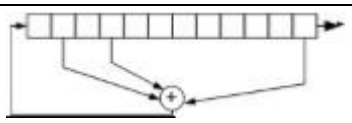
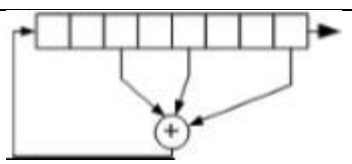
Рисунок 4 – Спектр сигнала а) до скремблирования; б) после скремблирования

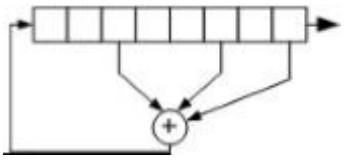
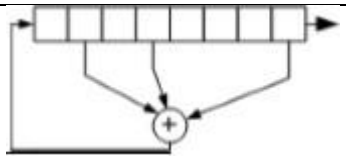
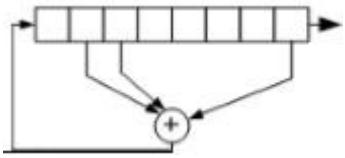
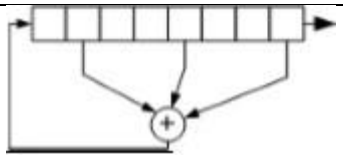
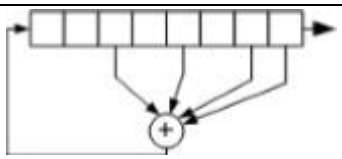
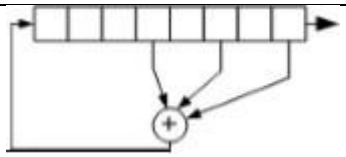
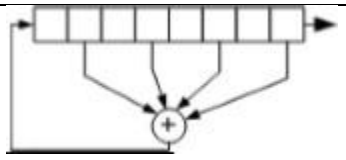
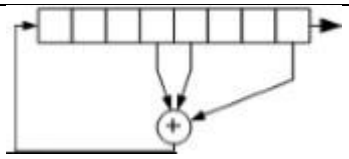
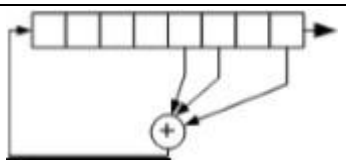
На рисунке 4, а. изображен пример энергетического спектра для периодического сигнала с периодом T, содержащим 6 двоичных элементов с длительностью T_0. После скремблирования $M = 2^n - 1$ элементами ПСП с

спектр существенно «обогащается» (Рис. 4, б). При этом число составляющих спектра увеличилось в M раз, причѐм, уровень каждой составляющей уменьшается в такое же число раз.

3. ВАРИАНТЫ ЗАДАНИЙ

№	Тип скремблера/ дескремблера	Кол - во рег - ов	Сдвиговый регистр	Передаваемое сообщение
1	Самосинхронизирующийся	10		0010010110
2	Аддитивный	10		0010010110
3	Самосинхронизирующийся	10		0000100100
4	Аддитивный	10		0000100100
5	Самосинхронизирующийся	10		1111111111
6	Аддитивный	10		1111111111
7	Самосинхронизирующийся	10		1111011110
8	Аддитивный	10		1111011110
9	Самосинхронизирующийся	10		0000100100
10	Аддитивный	10		0000100100

№	Тип скремблера/ дескремблера	Кол - во рег - ов	Сдвиговый регистр	Передаваемое сообщение
11	Самосинхронизирующийся	12		1111111111
12	Аддитивный	12		1111111111
13	Самосинхронизирующийся	12		000101001101
14	Аддитивный	12		000101001101
15	Самосинхронизирующийся	12		110101001001
16	Аддитивный	12		110101001001
17	Самосинхронизирующийся	12		110001010001
18	Аддитивный	12		110001010001
19	Самосинхронизирующийся	12		001011010100
20	Аддитивный	12		001011010100
21	Самосинхронизирующийся	8		00110010

№	Тип скремблера/ дескремблера	Кол - во рег - ов	Сдвиговый регистр	Передаваемое сообщение
22	Аддитивный	8		00110010
23	Самосинхронизирующийся	8		00101010
24	Аддитивный	8		00101010
25	Самосинхронизирующийся	8		00100011
26	Аддитивный	8		00100011
27	Самосинхронизирующийся	8		10010110
28	Аддитивный	8		10010110
29	Самосинхронизирующийся	8		11100111
30	Аддитивный	8		11100111

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Панасенко, С. Алгоритмы шифрования [Текст] / С. Панасенко. Спб: БХВ-Петербург, 2009, 576 стр.

2) Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография [Текст] / А.Г. Ростовцев, Е.Б. Маховенко. М: АНО НПО "Профессионал", 2005, 480 стр.

3) Рябко, Б. Я., Фионов, А. Н. Основы современной криптографии для специалистов в информационных технологиях [Текст] / Б. Я. Рябко, А. Н. Фионов. М: Научный мир, 2004, 179 стр.

4) Смарт, Н. Криптография [Текст] / Н. Смарт. М: Техносфера, 2006, 528 стр.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__4__» _04_____ 2022 г.

Ассиметричные криптоалгоритмы. Метод RSA

Методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Ассиметричные криптоалгоритмы. Метод RSA: методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 11 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ_____.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Получение навыков создания зашифрованного сообщения при помощи алгоритма шифрования RSA

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Алгоритм *RSA*, первый из алгоритмов шифрования с открытым ключом, достойно выдержал испытание временем. Этот алгоритм основывается на задаче *RSA*, с которой мы познакомились в предыдущем параграфе. Как Вы помните, она сводится к поиску простых делителей больших натуральных чисел. Так что можно утверждать, что криптостойкость алгоритма *RSA* базируется на сложности проблемы факторизации, хотя и не в полной мере, поскольку задачу *RSA* можно решать, не прибегая к разложению на множители.

Предположим, Алиса считает нужным разрешить всем желающим отправлять ей секретные сообщения, расшифровать которые способна только она. Тогда Алиса подбирает два больших простых числа p и q. Держа их в секрете, Алиса публикует их произведение

$$N = p \cdot q$$

которое называют *модулем* алгоритма. Кроме того, Алиса выбирает шифрующую экспоненту *E,* удовлетворяющую условию

$$НОД\left(E,(p-1)(q-1)\right) = 1$$

Как правило E берут равным 3, 17 или 65 537. Пара, доступная всем желающим, — это (N, E). Для выбора секретного ключа Алиса применяет расширенный алгоритм Евклида к паре чисел E и $(p-1)(q-1)$, получая при этом расшифровывающую экспоненту d. Найденная экспонента удовлетворяет соотношению

$$E \cdot d = 1\left(\mathrm{mod}(p-1)(q-1)\right)$$

Секретным ключом является тройка (d, p, q). Фактически, можно было бы выбросить простые делители p и q из ключа и помнить лишь о d и всем числе N.

