**Лекция 7. Исследование безопасности приложений. Обзор инструментов.**

Инструментов для обратной разработки – огромное количество, также они делятся на различные категории:

1. ПО для первичного анализа бинарного файла.
2. Дизассемблеры.
3. Отладчики.
4. Фаззеры.
5. Прочее специализированное ПО.

**Первая категория** включает в себя достаточно большое количество различных инструментов начиная от ПО находящего строки, заканчивая шестнадцатеричные редакторы и анализаторами упаковки файла. Основным инструментов в данной категории можно считать шестнадцатеричный редактор – Hex-redactor. Существует огромное количество различных редакторов, от самых простых, которые показывают лишь байты до редакторов с шаблонами популярных форматов и поддерживающих расширения. В теме данного курса будет рассмотрен простейший редактор HexDen (HxD).



Рисунок 1 - Главное окно редактора HexDen.

Шестнадцатеричный редактор используется для анализа внутреннего устройства файла, позволяет удобно исследовать заголовок файла. Если для какого-либо не текстового файла отсутствует приложение, способное работать с данным форматом файла, то анализ файла как раз и осуществляется в шестнадцатеричном редакторе.

На Рис. 1 изображён шестнадцатеричный дамп исполняемого файла под Windows, в самом начале файла мы видим заголовок формата исполняемого файла под Windows. Далее идут системные заголовки, которые определяют правила для работы загрузчика.

В шестнадцатеричном дампе файла можно найти все данные, которые могут содержаться в данном файле. Например, для исполняемого файла под Windows в шестнадцатеричном дампе можно найти ссылки на библиотеки динамической линковки.



Рисунок 2 - Библиотека динамической линковки в шестнадцатеричном дампе.

Обычно шестнадцатеричные редакторы используются для первичного анализа исполняемого файла (в контексте тематики исследования приложений).

 **Вторая категория** подразумевает под собой дизассемблеры. Дизассемблер представляет собой программу, позволяющую преобразовывать машинный код в код на языке ассемблера. Данное определение довольно упрощено, однако достаточно.

Рынок дизассемблеров не особо пестрит различными продуктами. Это связано с множеством различных причин, одна из которых – отсутствие необходимости в обратной разработке. Если попытаться разбить дизассемблеры по категориям, то целесообразнее будет разбить их на 2 класса: интерактивные и не интерактивные. К интерактивным относятся дизассемблеры, позволяющие взаимодействовать с собой для улучшения вывода и упрощения анализа. Такими дизассемблерами являются **radare2** и **IDA PRO.**

Не интерактивные отображают результат своей работы и не позволяют удобно взаимодействовать с собой для улучшения вывода – SoftIce, ObjDump, и прочие старые дизассемблеры.

Основным дизассемблером считается IDA PRO. Данный инструмент имеет огромные возможности для исследователя. Данным дизассемблером поддерживается огромное количество архитектур, что позволяет успешно проводить исследования различного встраиваемого ПО. Также IDA PRO имеет хорошее подспорье для расширения, так как имеет свой встроенный скриптовый язык - **idc**, а также удобное API на Python, которое позволяет использовать мощь скриптового языка. В данном курсе основным дизассемблером будет считаться – IDA PRO.

Есть довольно тонкий момент, связанный с IDA PRO. Дело в том, что полная версия имеет довольно приличную стоимость. Бесплатные версии довольно узкофункциональные, и поддерживают лишь 2 архитектуры x86, x86-64. Однако большинство начинающих исследователей, не имеющих работу и официальной лицензии для IDA, используют пиратские версии. В сети интернет, множество различных полных версий IDA PRO, от 6.6 до 7.0. Также в лицензионных версиях IDA часто можно встретить совместно идущее дополнение – декомпилятор HexRays. Данный декомпилятор упрощает анализ приложений с архитектурой x86, x86-64.

 IDA Pro является довольно сложным инструментов для полного освоения, из-за большого количества доступных функций и неоднозначности работы некоторых функций. Однако базовые навыки работы можно получить после нескольких часов работы.

**Работа с IDA Pro. Основы.**

Для загрузки файла (под файлом подразумевается исполняемый файл или, например, какая-нибудь прошивка) в IDA Pro, можно использовать 2 способа. Первый заключается в том, что после запуска IDA Pro, можно перенести необходимый файл в окно с IDA. Второй способ – в панели управления IDA найти вкладку «File», в выпадающем меню найти опцию «Open», далее найти необходимый файл и открыть его.

