Практическое занятие №16

«Исследование кода вредоносных программ»

Цели: изучение методик и техник анализа бинарного кода вредоносных программ с использованием специального ПО.

Теоретические сведения:

Методы анализа вредоносных программ

Чаще всего при анализе вредоносного программного обеспечения в нашем распоряжении будет только бинарный код, то есть исполняемый файл или библиотека, скомпилированный в двоичном виде. Для того чтобы понять, как этот файл, а точнее его бинарный код работает, нужно будет использовать специальные инструменты и приемы.

Существует два основных подхода к анализу вредоносных программ: **статический** и **динамический**. При статическом анализе вредоносные программы изучают, не запуская вредоносный код на исполнение. Динамический же анализ включает в себя запуск вредоносных программ и манипуляции с запущенным процессом в оперативной памяти.

Также оба метода условно можно разделить на **базовый** и **продвинутый** анализ. **Базовый статический анализ** состоит из изучения исполняемого файла без просмотра машинных инструкций. По сути, это первичный анализ, который может либо подтвердить, либо опровергнуть предположение о том, что файл вредоносен. **Базовый динамический анализ** связан с запуском вредоносного кода и наблюдением его поведения в целевой системе с помощью специальных инструментов. **Продвинутый статический анализ** подразумевает под собой загрузку исполняемого файла в дизассемблер без запуска кода в оперативной памяти и просмотр ассемблерных инструкций на предмет того, что делает код программы в целевой системе. **Продвинутый динамический анализ** использует отладчик для изучения внутреннего состояния выполняемого кода в оперативной памяти.

Дизассе́мблер — транслятор, преобразующий машинный код, объектный файл или библиотечные модули в текст программы на языке ассемблера.

По режиму работы с пользователем делятся на

- Автоматические
- Интерактивные

Примером автоматических дизассемблеров может служить Sourcer. Такие дизассемблеры генерируют готовый листинг, который можно затем править в текстовом редакторе. Пример интерактивного — IDA. Он позволяет изменять правила дизассемблирования и является весьма удобным инструментом для исследования программ.

Дизассемблеры бывают **однопроходные и многопроходные**. Основная трудность при работе дизассемблера — отличить данные от машинного кода, поэтому на первых проходах автоматически или интерактивно собирается информация о границах процедур и функций, а на последнем проходе формируется итоговый листинг. Интерактивность позволяет улучшить этот процесс, так как

просматривая дамп дизассемблируемой области памяти, программист может сразу выделить строковые константы, дать содержательные имена известным точкам входа, прокомментировать разобранные им фрагменты программы.

Чаще всего дизассемблер используют для анализа программы (или ее части), исходный текст которой неизвестен — с целью модификации, копирования или взлома. Реже — для поиска ошибок (багов) в программах и компиляторах, а также для анализа оптимизации создаваемого компилятором машинного кода. Обычно однопроходный дизассемблер (как и построчный ассемблер) является составной частью отладчика.

Защита от дизассемблирования

Первое направление защиты, как правило, реализуется значительно легче, чем второе, поэтому будет приведен лишь краткий обзор данного направления. При реализации защиты программ от дизассемблирования можно применять различные приемы.

Среди них наиболее часто используемым и эффективным приемом является зашифровка и/или запаковка отдельных участков исходного кода или всего кода целиком, при этом необходимо позаботиться о распаковке/расшифровке программы на точке входа. Таким образом, при просмотре исполняемого машинного кода исполняемого файла вместо рабочего кода программы будет отображен лишь бессмысленный набор операций. При реализации защиты от дизассемблирования используется также множество приемов, которые реализуются с целью запутать потенциального взломщика. Можно навести несколько примеров такого вида приемов:

- увеличение исходного кода программы добавлением множества «бессмысленных» операций, а рабочий участок программы записать в определенное место этого множества;
- замена местами адресов обработчиков (векторов) прерываний, например, поменять местами вектор прерывания видео сервиса (INT 10h) с вектором прерывания сервиса DOS (INT 21h), после такой замены для вызова из программы какой-либо функции прерывания INT 21h необходимо пользоваться вызовом прерывания INT 10h.

Практическое задание:

Порядок выполнения работы:

1. Ознакомиться с информацией https://xakep.ru/2016/12/08/reversing-malware-tutorial-part1/#toc03 .

2. Составить таблицу основных и широко используемых утилит для исследования вредоносного кода:

Название утилиты	Тип анализа	Ключевые особенности и функции	Разработчик	Первый выпуск	Языки интерфейса

Контрольные вопросы:

1. Что такое дизассемблер?
2. Как происходит защита программ от дизассемблирования?
3. Как происходит защита программ от отладки?
4. Какие виды отладчиков вы знаете?
5. Что такое декомпилятор?
6. Какие он функции выполняет?