Допустим теперь, что Боб намерен зашифровать сообщение, адресованное Алисе. Он сверяется с открытым ключом и представляет сообщение в виде числа m, строго меньшего модуля N алгоритма. Шифротекст C получается из га по следующему правилу:

$$C = m^E \pmod N$$

Алиса, получив шифрограмму, расшифровывает еѐ, возводя число C в степень d:

$$m = C^d \pmod N .$$

Равенство имеет место в связи с тем, что порядок группы (равен / $\phi(N) = (p-1)(q-1)$. Поэтому, по теореме Лагранжа,

$$x^{(p-1)(q-1)} = 1 \pmod N$$

для любого числа. Поскольку E и d взаимно обратны по модулю $(p-1)(q-1)$, при некотором целом числе s получается равенство

$$Ed - s(p-1)(q-1) = 1.$$

Следовательно,

$$C^d = (m^E)^d = m^{Ed} = m^{1+s(p-1)(q-1)} = m \cdot m^{s(p-1)(q-1)} = m \pmod N$$

Для прояснения ситуации рассмотрим детский пример. Пусть $p = 7$ и $q = 11$. Тогда $N = 77$, а $(p-1)(q-1) = 6 \cdot 10 = 60$. В качестве открытой шифрующей экспоненты возьмѐм число $E = 37$, поскольку $НОД(37, 60) = 1$.

Применяя расширенный алгоритм Евклида, найдѐм $d = 13$, т. к.

$$37 \cdot 13 = 481 = 1 \pmod{60}.$$

Предположим, нужно зашифровать сообщение, численное представление которого имеет вид: $m = 2$. Тогда мы вычисляем

$$C = m^E \pmod{N} = 2^{37} \pmod{77} = 51.$$

Процесс расшифровывания происходит аналогично:

$m = C^d \pmod N = 51^{13} \pmod{77} = 2$.

3. ВЫПОЛНЕНИЕ РАБОТЫ

В RSA открытый и закрытый ключ состоит из пары целых чисел. Закрытый ключ хранится в секрете, а открытый ключ сообщается другому участнику, либо где-то публикуется.

Генерация ключей RSA

Шифрование начинается с генерации ключевой пары (открытый, закрытый ключ). Генерация ключей в RSA осуществляется следующим образом:

1. Выбираются два простых числа p и q (такие что p неравно q).
2. Вычисляется модуль $N = p \cdot q$.
3. Вычисляется значение функции Эйлера от модуля N: $\phi(N) = (p-1)(q-1)$.Выбирается число еѐ, называемое открытой экспонентой, число e должно лежать в интервале $1 < e < \phi(n)$,а так же быть взаимно простым со значением функции $\phi(N)$.
4. Вычисляется число d, называемое секретной экспонентой, такое, что $d \cdot e = 1 \pmod{\phi(N)}$ то есть является мультипликативно обратное к числу e по модулю $\phi(N)$.

Итак, мы получили пару ключей:

Пара (e, N) - открытый ключ.

Пара $d(N)$ - закрытый ключ.

Шифрование и расшифровывание в RSA

Есть следующий сценарий: Боб и Алиса переписываются в интернете, но хотят использовать шифрование, чтобы поддерживать переписку в секрете. Алиса заранее сгенерировала закрытый и открытый ключ, а затем отправила открытый ключ Бобу. Боб хочет послать зашифрованное сообщение Алисе:

Шифрование: Боб шифрует сообщение m, используя открытый ключ Алисы (e, N) : $C = E(M) = M^e \bmod(N)$, и отправляет с Алисе.

Расшифровывание: Алиса принимает зашифрованное сообщение С. Используя закрытый ключ (d, N), расшифровывает сообщение $M = D(C) = C^d \bmod(N)$.

Теорема Эйлера для понижения степени:

Теорема Эйлера. Для любого модуля ***m*** и целого числа ***a***, взаимно простого с ***m***, справедливо сравнение $a^{\phi(m)} \equiv 1 \bmod m$

Следствие 1 (малая теорема Ферма). Для любого простого числа ***p*** и натурального числа ***a***, взаимно простого с ним, верна формула Ферма:

$a^{p-1} \equiv 1 \bmod p$

Следствие 2 (о вычислении обратного элемента).

$$a^{-1} \equiv a^{\phi(m)-1} \bmod m$$

для любых двух натуральных простых чисел ***a*** и ***m***.

Пример. Вычислите значение выражения $11^{219} \bmod 91$. Решение.

$91 = 7 \cdot 13;$

$\phi(91) = 6 \cdot 12 = 72;$

$(11, 91) = 1$

По теореме Эйлера имеем:

$$11^{219} \bmod 91 = 11^{72 \cdot 3+3} \bmod 91 = (11^{72})^3 \cdot 11^3 \bmod 91 \equiv$$

$$\equiv 11^3 \bmod 91 \equiv 121 \cdot 11 \bmod 91 \equiv 330 \bmod 91 \equiv 57 \bmod 91 = 57$$

Пример шифрования и расшифровывания в RSA

Шифрование:

Выбираем простые числа:

$p = 3, q = 11$

Вычисляем модуль $n = p \cdot \ q = 3 \cdot \ 11 = 33$

Вычисляем функцию Эйлера от модуля n : $\phi\left(N\right) = \left(p - 1\right)\left(q - 1\right) = 2 \cdot \ 10 = 20$.

4. Выбираем открытую экспоненту $e = 7$

5. Определяем закрытую экспоненту $d : d * e = 1\left(\mathrm{mod}\phi\left(N\right)\right) \Rightarrow d = 3$

Будем шифровать сообщение RSA, пусть букве А соответствует цифра 1, В - 2, С - 3 и т.д., тогда:

R=18; S=19; A=1;

Открытый ключ: $\left(e, n\right) = \left(7,33\right)$

$C = \left(18^{7}\right)\mathrm{mod}\ 33 = 6$

$C = \left(19^{7}\right)\mathrm{mod}\ 33 = 13$

$C = \left(1^{7}\right)\mathrm{mod}\ 33 = 1$

$C\left(" \ RSA \ "\right) = 6,13,1$

Пример расшифровывания RSA

Используем закрытый ключ $\left(d, n\right) = \left(3,33\right)$

$M_1 = \left(6^{3}\right)\mathrm{mod}\ 33 = 18$

$M_2 = \left(13^{3}\right)\mathrm{mod}\ 33 = 19$

$M_3 = \left(1^{3}\right)\mathrm{mod}\ 33 =$

$1\ 18 = R; 19 = S; 1 =$

$A;$

Получаем исходное сообщение - RSA.

4. ВАРИАНТЫ ЗАДАНИЙ

№	Исходный текст
1	Шумит дубравушка к непогодушке
2	Утром вороны каркают к дождю
3	Сорока на хвосте принесла
4	Снег холодный, а от мороза укрывает
5	Сирень или берèза, а всѐ дерево
6	Сегодня не тает, а завтра кто знает
7	Розы без шипов не бывает
8	Не высок лесок, а от ветра защищает
9	На всех и солнышко не светит
10	Красна ягодка, да на вкус горька
11	В осеннее ненастье семь погод на дворе
12	Ветром ветра не смеряешь
13	Пропущенный час годом не нагонишь
14	Счастливые часов не наблюдают
15	Друг неиспытанный, как орех не расколотый
16	Дружи с теми, кто лучше тебя самого
17	Крепкую дружбу и топором не разрубишь
18	Кто друг прямой, тот брат родной
19	лучше выслушать упрèки друга, чем потерять его
20	Одна пчела много мѐду не принесѐт
21	С тем не ужиться, кто любит браниться
22	Старый друг лучше новых двух
23	На чужой сторонушке рад родной воробушке
24	Народы нашей страны дружбой сильны
25	Поднявший меч от меча и погибнет
26	При солнце тепло, при Родине добро
27	Старая слава новую любит
28	Любишь кататься - люби и саночки возить
29	Кто пахать не ленится, у того хлеб родится
30	На печи не храбрись, а в поле не трусь

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Панасенко, С. Алгоритмы шифрования [Текст] / С. Панасенко. Спб: БХВ-Петербург, 2009, 576 стр.

2) Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография [Текст] / А.Г. Ростовцев, Е.Б. Маховенко. М: АНО НПО "Профессионал", 2005, 480 стр.

3) Рябко, Б. Я., Фионов, А. Н. Основы современной криптографии для специалистов в информационных технологиях [Текст] / Б. Я. Рябко, А. Н. Фионов. М: Научный мир, 2004, 179 стр.

4) Смарт, Н. Криптография [Текст] / Н. Смарт. М: Техносфера, 2006, 528 стр.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__4__»_____04_____2022 г.

Обработка на базе клеточных автоматов

Методические указания для выполнения лабораторных и практических работ
студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00,
12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Обработка на базе клеточных автоматов: методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 18 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ 1293.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Цель работы: знакомство с применением клеточных автоматов в решении задач математического моделирования, настройка клеточного автомата.

ЗАДАНИЕ

Задание**:** изучить теоретическую часть и пример кода клеточного автомата, найти вкравшуюся в код ошибку. Изменяя входные параметры системы в соответствии с заданием, изучить их влияние на результат работы модели. Подготовить содержательный отчет, сделать выводы.

Для выполнения настоящей работы используется бесплатная программа GNU Octave (http://www.octave.org), имеющая существенную совместимость с Matlab и близость с Octave.

СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист
2. Краткая теория
3. Выполненное задание
4. Вывод

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Клеточный автомат (КА) — дискретная модель, изучаемая в математике, теоретической биологии, физике, гидравлике и т.д. Включает регулярную решётку ячеек, каждая из которых может находиться в одном из конечного множества состояний, таких как 1 и 0.

Решетка может быть любой размерности (соотношение длины сторон). Для каждой ячейки определено множество ячеек, называемых окрестностью.

К примеру, окрестность может быть определена как все ячейки на расстоянии не более 2 от текущей (окрестность фон Неймана ранга 2).

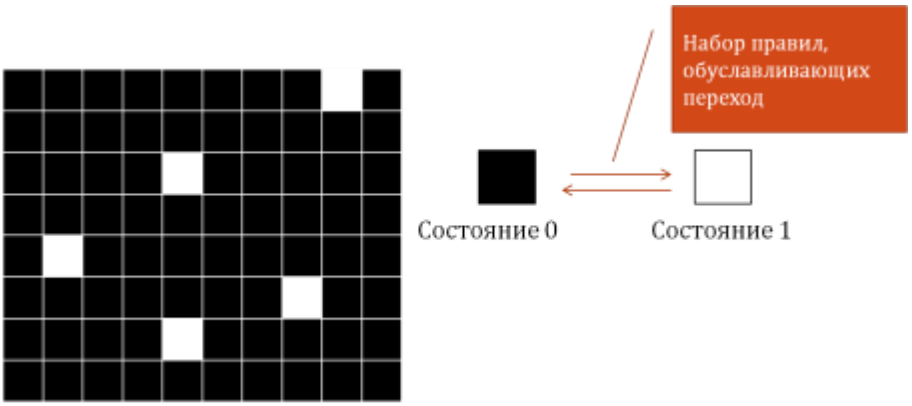


Рис. 1. Схематическое представление простейшего клеточного автомата

Для работы клеточного автомата требуется задание начального состояния всех ячеек, и правил перехода ячеек из одного состояния в другое. На каждой итерации (ходе), используя правила перехода и состояния соседних ячеек, определяется новое состояние каждой ячейки.

Обычно правила перехода одинаковы для всех ячеек и применяются сразу ко всей решётке.

ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Пример: игра «Жизнь» Дж Конвея. Для моделирования поведения популяции одноклеточных простейших используется решетка (матрица) размерностью100x100 элементов. Решетка заселяется некоторым исходным количеством организмов. Затем, с течением времени для всей решетки выполняется ряд правил, управляющих поведением
«организмов», т.е. элементов матрицы:

1) Выживание. Каждая фишка, имеющая вокруг себя две или три соседние фишки, выживает и переходит в следующее поколение.

2) Гибель. Каждая фишка, у которой больше трёх соседей, погибает, то есть снимается с доски из-за перенаселённости.

Каждая фишка, вокруг которой свободны все соседние клетки или же занята всего одна клетка, погибает от одиночества.

3) Рождение. Если число фишек, с которыми граничат какая-нибудь пустая клетка, в точности равно трём (не больше и не меньше), то на этой клетке происходит рождение нового «организма», то есть следующим ходом на неё ставится одна фишка.

Код основного исполняемого модуля игры «Жизнь»

```
 % life.m - Conway's Game of Life
 %
 % A grid of dead and living cells is made.
 % Cells are born to three adjacent parents,
 % and die of overcrowding or loneliness.
 %   Iain Haslam, December 2005

 len=50; GRID=int8(rand(len,len));
 up=[2:len 1]; down=[len 1:len-1]; %the world is
round colormap(gray(2));
 for i=1:1000
         neighbours=GRID(up,:)+GRID(down,:)+GRID(:,up)+GRID(:,down)+..
    .
             GRID(up,up)+GRID(up,down)+GRID(down,up)+GRID(down,d
    own); GRID = neighbours==3 | GRID & neighbours==2;
         image(GRID*2);
    pause(0.02); axis tight;
 end
```

После начала игры, «популяция» быстро начинает сокращаться, уступая место долгоживущим структурам – «глайдерам», которые способны двигаться. Их столкновения друг с другом приводят к разрушению стабильности.

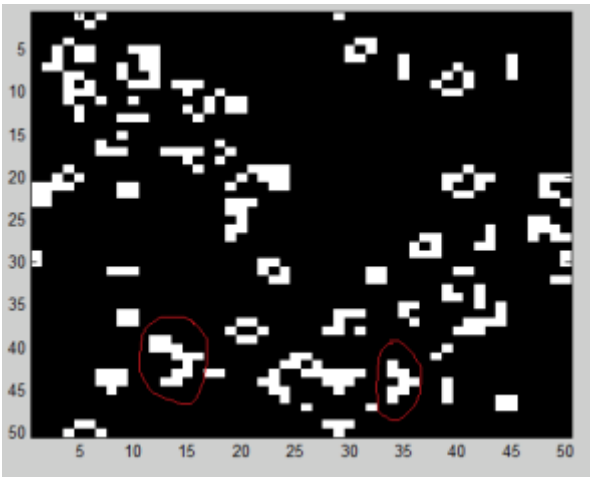


Рис. 1. Долгоживущие самодвижущиеся структуры игры «Жизнь» – «глайдеры» (обведены красным)

Самостоятельно. Разобрать структуру программы «Жизнь». Изучить поведение системы при изменении входных параметров – размеров поля, времени жизни клеток и других.

