Reverse Engineering — Lurkmore

В эту статью нужно добавить как можно больше не софтваре реверс инжиниринга, ибо ваистенну.

Также сюда можно добавить интересные факты, картинки и прочие кошерные вещи.

«I wanna dig inside to find a little bit of me »

— *Slipknot*

Reverse Engineering (ревёрсинг, обратная разработка) — процесс ~~низженья~~ восстановления исходников из конечного продукта инженерной и/или научной деятельности, интуитивно конструируя внутреннюю механику по принципу *«а какие процессы должны вызвать такое вот внешнее поведение этого продукта?»*. Ориентируясь на нюх, так сказать. Иногда приводит к написанию собственного сорца али моделированию/разводке/пайке железячного аналога. Много их — ибо ваистену…

В общих чертах

Возможно, все мы в детстве сломали немало игрушек пытаясь понять «как оно там устроено». С возрастом это проходит. Но не у всех.

Bryan Adams - Inside Out
Брайан Адамс заглядывает внутрь.

Учёные пытаются дизассемблировать Вселенную, чтобы восстановить исходники Б-га, который явно наложил несколько пакеров и обфускаторов. Военные первым делом подбирают оружие на поле боя — чтобы посмотреть, как устроена эта штурмовая винтовка Stg-44 и можно ли из неё сделать Калаш, а уже потом закапывают трупы. Китайские дизайнеры успешно реверсируют дольче-габбано, чтобы произвести его больше, дешевле и хреновее. Ну а некоторые реверсируют программы, чтобы исправить в них некоторые ошибки (одна из самых распространённых — жадность разработчика) или просто посмотреть, как она устроена, что умеет, и сделать свою, с маджонгом и гейшами. Они-то и являются илитой IT.

Алсо, чуть менее, чем все советские микросхемы представляли собой аналоги забугорных, скопированные с готовых продуктов. В связи с этим в этой стране даже создан закон о правовой охране топологий интегральных микросхем, чтоб у нас в свою очередь не тырили.

Software Реверсинг

Реверсинг ПО — восстановление принципов/идей/алгоритмов работы программы для исследования и/или создания аналогичного ПО. Часто применяется для:

- Анализа вирусов/троянов/червей и прочего дерьма с целью создания средств защиты.
- Поиска дырок в закрытом софте с целью создания вирусов/троянов/червей/сплойтов и прочего дерьма.
- Создания описаний для форматов данных/протоколов, используемых в программах и прочем дерьме. Пример с преподской тестирующей программой относится сюда.
- Анализа работы закрытых драйверов и прочего дерьма для создания открытых линупсовых.
- Изготовление пиратских серверов серверных игр вроде WoW и допилка их рубанком до сходства с официальным, ясное дело, дерьмом.

Не следует путать взлом ПО с его реверсингом: для взлома достаточно разобрать принцип работы лицензионной проверки, а реверсинг — это полный разбор программы по кирпичикам и раскладывание своего мозга по уровню кошерности.

Для успешного кодопиздинга требуется знать ассемблер, иметь общее представление о криптографии, знать мунспик и теорию вероятностей — не быдлокодер ли твой предшественник? Ну и матан — например, теорию графов для анализа ветвлений. Но можно и не знать, главное — уметь быстро найти, что почитать, и резво разобраться.

Hardware Реверсинг

Реверсинг железа — всяких блоков управления стиральными машинами, субару импрезами, контрольно-кассовыми машинами. Обычно применяется для:

- Скручивания одометров на прульном дерьме, патчей блоков управления подушками безопасности (чтобы быдло думало, что


Реверс инжиниринг дивайса

его 15-летнее БМВ ещё не побывало в аварии).
- Патчей для тепло-, водо-, электросчётчиков, контрольно-кассовых машин и прочего дерьма.
- Патчей блоков управления моторами с целью добытия большего количества ~~дерьма~~ лошадиных сил из двигателя и на порядок раннего экстерминатуса мотора. Алсо для снятия ограничителя скорости.
- Снятие каких-либо ограничений, заложенных производителем в девайс (как в модемах US robotics и тому подобном дерьме).
- Восстановление документации, если оная утеряна либо уничтожена из-за древности, но очень надо получить ещё сотню таких же устройств.
- Поддержка жизненного цикла дорогих проприетарных девайсов, купленных планктоном, сразу же распилившим средства на их техподдержку на что-нибудь более уютненькое. Тру «форд фокус», например.