Загрузка файла в IDA происходит по-разному, в зависимости от файла. Если файл содержит мало кода и данных, то загрузка и автоматический анализ пройдёт почти не заметно. Также нужно отметить, что для известных форматов исполняемых файлов, автоматический анализ будет проведён крайне успешно, так-как IDA сможет корректно разобрать все заголовки и выгрузить все данные верно. Что это даёт? Все функции и ссылки будут успешно распознаны и восстановлены. То, есть можно достоверно сказать, что данная функция использует данную строку из сегмента данных.

Однако, иногда можно встретить файлы, защищённые упаковщиками, различными технологиями и обсуфкаторами. В таком случае результат автоматического анализа IDA будет довольно бесполезным. Без помощи пользователя, упакованные файлы не будут верно распознаны.

После автоматического анализа в окне IDA Pro будет отображён граф порядка вызовов процедур (см. Рис. 3), для перехода в режим листинга ассемблерного кода необходимо нажать «Space».



Рисунок 3 - Граф вызова функций.

Одно из основных удобств и преимуществ IDA является графическое отображения ветвлений в коде функций, что позволяет разбивать функцию на логические блоки и удобнее наблюдать за условиями перехода в тот или иной блок кода.



Также, стоит отметить и отображение информации об использовании стека исследуемой функцией. Перед первой командой функции чаще всего можно заметить, указание на количество используемых локальных переменных, размещаемых на стеке и количество аргументов, а также увидеть смещения, по которым локальные переменные находятся на стеке.



 Из-за многофункциональности IDA существует большое количество так называемых «горячих клавиш», позволяющий быстро выполнять определённые действия. Данный список достаточно обширен и учить его полностью целесообразно только в том случае, если вы собираетесь заниматься обратной разработкой довольно серьёзно. Однако некоторые из данных сочетаний всё же нужно знать.

 «Space(пробел)» - переход от графового представления к текстовому и обратно.

 «Shitf+f12» - отображение секции строк.

 «a» - конвертирование данных, на которых находится курсор в строку (удобно для преобразования шестнадцатеричных констант, являющихся кодами таблицы ASCII в символьный вид)

«n» - переименование (функции, метки, переменной итд)

«Alt+t» - поиск по отображённому в текущем окне тексту.

«x» - поиск перекрёстных ссылок на выделенный объект (выделение происходит путём одинарного нажатия ЛКМ на интересующий объект, который может являться функцией, строкой, меткой итд), одна из наиболее часто используемых клавиш при анализе кода, т.к. позволяет понять где была вызвана функция, или где используется строка.

 Все из приведённых сочетаний можно найти в верхнем меню IDA, однако, их поиск и нажатие мышкой будет занимать достаточно много времени, по сравнению с нажатием на клавиатуре.

**Теперь перейдём к отладчикам.**

 Отладчик позволяет вам анализировать программу динамически. То есть вы можете пошагово (инструкция за инструкцией) выполнить программу и пронаблюдать состояние и поведение программы на каждом шагу. Стоит отметить, что данный вид анализа применим, только если у вас есть платформа, на которой можно произвести запуск исполняемого файла и отладчик для этой платформы, иначе отладка не будет возможной. Как уже говорилось, исполняемые файлы для Linux не могут быть запущены под Windows и наоборот, без специальных вспомогательных программ. Такая же ситуация и с отладкой, есть отладчики под Windows, а есть под Linux.

 Также, стоит отметить, что речь идёт не о встроенных в IDE (интерактивная среда разработки) отладчиках, которые чаще всего позволяют отлаживать исходный код, а о отладчиках, работающих с финальными исполняемыми файлами, и отлаживающие по сути двоичный код, в ассемблерном представлении.

 Существует множество различных отладчиков как для Windows, так и для Linux, однако в теме данного курса будут рассмотрены наиболее популярные и видные представители – gdb (Linux), Immunity debugger (Windows).

 Работа с отладчиком будет подробна рассмотрена в одной из следующих лекций, посвящённых поиск уязвимостей, их отладке и написанию эксплойтов.