Рекомендации. В Matlab/Octave для вывода строковых сообщений используются следующие функции. Преобразование числа в строку, выполняемое в Octave функцией string(Число), в Matlab/Octave требует применения функции num2str(Число). Объединение строк, осуществляемое в Octave оператором сложения, в Matlab/Octave требует strcat(Строка1, Строка2,…Строка n).

Существуют и другие несовместимости. При возникновении технических затруднений, необходимо читать справку по функциям, применяемым в Matlab/GNU Octave.

Прервать выполнение программы можно, нажав в командном окне

комбинацию клавиш «Ctrl+C».

Внимание! Начало координат изображения в Matlab/Octave/Octave – левый верхний угол. Столбцы и строки графического изображения развернуты на 90 относительно его матричного представления.

Для создания клеточного автомата возьмем следующую ситуацию. В резервуаре с жидкостью на дно падает дробинка. На нее воздействуют Архимедова сила F_a, стремящаяся вытолкнуть дробинку и сила тяжести F_g, направленная ко дну сосуда. Если плотность жидкости больше плотности дробинки, последняя плавает, в противном случае – тонет (рис. 2). Сопротивлением жидкости и ее вязкостью при падении в ней дробинки следует пренебречь.

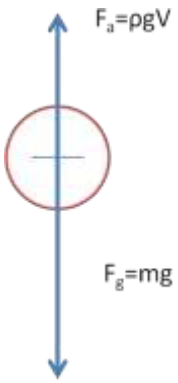


Рис. 2. Схема сил, действующих на падающих в жидкости дробинку. Fg – сила тяжести, Fa – Архимедова сила

Модель, реализуемая в виде программы, должна удовлетворять следующим условиям:

1. Дробинка движется с ускорением;
2. Если $F_a>F_g$ дробинка не тонет, выдается сообщение о плавании дробинки;
3. Максимальный срок выполнения программы 100 кадров (шагов) или момент касания дна;

4. По завершении работы модели выдается график изменения глубины дробинки и ее ускорения (рис. 3).

Простой клеточный автомат, реализующий падение дробинок в воде, показан ниже (pellet_ca.m). Его необходимо запустить, понять как работает и отразить в отчете особенности его работы:

```
 %pellets_ca.m
clear; close all;
 len=50;
GRID=rand(len,len)>0.99;
colormap(gray(2));
 g=9.81; %gravity constant
w_Rho=1000; %density of fluid
p_Rho=11000; %density
p_m=0.050; %mass
p_V=p_m/p_Rho; %volume

 for i=1:50
         Fp=p_m*g.*GRID;
    Fa=w_Rho*g*p_V.*GRID;
    dF=Fp-Fa;
         [r
    c]=find(dF>0);
    r=r+1;
    r(r>=len)=len;
         ind=sub2ind(size(GRID)
    ,r,c); GRID=zeros(len,len);
    GRID(ind)=1;
```

Код Matlab с ошибками для программы, реализующей падение дробинки в резервуар дан в приложении Б. Эта программа, выводит аналитический результат, однако, строго говоря, клеточным автоматом не является (почему?).

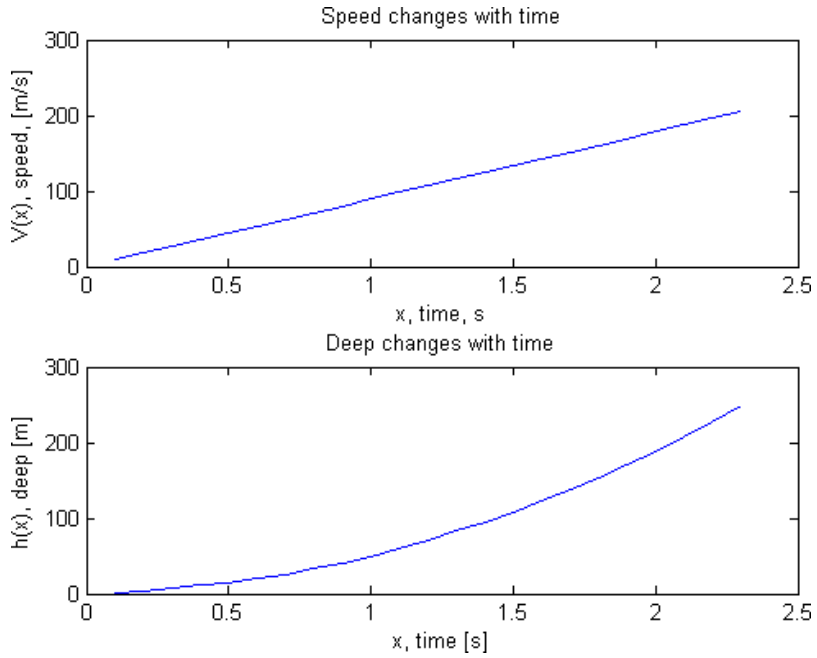


Рис. 3. Графики, создаваемые программой по завершении ее выполнения.

Верхний – изменения скорости со временем, нижний – увеличение глубины со временем.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Исправить код Matlab, найдя ошибки, препятствующие его нормальной работе (pellet_b.m);
2. Используя комментарии в коде программ, детально выяснить порядок их работы, значение переменных и алгоритмы, применяемые в функциях;
3. Понять, и отразить в отчете работу программы pellet_ca.m. Усовершенствовать программу, добавив возможность накопления частиц в несколько слоев;
4. Использовать программу для исследования падения дробинки плотности, заданной в варианте (Приложение А), в жидкости №1 и №2;
5. Описать проводимый эксперимент, сравнить результаты и обосновать выводы;
6. Ответить на вопрос: «Как поведет себя дробинка из свинца в ртути?». Обосновать ответ аналитически.
7. Составить блок-схему работы программы «Падение дроби»

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Алексеев Е.Р. Octave: Решение инженерных и математических задач / Е.Р. Алексеев, О.В.Чеснокова, Е. А.Рудченко // М. : ALT Linux ; БИНОМ. Лаборатория знаний, 2008. 260 с.
2. Павлова М.И. Руководство по работе с пакетом Octave 2.6. 2003. 200 с.
3. Поршнев С. В. Компьютерное моделирование физических процессов в пакете MATLAB / М.:, 2003. 593 с.
4. Эдвардс Ч., Пенни Д. Дифференциальные уравнения и краевые задачи: моделирование и вычисление с помощью Mathematica, Mapple и Matlab / М.: 2008, 1104 с.

Варианты исходных данных для решения самостоятельной задачи.