Любой разработчик и производитель имеет свою лабораторию анализа отказов для *настоящего* чип-реверсинга. Вот, например, фотоотчёт о лаборатории Atmel.

В той стране от безысходности вообще было принято решение об отказе от разработки собственных чипов. [*ЩИТО?*] Копипиздинг топологии ИМС был поставлен на широкую ногу.

Инструменты

Статический анализ скомпилированных программ

В данном разделе представлены инструменты для "декомпиляции", в смысле превращения машинного кода в понимание, что же он такое делает. При этом сюда же включены и отладчики, потому что главная функция отладчика в контексте RE - это в первую очередь оптимизация навигации по коду.

Теперь о классификации инструментов. Есть 2 сорта инструментов - универсальные и узкоспециализированные. Узкоспециализированные оптимизированны для конкретной архитектуры. Они имеют меньшее потребление памяти и работают шустрее. Универсальные оптимизированны для code reuse. Они используют промежуточное представление и толстый middleware, в них может быть легче (если знаком с middleware) или тяжелее добавлять новые фичи, но когда это сделано, пользу получают все поддерживаемые архитектуры.

Также можно классифицировать на 2 класса по возможности вклиниться между стадиями.

1. Конвейер. Парсер формата ("загрузчик") парсит заголовки и извлекает метаинформацию, метаинформация используется для правильного парсинга машинного кода в представление в памяти, специфичное для архитектуры ("дизассемблер"), "лифтер" преобразует это представление в нейтральное промежуточное представление, которое оптимизируется, после преобразуется "декомпилятором" в нужный язык программирования. Конвейер можно прервать между стадиями и поковыряться в промежуточных результатах. Обычно универсальные.

2. Монолит. Тут всё вышеперечисленное слито вместе, а нейтральное представление упразднено. Не между всеми стадиями можно вклиниться. Монолиты, обычно классифицируют по последней стадии. Обычно узкоспециализированные.

Инструменты можно ещё классифицировать как интерактивные и пассивные.

Интерактивные имеют пользовательский интерфейс, позволяющий человеку управлять его работой, подключая свою естественную нейронную сеть в качестве модуля. Пассивные просто получают на вход одно и выдают другое.

Полные конвейеры / комбайны

IDA

Как следует из названия, интерактивный. Основной декомпилятор и дизассемблер на сегодняшний день, скомбинированный с отладчиком. Обладает множеством функций типа поддержки скриптов, графического режима или встроенных плагинов вплоть до эмулятора x86. Не тупой интерфейс — множество интуитивно понятных «горячих» клавиш и т. п. Проекту уже лет 15, и он единственный из дизассемблеров живёт активной жизнью и поныне: у других в новых версиях фичи почти не родятся. До сих пор таскает с собой DOS-GUI версию для олдфагов. Также известен нефиговой ценовой и KYC политикой.

Пишет его татарский расовый мудак по имени Ильфак Гильфанов. Детали истории создания дизассемблера покрыты мраком, но известно, что Ильфаку, как и Биллу Гейтсу с DOS, добрые люди помогли написать первые версии. И задумывался он как инструмент для восстановления прошивок HDD.

Дальше, непонятно как и почему, люди разошлись, и остался он один. Потом Ильфак всплывает с версией 3.x уже от лица фирмы Datarescue, которая купила проект. Начиная с версии 4.0, у него появляется GUI, написанный на BCB! С этого времени продукт начинает приобретать массовую известность в узких кругах по двум причинам:

- MUST HAVE
- «Ильфак — пидорас»

Остановимся на втором пункте подробнее.

Ильфак, будучи на данном рынке монополистом, давно забил на пользователей хуй. Потому что берут и так. Любой юзер IDA, которому приходилось иметь дело с поддержкой, получал кучу отрицательных эмоций: получив продукт с кучей багов, вы предоставлены себе в их исправлении. Ошибки исправлять никто не спешит — можно легко прождать полгода, даже если баг вопиющий. Продукт практически не тестировался перед релизами и не тестится до сих пор. Юзеры IDA вынуждены хранить несколько версий, потому что в одной хорошо работает одна фича, в другой — другая, в третьей — третья.

А среди «неофициальных» юзеров Ильфак нелюбим за то, что без спроса берёт идеи. За счёт того, что IDA поставляется с SDK, юзеры могут делать свои плагины к IDA, и таки делают немало. Плагины пишутся как для того, чтобы исправить баги оригинальной функциональности, так и ради добавления своих фич для решения конкретных задач. Уровень альтруизма среди «неофициальных» юзеров высок, и исходники плагинов выкладываются на всеобщее обозрение. Так вот, примерно через полгода после появления полезных исходников это становится фичей IDA. У Ильфака на этот счёт железное алиби: «То, что тут придумали — это не новая идея, это меня пользователи попросили ещё год назад».

Вначале народная ненависть выливалась в троллинг на форуме официального русского сайта. Но потом всё быстро утихло, ибо Ильфак анально отгородил форум официальных юзеров, а с неофициальными общаться перестал. Посему примерно раз в год то китайцы, то албанцы, то русские (был интересный случай, когда народ скидывался на руборде, чтобы купить одну копию и поделиться), то немцы тырят с тренингов, кардят, хачат сервера с IDA и коммуниздят очередную версию. Чем каждый раз нереально доставляют целевой аудитории IDA. А официальные юзеры потом смотрят в закрытые форумы поддержки и скидывают цитаты баттхёрта Ильфака в публику, повышая градус веселья.

Потом Ильфак свалил с Datarescue и стал делать IDA от лица своей конторы Hex-Rays. Ходят слухи, что он нанял наконец программеров с прямыми руками, которые наконец начали чистить его говнокод. Поэтому версия 5.5 была воспринята с одобрением, и некоторые неофициальные пользователи даже начали помышлять о приобретении. Но эйфория продлилась недолго, баги снова были найдены, и всё вернулось на круги своя.

Ознакомиться с историей спионеривания можно тут. В июле 2011 года у антивирусной компании была упёрта полная версия декомпилятора версии 6.1. ТЕН DRAMA!

NSA Ghidra

Универсальный интерактивный фреймворк от АНБ США, написанный на Java и использует Pcode для парсинга семантики инструкций, который скармливается декомпилятору и анализатору потока управления (поддерживаются типы перехода call, branch, return, terminated return, aka exit (для крашей, main и т.д.)), а также Sleigh для дизасма. Содержит в себе гораздо более качественный декомпилятор для цпушных бинарных языков (вроде x86, PPC, ARM, MIPS и прочих с указателями), но фейлится для управляемого байткода (JVM, Dalvik и прочее ООП), т.к. не пережёвывает синтаксический сахар, исключения и прочее. Содержит эмулятор, который при должном использовании становится убийцей отладчиков, поскольку те легко детектируются, а в эмулятор можно всунуть что угодно без патчинга, вплоть до перезаписи функций проверок наличия отладки и т.п. Плюс можно эмулировать для кросс-тестирования декомпиленного кода и пкода, т.к. декомпилятор может ВНЕЗАПНО допускать ошибки, приводящие к неопределённому поведению, но это так, на случай, если кому-то нужно получить функционально эквивалентный проект на C (нет, никакого ЦПП, только в C или, если загрузчик для языка так настроен (пока что только JVM, Dalvik), в Java, ибо такой сахар, как шаблоны, почти всё ООП, начиная с классов и заканчивая RAII, просто не выживает компиляцию).