Параметры: $\rho_{f,}$ №1 – плотность жидкости №1, $\rho_{f,}$ №2 – плотность жидкости №2, ρ_p – плотность дробинки

Вариант	$\rho_{f,}$ №1	$\rho_{f,}$ №2	ρ_p
1	550	2500	9000
2	825	4480	10083
3	597	3999	9912
4	526	3205	8012
5	425	2324	8561
6	756	2293	10051
7	582	4066	9714
8	408	3282	7919
9	579	2143	9800
10	368	4026	9507
11	581	2392	10306
12	386	4149	8249
13	464	2877	7142
14	678	2407	9185
15	513	2863	10175
16	683	4312	9278
17	308	4112	7740
18	658	4433	8567
19	383	3930	7323
20	637	4784	10749
21	309	2688	9958
22	464	4020	11745
23	740	2085	7778

24	726	2170	11316
25	309	4993	7446
26	357	4620	11329
27	593	3250	9555
28	941	4828	10039
29	310	4840	11481
30	433	3626	10787
31	489	2761	7316
32	863	2301	7539
33	570	2547	10720
34	493	3379	10313
35	587	2128	11173
36	964	3426	8776
37	477	3655	8119
38	716	2634	10668
39	446	4891	10074
40	399	3931	10552
41	873	4258	8187
42	312	2434	7582
43	427	2897	11802
44	875	4457	9419
45	546	3897	8065
46	519	3353	11014
47	677	4999	10135
48	896	4985	8429
49	540	4982	9134
50	923	2894	8946

Программа «Падение дробинки»

(с ошибками!)

```
%высота "стакана" h=250 м. Масса дробинки m=0.010 кг. Rho [kg/m3]
%settings
glass=zeros(250,50); %reservoir size of 250x50 px
%environment, gravity
g=9.81;         %gravity constant
%fluid init
w_Rho=1000;    %density of fluid

%pellet initial state and
position p_Rho=11000;
%density
p_xy=[25 1];
%coordinates
p_sp=0;
                %initia
n speed p_sz=2E-
6;              %size
p_m=0.050;
                %mass
p_V=p_m/p_Rho;
%volume

%action
imshow(glass); %output on start

%data logging
```

```
sec_x=[]; %arrays for pellet
state logging speed_y=[];
deep_y=[];

%2 forces act in that model
%Fp=mg Fa=Rho_g_V - gravity and Archimeds
forces description Fp=p_m*g;
Fa=w_Rho*g*p_V;
delta_F=Fa-Fp; %<0 sinks, >0 floats
p_a=abs(delta_F)/p_m; %pellet's acceleration,
nonzero, abs(delta_F) t=0.1; %discretisation of time
in experiment 0.1 second size_glass=size(glass);
%reservoirs height and width

for i=1:100 %100 falling of
 100 frames max
 p_sp=p_sp+p_a; %speed
 acceleration dS=p_sp*t;
 %speed delta

 if (delta_F<0) %pellet sinks,
 because Fa<Fg p_xy=p_xy+[0
 round(dS)]; %enlarge depth
 end;
 if (delta_F>0) %pellet sinks,
      because Fa<Fg disp('Pellet
      is floating!)

 if p_xy(2)>size_glass(1) %checking of
      bottom reaching disp ('bottom is
```

```
        reached');
        break; %break up the cycle
    in that case end;

    %preparation of animated frame
    before output pict_out=glass;
    pict_out(p_xy(2),p_xy(1))=1;

    %changing frame in
    visual picture
    imagesc(pict_out);

    %data logging
    sec_x=[sec_x
    (i*t)];
    speed_y=[spee
    d_y p_sp];
    deep_y=[deep_
    y p_xy(2)];
    %pict heading
    heading=strcat('Frame ' num2str(i),' of 100; ', 'T [sec] =',
    num2str((i*t))); title(heading);
    pau
se
(1);
end;
%final
graphics
output figure;
subplot(2,1,1)
```

```
;
plot(sec_x,sp
eed_y);
title ('Speed changes
with time'); xlabel('x,
time, s'); ylabel('V(x),
speed, [m/s]');
subplot(2,1,2);
plot(sec_x,deep_y);
title ('Deep changes
with time'); xlabel('x,
time [s]');
ylabel('h(x), deep [m]');
```

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«__4__»_____04_____2022 г.

Интеграция механизмов защиты в программное обеспечение

Методические указания для выполнения лабораторных и практических работ
студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00,
12.03.04, 38.05.01, 45.03.03

Курск 2022

УДК 004.056.55

Составители: О.А. Демченко

Рецензент

Кандидат технических наук, доцент *А.Л. Марухленко*

Интеграция механизмов защиты в программное обеспечение: методические указания для выполнения лабораторных и практических работ студентами групп специальностей 02.03.03, 09.00.00, 10.00.00, 11.00.00, 12.03.04, 38.05.01, 45.03.03 / Юго-Зап. гос. ун-т; сост. О.А. Демченко Курск, 2022. - 20 с.

Содержат сведения по вопросам информационной безопасности. Указывается порядок выполнения лабораторной работы, пример выполнения работы, правила оформления, содержание отчета, варианты заданий.

Методические указания разработаны для изучения дисциплин, связанными с безопасностью эксплуатации телекоммуникационных систем.

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл.печ. л._____ . Уч.-изд.л _____. Тираж 30 экз. Заказ_____.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Цель работы: реализовать механизм защиты программного обеспечения от анализа кода программы с помощью метода шифрования исполняемого файла.

ЗАДАНИЕ

Задание**:** овладеть практическими навыками составления шифра вручную, разработки и программирования вычислительного процесса шифрования комбинированным методом с использованием методов перестановки, моно- и поли- алфавитной подстановки, метод гаммирования, сделать отчет и написать вывод.

СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист
2. Краткая теория
3. Выполненное задание
4. Вывод

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Защита программного обеспечения — комплекс мер, направленных на защиту программного обеспечения от несанкционированного приобретения, использования, распространения, модифицирования, изучения и воссоздания аналогов.

Основными инструментами для исследования программ являются дисассемблеры и отладчики.

Дизассемблирование — это получение из исполняемого кода программы код на языке ассемблера.

Дизассемблер - программа, осуществляющая дисассемблирова-ние.

Интерактивный дизассемблер - программа, тесно взаимодействующая с пользователем в процессе дисассемблирования.

Отладчик - программа, предназначенная для анализа поведения другой программы, обеспечивающая остановку в указанных точках и позволяющая просматривать (редактировать) содержимое ячеек памяти, регистров процессора и команды программы.

Эмулирующий отладчик - отладчик, который самостоятельно интерпретирует и выполняет команды программы (без использования реального процессора).

Существует также множество программ-утилит, предназначенных для вспомогательных операций по изучению логики работы механизма защиты. Широко используются

- шестнадцатеричные просмотрщики - редакторы;
- редакторы таблиц экспорта/импорта;
- файловые мониторы, позволяющие отслеживать операции работы с файлами;
- мониторы реестра, создающие протокол обращений к реестру
- и многие другие.

Например, с помощью файлового монитора (FileMonitor) взломщик

может отследить работу защищенной программы с файлами и обнаружить ключ (пароль), хранящийся в некотором файле. Произведя анализ протокола обращений к реестру с помощью монитора реестра (RegMon), взломщик может обнаружить ключ (пароль), хранящийся в системной базе данных Registry.