Retargetable Decompiler, aka retdec

Универсальный декомпилятор-конвейер от Avast. Пассивный, что означает, что потвикать результаты декомпиляции модулем "естественный интеллект общего назначения" ты не сможешь. Вернее сможешь, закинув код в Студию/Creator, но декомпилятор после этого тебе помогать откажется. Тормознутый и ужасно жрёт оперативу на мизерных бинарях на несколько мегабайт, после чего падает по OOM. Частично обхозится отключением части проходов. Очень не любит виртуальные машины - на них обрезание проходов скорее всего не поможет. Как и большинство декомпиляторов, использует под капотом LLVM. Промежуточные этапы дампятся в файлы, которые потом можно скормить другим инструментам, использующим LLVM, или наоборот, выхлоп других инструментов скормить retdec.

GUI

Cutter

Интерактивный GUI к другим инструментам, в основном для Radare2, но туда ещё подключили retdec и hex-rays как плагины.

Дизассемблеры

Дизассемблер — это утилита, преобразующая бинарный код в ассемблерный листинг. Обычно монолиты. В простейших случаях этого вполне достаточно, и остаётся немного покурить и найти то, что нужно. В худших — это каша, и без скриптов и/или отладчика не обойтись.

Существует 3 вида дизассемблеров.

1. Дизассемблирует всё подряд между двумя адресами. Абсолютно бесполезны для реверсинга. 2. Дизассемблирует инструкции, переходя по переходам. При отсутствии контрмер, которые сильно зависят от платформы, умеют отличать код от данных и брать верные смещения. Полезны для реверсера. 3. Brute-force disassembly. То же самое, что и 2, но не ставятся в тупик контрмерами. Имеют смысл только для некоторых платформ.

Hiew (*Hacker's viewer*). Небольшая утилита, для нормальных людей бесполезная, юзают лишь надрочившиеся спецы, иногда — продвинутые пользователи (как шестнадцатеричный редактор). Давно существует куча GUI-шных тулзов для тех же целей, но по славной традиции тулзу продолжают использовать…

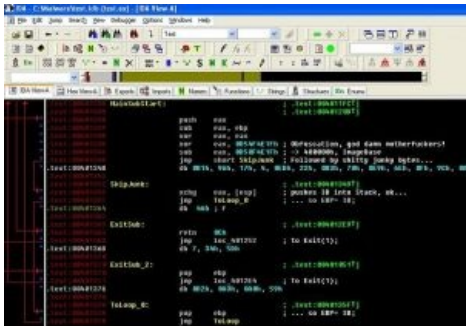
Sourcer от компании V Communications. Стародавний мощный дизассемблер времен DOS и разных версий Windows. Позволял из .COM или .EXE файлов получать непосредственно .ASM файл, причём для различных компиляторов ассемблера, таких как TASM или MASM. Если было задано достаточное количество проходов при дизасемблировании, то с высокой степенью вероятности можно было при компилировании только что созданного .ASM файла получить исходную исполнимую программу байт-в-байт. Доставлял хорошими комментариями в листинге, скоростью работы, поддержкой разных типов процессора и, конечно, олдскульным *сценовским* интерфейсом. Сейчас не актуален, хотя некоторые анонимусы продолжают держать его в своём инструментарии, например, для дизассемблирования MBR жёсткого диска, Boot-сектора, специальных утилит, загружаемых с дискеты без какой-либо ОС (например утилиты перепрошивки BIOS или утилиты от производителей HDD), и прочего. То есть для низкосистемных нужд. Последней явилась на свет версия v8.01 от 2001 года. Sourcer — помним, любим, скорбим!