Инструментарий современного хакера настолько развит, что все попытки авторов защит противодействовать исследованию программ, с точки зрения высококвалифицированных хакеров, считаются безрезультатными.

С помощью современных версий интерактивных дизассемблеров и эмулирующих отладчиков может быть обнаружена практически любая защита.

Тем не менее, отказываться от использования приемов и методов защиты от дизассемблирования кода программы и ее работы под отладчиком нельзя.

Напомним, что абсолютной защиты вообще не бывает. Эффективной защитой считается такая защита, на взлом которой необходимы материальные и трудовые затраты, во много раз превышающие затраты на покупку программного обеспечения. Поэтому затруднение взлома защиты любыми путями и методами является оправданным. Если преодолеть защиту не сможет молодой неопытный взломщик, и заказчику придется обращаться к высококвалифицированному специалисту, - это уже плюс.

Считая защиту от дизассемблирования и отладки, рассчитанной на взломщика средней квалификации, назовем такую защиту затруднением анализа программ. Рассмотрим ее основные моменты.

Универсальным методом противодействия дизассемблированию программы является шифрование. Очевидно, что дизассемблирование зашифрованного кода бесполезно.

Применяя шифрование кода программы для противодействия дизассемблированию, следует учитывать распространенные ошибки реализации данного метода. Напомним их.

Во-первых, неэффективной является такая реализация, когда исполняемый код в полном объеме и однократно шифруется/дешифруется (так как легко найти момент после дешифрования). Во-вторых, необходимо осуществить выбор эффективного ключа и, если необходимо, надежно хранить ключ. И в-третьих, следует учесть, что для защиты программ от дизассемблирования, не рекомендуется использование симметричных криптографических алгоритмов.

Рекомендуется использовать шифрование с открытым ключом (алгоритм RSA, шифр Эль-Гамаля и др.). В этом случае возможная удачная попытка расшифровать код и понять логику работы защитного механизма не позволит внести изменения в защищенный код, так как для полноценной последующей работы программы эти изменения необходимо внедрить в код в зашифрованном виде. А нарушителю доступен лишь ключ для расшифровки. Возможная атака в данном случае - нахождение «секретного» ключа с помощью трудоемких математических вычислений в зависимости от используемого алгоритма шифрования.

Усиливает защиту динамическое шифрование и многопроходная расшифровка кода.

На практике неплохо зарекомендовали себя и методы, использующие вместе с шифрованием архивирование программного кода. К достоинствам данного метода относят и уменьшение размера исполняемого файла. Однако следует учитывать, что алгоритмы работы широко используемых архиваторов известны многим взломщикам.

Широко распространены на практике методы, основанные на динамическом изменении кода программы в процессе выполнения. Суть этих методов сводится к получению истинных исполнимых команд на этапе выполнения программы путем некоторого преобразования первоначальных кодов. Часто этот способ защиты называют самогенерируемыми, или самомодифицирующимися кодами.

Авторы предлагают различные преобразования, например,

- перемещения участков кода;
- всевозможные функции от истинного кода (контрольной суммы истинного кода);
- или для генерации кода одного участка используют коды предыдущего (или какого нибудь другого) участка программы (так называемая обратная связь).

Интересный прием защиты от дизассемблирования - использование нестандартной структуры программы. В этом случае дизассемблер «не поймет» нестандартную сегментацию программы.

Для того чтобы лучше понять методы борьбы с отладчиками и пути увеличения эффективности этих методов, напомним суть процесса работы программы под отладчиком.

Существует два отладочных механизма:

1) контрольные точки останова и
2) трассировка программы.

Контрольные точки останова

Идея механизма заключается во внесении в програм-мный код специального однобайтового кода (0xCC) - так называемой контрольной точки останова.

Заметим, что можно внести в про-граммный код любое количество таких точек.

Напомним, во время выполнения программы при достижении контрольной точки останова возникает исключительная ситуация - прерывание int 3. В этот момент процессор останавливает работу программы для дальнейших распоряжений пользователя. Для того, чтобы позже продолжить работу программы с точки останова, в стеке запоминаются значения регистра флагов, регистра CS (указатель текущего кодового сегмента) и регистра IP (указатель на следующую выполнимую команду).

При этом сбрасывается флаг трассировки.

Итак, для анализа программы с помощью отладчика в исполняемый код программы необходимо внести контрольные точки останова, следовательно, изменить код!

Этим фактом успешно пользовались авторы защит: достаточно было обнаружить модификацию кода и либо удалить точку останова, либо прекратить дальнейшее выполнение программы.

Для обнаружения модифицированного кода традиционно применялись следующие методы:

- подсчет контрольных сумм критических участков;
- использование контрольной суммы всего кода для расшифровки некоторого фрагмента;
- многопроходная расшифровка кода с ключом, вычисляемым на основе контрольной суммы всего кода либо критического участка;
- использование корректирующих кодов, позволяющих определить местоположение контрольного байта;
- контроль времени выполнения критического участка по сравнению с эталонным временем;
- контроль относительного времени выполнения участка программы (относительно другого участка)
- и другие.

Интересные методы противодействия отладчикам реального (!) режима основаны на перехвате необходимого отладчику прерывания int 3. Предлагается заменить команды обработчика прерывания int 3 «мусором» либо использовать обработчик прерывания в целях защиты, например, для расшифровки кода. Тогда в первом случае отладчик будет просто

нейтрализован, а во втором - возникнет конфликт между механизмом защиты и отладчиком.

Существует целая группа приемов, основанных на использовании отладчиком стека.

Одним из таких приемов является следующий. При выполнении критического участка необходимо присвоить указателю стека нулевое значение. Таким образом, стек считается полным и не может быть использован для сохранения необходимых регистров. Операционная система в таком случае завершает работу приложения.

Другой прием - хранить в стеке (полностью используя стек) данные, необходимые для работы программы. Отладчик, при использовании стека для записи значений необходимых регистров, удалит (затрет) критические данные, и программа станет неработоспособной.

Рассмотрим второй отладочный механизм - трассировку кода программы.

Трассировка - это пошаговое выполнение программы.

Напомним, установка специального флага трассировки (TF) приводит к генерированию после каждой команды исключительной ситуации - прерывания int 1, называемого трассировочным прерыванием.

При этом, аналогично обработке исключительной ситации int 3, в стеке сохраняются необходимые для дальнейшей работы значения регистра флагов и регистра IP (указатель на следующую выполнимую команду).

Поэтому наряду с простыми проверками - установлен ли флаг трассировки – для противодействия трассировке могут быть использованы и перечисленные выше приемы.

Очевидно, что трассировщик будет стараться следить за флагом TF и корректировать его. Поэтому необходимо для получения оригинального флага трассировки использовать специальные приемы.

Проверять установку флага трассировки можно, исходя из аппаратных особенностей процессора, например, используя потерю трассировочного

прерывания. Этот метод базируется на том, что после команд, изменяющих сегментный регистр SS (до процессора Intel 80386 - любой сегментный регистр), не происходит трассировочное прерывание даже при установленном флаге трассировки. Чтобы получить истинное значение флага трассировки, достаточно перед проверкой регистра флагов переслать сегментный регистр SS.