Отладчики

Отладчик — программа для контролируемого исполнения программы или в общем случае любого произвольного куска кода, вплоть до инжектированной в другой процесс DLL. Продвинутый отладчик умеет использовать аппаратные средства отладки, ну и, конечно, дизассемблировать.

x64dbg

Основной отладчик для Windows на данный момент. Скриптуется, плагинится, но имеет отвратительно управление, к которому надо привыкать.

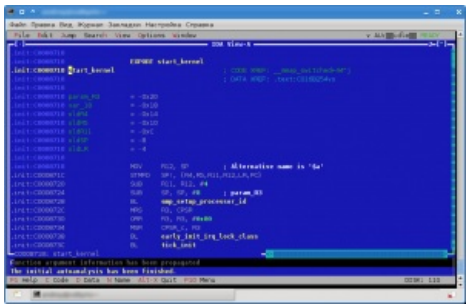


Матрица поимела тебя. Анализ кода в IDA.

SoftIce

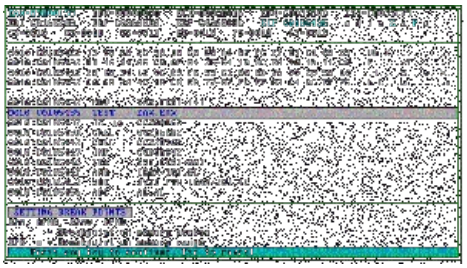
Дебаггер, работающий на нулевом уровне, на котором можно всё. Появляется по нажатию CTRL+D, и всё остальное — программы, драйвера и даже некоторые прерывания перестают работать. Переставало идти даже «время», и задебажившийся индивид мог потом реально фалломорфировать, осознав сколько прошло времени в «реальном мире». Заодно обрубалась модемная связь, вызывая лютый багет от забытого брейкпоинта при ночном дозвоне по дайлапу. Можно трассировать что угодно: драйвер HDD, USB-антенны и т. п. А ещё там работает мышь и можно делать скрины.

SoftIce вышел на рынок как отладчик уровня ядра совершенно неожиданно для самих создателей этого ядра — MS. Позиционировался как тулза «для нахождения багов в софте», но последние весьма разнообразны: сюда можно отнести MS Office, требующий ключ активации. Немаловажное преимущество: он умеет считывать при загрузке инфу о библиотеке, в процессе отладки показывая имена, например, Kernel.FindClose+105h.



IDA в линуксе или линукс в IDA?

Нынче похож на свежий навоз: его уже n лет никто не обновляет. И не будет — в связи с выпилом конторы разработчиков: Numega невозбранно была проебана за 30 сребреников мудацкой Compuware. В итоге всё закончилось тотальным выпилом как нумеговских проектов, так и собственно офиса бывшей Нумеги. А остатки Девпартнера отдали какому-то ебучему реселлеру Microfocus, потому что Compuware после тотального экстерминатуса Нумеги забил на все эти ваши девпартнеры-девшмартеры болт.



Прошлый desktop-тред в бамплимите, начинаем новый...

Radare2

Целый фреймворк для реверс-инжиниринга. Изначально hex-редактор включает в себя дизассемблер, декомпилятор... Лучше сами смотрите. Кроссплатформенный, с несколькими интерфейсами на выбор, а исходный код его открыт. Ня!

Immunity Debugger

Очень популярный отладчик с поддержкой автоматизации на питоне. Но дохлый.

Остальные

WinDbg — как бы быдлокодерам, испытывающим от слова «Майкрософт» жжение в области ануса, ни было обидно, это самый популярный kernel-mode отладчик со времён смерти SoftIce. Да и в user-mode Винбэг обеспечивает лютый вин, кладя болт на эти ваши Debugging API и позволяя, например, заглянуть в native code для процесса, в который уже впилен отладчик для managed code.

Syser — китайский дебагер уровня ядра с GUI. Пришёл на смену SoftIce, загнулся в 2011, проиграв войну новым 64-битным ядрам.