Для усиления защиты от пошагового выполнения программы авторы успешно используют значение флага трассировки (в совокуп-ности с другими параметрами) для расшифровки критических участ-ков или, что еще сложнее для взлома, в арифметических выражениях.

Современные операционные системы и отладчики позволяют установить аппаратные точки останова.

Аппаратная отладка основана на использовании специальных отладочных регистров (напомним, что их всего 8 регистров) и возможности простановки четырех контрольных точек останова.

Аппаратные точки останова никак не модифицируют код и, следовательно, не могут быть обнаружены традиционными средствами.

Однако существуют приемы, позволяющие, в принципе, обнаружить работу программы под аппаратной отладкой. Для этого можно использовать все отладочные регистры для нужд программы или поместить в отладочные регистры «мусор». Усиливается защита использованием всех четырех контрольных точек останова. Но реализация таких приемов требует высокой квалификации программиста и вряд ли оправдана, так как для взлома защиты нарушитель может воспользоваться другими технологиями и инструментами.

Для взлома механизмов защиты сегодня применяются мощные эмулирующие отладчики. Они самостоятельно (без помощи процессора) интерпретируют и выполняют команды исследуемой программы. Существуют так называемые отладчики с неполной эмуляцией. Эти отладчики интерпретируют только некоторые команды, а остальные выполняют на реальном процессоре.

Очевидно, что против таких отладчиков бессильны любые приемы противодействия. Поэтому «сегодня уже мало кто решается противодействовать отладчику и включать в свое приложение антиотладочный код. Мода на это давно прошла».

Единственным средством борьбы с отладкой сами хакеры называют эмуляторы процессора.

В рамках примера выполнения практической работы будет реализовано двойное гаммирование. Программа дважды просит пользователя ввести строку-ключ для шифрования. Затем каждый раз введённая строка преобразуется в число, которым инициализируется генератор псевдослучайных чисел. После этого генератор порождает последовательность псевдослучайных чисел такой же длины, что и длина файла и потом каждый байт этой последовательности накладывается по логической операции XOR с соответствующим байтом файла. Для надёжности опять зашифруем уже зашифрованный файл таким же образом. Дешифрование происходит точно также и поэтому одна и та же программа используется для шифрования и для дешифрования.

ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Воспользуемся методом перестановок. Идея этого метода криптографии заключается в том, что запись открытого текста и последующее считывание шифровки производятся по разным путям внутри некоторой геометрической фигуры (например, квадрата).

Для пояснения идеи возьмем квадратную таблицу (матрицу) размером 8 х 8 (будем записывать текст последовательно по строкам сверху вниз, а считывать -последовательно по столбцам слева направо.

Предположим, что требуется зашифровать сообщение:

НА ПЕРВОМ КУРСЕ ТЯЖЕЛО УЧИТЬСЯ ТОЛЬКО ПЕРВЫЕ ЧЕТЫРЕ ГОДА ДЕКАНАТ

Запишем его в матрицу:

	1	2	3	4	5	6	7	8
1	Н	А		П	Е	Р	В	0
2	М		К	У	Р	С	Е	
3	Т	Я	Ж	Е	Л	0		У
4	Ч	И	Т	Ь	С	Я		Т
5	0	Л	Ь	К	0		П	Е
6	Р	В	Ы	Е		Ч	Е	Т
7	Ы	Р	Е		г	0	Д	А
8		Д	Е	К	А	Н	А	Т

В матрице символом «_» обозначен пробел. В результате преобразований получится шифровка:

НМТЧРЫ_А_ЯИЛВРД_КЖТЬЫЕЕПУЕЬКЕ_КЕРЛСО_ГАРСОЯ_ЧОНВЕ_ПЕДАО_УТЕТ АТ.

Ключом в данном случае является размер матрицы, порядок записи открытого текста и считывания шифрограммы. Естественно, что ключ может быть другим. Например, запись открытого текста по строкам может производиться в таком порядке номеров строк:
4 8 1 2 7 6 5 3 , а считывание криптограммы может происходить по столбцам в следующем порядке: 8 1 3 5 7 6 4 2 .

Будем называть порядок записи в строки матрицы ключом записи, а порядок считывания шифрограммы по столбцам — ключом считывания.

Чтобы дешифровать криптограмму, полученную с помощью матрицы размером п х п, нужно разбить эту криптограмму на группы символов по п символов в каждой группе. Крайнюю левую группу записать сверху вниз в столбец, номер которого совпадает с первой цифрой ключа считывания. Вторую группу символов записать в столбец, номер которого совпадает со второй цифрой ключа считывания, и т. д. Открытый текст считывать из матрицы по строкам в соответствии с цифрами ключа записи.

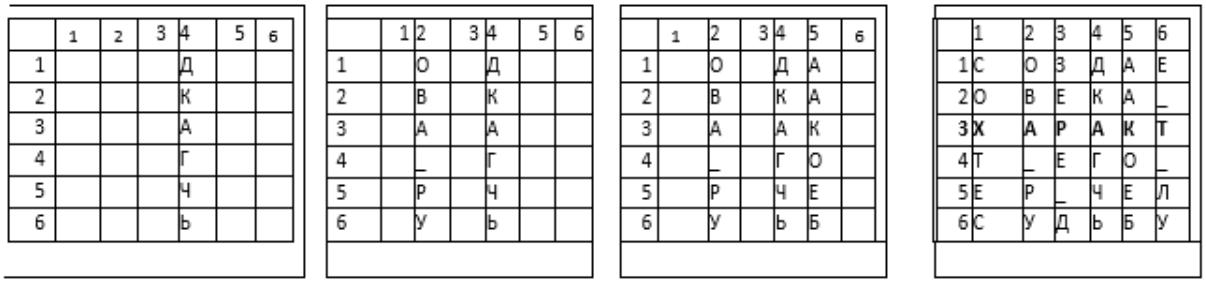
Рассмотрим пример дешифрации криптограммы, полученной методом перестановок. Известно, что при шифровании использованы матрица размером 6х6, ключ записи 3 5 2 1 4 6 и ключ считывания 4 2 5 3 1 6 . Текст шифрограммы таков:

ДКАГЧЬОВА_РУААКОЕБЗЕРЕ_ДСОХТЕСЕ_Т_ЛУ

Разобьем шифрограмму на группы по 6 символов:

ДКАГЧЬ ОВА_РУ ААКОЕБ ЗЕРЕ_Д СОХТЕС Е_Т_ЛУ

Первую группу символов запишем в столбец 4 матрицы, так как первая цифра ключа считывания — 4. Вторую группу из 6 символов запишем в столбец 2, третью группу символов — в столбец 5 и т. д.:



	1	2	3	4	5	6
1				Д		
2				К		
3				А		
4				Г		
5				Ч		
6				Ь		

	1	2	3	4	5	6
1		О		Д		
2		В		К		
3		А		А		
4		_		Г		
5		Р		Ч		
6		У		Ь		

	1	2	3	4	5	6
1		О		Д	А	
2		В		К	А	
3		А		А	К	
4		_		Г	О	
5		Р		Ч	Е	
6		У		Ь	Б	

	1	2	3	4	5	6
1	С	О	З	Д	А	Е
2	О	В	Е	К	А	_
3	Х	А	Р	А	К	Т
4	Т	_	Е	Г	О	_
5	Е	Р	_	Ч	Е	Л
6	С	У	Д	Ь	Б	У

Считывание открытого текста в соответствии с ключом записи начинаем со строки 3, затем считываем строку 5 и т. д. В результате дешифрования получаем открытый текст:

ХАРАКТЕР ЧЕЛОВЕКА СОЗДАЕТ ЕГО СУДЬБУ

Естественно, что описанную процедуру дешифрования криптограммы можно произвести автоматически на компьютере с помощью заранее разработанных программ.

Эффективным средством повышения криптостойкости шифров является комбинированное использование нескольких различных способов шифрования с помощью двух или более методов.

Как показали исследования, стойкость комбинированного шифрования Sк не ниже произведения стойкости используемых способов Si , т.е. Sк>= П Si

Комбинировать можно любые методы шифрования и в любом количестве, однако на практике наибольшее распространение получили следующие комбинации:

1. Перестановка + подстановка(замена)
2. Перестановка + гаммирование
3. Гаммирование+ подстановка(замена)
4. Гаммирование+ гаммирование

Типичным примером комбинированного шифра является национальный стандарт США криптографического закрытия данных (DES).

Английский алфавит

A B C D E F G H I J K L M N O P Q P S T U V W X Y Z 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1.

Составить программу , которая шифрует сообщение Complete support of RAR and ZIP archives следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром по Таблице Виженера:

1. *3 уровень*

C. Дополнить полученную программу шифром Атбаш, расположив буквы построчно.

Вариант 2.

Составить программу , которая шифрует сообщение Make your own settings by modifying or creating a KBD file следующим образом:

1 уровень

A. Составить программу , которая первым использует шифр перстановок.

2 уровень

B. Дополнить полученную программу, которая использует Аффинные криптосистемы a=5 и в=3

2. *3 уровень*

C. Дополнить полученную программу шифром Квадрата Бьюфорта, расположив буквы построчно.

Вариант 3.

Составить программу , которая шифрует сообщение Use Script to assign a keystroke to a script command or function следующим образом:

1 уровень

A. Составить программу которая использует метод перестановок.

2 уровень

B. Дополнить полученную программу, которая использует Аффинные криптосистемы a=5 и в=7

3. *3 уровень*

C. Дополнить полученную программу шифром по Таблице Виженера, расположив буквы построчно.

Вариант 4.

Составить программу , которая шифрует сообщение *You can map your keyboard the following ways* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу, которая шифрует сообщение по Таблице Бьюфорта

4. *3 уровень*

C. Дополнить полученную программу, которая шифрует сообщение Квадратом

Полибия [6x6]

Вариант 5.

Составить программу , которая шифрует сообщение *In the IDE, the keystroke repeat rate defaults to OFF* следующим образом:

1 уровень

A. Составить программу, которая которая использует метод перестановок

2 уровень

B. Дополнить полученную программу, которая шифрует сообщение шифром Атбаш

5. *3 уровень*

C. Дополнить полученную программу, которая шифрует сообщение квадратом Полибия [6x6]

Вариант 6.

Составить программу , которая шифрует сообщение *Click one of the following buttons to open a KBD file in Notepad* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 *уровень*

B. Дополнить полученную программу, которая шифрует сообщение квадратом Бьюфорта

6. *3 уровень*

C. Дополнить полученную программу, которая шифрует сообщение Таблицей Виженера

Вариант 7.

Составить программу , которая шифрует сообщение *Changing the mapping of your keyboard* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу, которая использует шифр Цезаря

7. *3 уровень*

C. Дополнить полученную программу, которая шифрует сообщение Афинными криптосистемами a=9, b=3

Вариант 8.

Составить программу , которая шифрует сообщение *The available Editor SpeedSettings are* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу, которая использует Аффинные криптосистемы a=5 и в=9

8. *3 уровень*

C. Дополнить полученную программу шифром Квадратом Полибия [6x6]

Вариант 9.

Составить программу , которая шифрует сообщение *To customize how text is displayed*

следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок.

2 уровень

B. Дополнить полученную программу шифром, который при закрытии сообщения использует шифр Цезаря

9. *3 уровень*

C. Дополнить полученную программу шифром, который закрывает сообщение Квадратом Полибия [6x6]

Вариант 10.

Составить программу , которая шифрует сообщение *Select the type of text you want* следующим образом:*уровень*

A. Составить программу, которая использует метод перестановок

1 уровень

B. Дополнить полученную программу шифром Атбаш.

10. *3 уровень*

C. Дополнить полученную программу шифром Цезаря.

Вариант 11.

Составить программу , которая шифрует сообщение *The number of available text types* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром, который закрывает сообщение Таблицей Виженера.

11. *3 уровень*

C. Дополнить полученную программу шифром Цезаря.

Вариант 12.

Составить программу , которая шифрует сообщение *Choose a font, size, attribute, and color* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром по Таблице Виженера

12. *3 уровень*

C. Дополнить полученную программу шифром, который закрывает сообщение Квадратом Полибия [6x6]

Вариант 13.

Составить программу , которая шифрует сообщение *For example,*

only one type of text следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром Атбаш.

13. *3 уровень*

C. Дополнить полученную программу шифром, который закрывает сообщение Таблицей Виженера.

Вариант 14.

Составить программу , которая шифрует сообщение *Makes the specified text display in boldface* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром Цезаря.

14. *3 уровень*

C. Дополнить полученную программу шифром Атбаш

Вариант 15.

Составить программу , которая шифрует сообщение *Select a view and the type of text*

следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу шифром Цезаря.

15. *3 уровень*

C. Дополнить полученную программу Таблицей Виженера

Вариант 16

Составить программу , которая шифрует сообщение *Changing the mapping of your keyboard* следующим образом:

1 уровень

A. Составить программу, которая использует метод перестановок

2 уровень

B. Дополнить полученную программу, которая использует шифр Атбаш

16. *3 уровень*

C. Дополнить полученную программу, которая шифрует сообщение Афинными криптосистемами a=9, b=3

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Ярочкин, В.И. Информационная безопасность: учебник для вузов по гуманит. и социал.-экон. спец. / В.И. Ярочкин – М.: Академический Проект: Трикста, 2005. – 543 с. (шифр: 004 Я805).
2. Башлы, П.Н. Информатика: учебное пособие / П.Н. Башлы. – Ростов-н/Д.: Феникс, 2006. – 250 с. (шифр: 004 Б335).
3. Торокин, А.А. Инженерно-техническая защита информации: учебн. пособие для студентов, обучающихся по специальности в области информ. безопасности / А.А. Торокин. – М.: Гелиос АРВ, 2005. – 960 с.
4. Зайцев, А.П. Техническая защита информации: учебн. пособие / А.П. Зайцев, А.А. Шелупанов. – М.: Горячая линия-Телеком, 2007. – 616 с.
5. Хореев, А.А. Техническая защита информации: учеб. пособие для студентов вузов. В 3 т. Том 1. Технические каналы утечки информации / А.А. Хореев. – М.: НПЦ «Аналитика», 2008. – 436 с.