OllyDbg — отладчик третьего уровня, гуишный, симпатяффный, функционала выше крыши, применим для 95 % программ, а пропатчен ручками и для всех 98%. Но из-за того, что r3 среди профессионалов считатся не труъ-тулзой, подходящей скорее для скрипткидисов, чем для разбора чего-то реально сложного, автор положил на него хуй, и теперь отладчик разрабатывают новые умельцы, пафосно назвав OllyDbg 2.0. [Отечественная разработка http://www.ollydbg.de], хоть и менее известна, чем SoftIce, вполне кошерна и к тому же бесплатна. Релизы выглядят хреново, но парни явно набираются опыта и идут в правильном русле.

Лифтеры

Как уже сказано, переводят ("поднимают", "отрывают") бинарники в промежуточное абстрактное представление. Промышленный фактический стандарт промежуточного представления - это биткод LLVM, файлы имеют расширение bc. LLVM - это фреймворк для построения компиляторов, содержащий в себе код для оптимизации, и всеми нами любимый clang построен на нём, как и большая часть компиляторов шейдеров для видеокарт, входящих в состав драйверов.

- retdec - используйте лучше его для лифтинга.

ещё следует отметить

- McSema - без IDA Pro бесполезен, так как полагается на него для дизассемблирования
- llvm-mctoll

Остальные все сдохли.

Декомпиляторы

Машинного кода

Общего назначения

- См. раздел "комбайны". Остальное - лишь пародии на них.
- Snowman
- здесь
- вот тут.

Специфичные для языков

Некоторые компиляторы для некоторых языков генерят весьма специфический субоптимальный машинный код, который имеет определённую структуру и привязан к некоторым runtime-библиотекам. Глубоко понимая структуру этого кода можно написать декомпилятор, восстанавливающий исходный код на исходном языке относительно близко к оригиналу. При этом декомпиляторы общего назначения захлебнутся это оптимизировать.

К сожалению, свободных тут почти нет, одна дорогущая проприетарь:

- VBDecompiler
- VBReformer

LLVM bitcode

Декомпилируют из поднятого лифтером представления.

- retdec
- llvm2c - работает кое-как, но на гораздо меньшем количестве бинарников, чем retdec.

Языки с байт-кодом

Современные языки программирования типа Java, C# и прочего .NET'а вовсе и не требуют для своего взлома применения дизассемблеров и отладчиков. Весь исходный код легко можно получить в считанные секунды, а если кодер не применил даже обфускацию, то полученные исходники от авторских будут отличаться разве что отсутствием комментариев в первых.

.Net

.NET Reflector — хороший инструмент для декомпиляции программ на C# и некоторых других языках. Авторы постоянно дорабатывают своё детище, вскорости обещают декомпилировать и другие основные языки. Изначально был бесплатен, но после появления широкой аудитории стал платным, впрочем, Reflector использовали для взлома платной версии.

ILSpy — свободный аналог рефлектора от разработчиков SharpDevelop.

dotPeek — детище небезызвестной JetBrains, к моменту выхода из альфы ставшее вполне себе годным инструментом, способным конкурировать с лидерами отрасли. Декомпилирует обфусцированную лапшичку, перед которой пасует даже знаменитый и всеми любимый .NET Reflector. Ах да, программа поставляется абсолютно бесплатно.

JustDecompile - декомпилятор от конкурентов из Telerik

JVM

- Krakatau - успешно декомпилирует то, на чём все другие декомпиляторы просто падают с ошибками, а именно программы, написанные не на Java, но тоже скомпилированные в байт-код JVM. Правда декомпилированный код содержит излишнюю информацию, которую приходится чистить руками.
- CFR - очень неплохой декомпилятор.
- luyten
- JD
- JavaDec, JavaByte, Cavaj, тысячи их... Инструментарий тут гораздо побогаче, чем для .net.

Python

Эталонная реализация питона, cpython, компилирует код в байт-код, и исполняет его. Возможно распространять не исходники, а байт-код.

К счастью есть uncompyle6.

Деобфускаторы

Нередко код бывает обфусцирован. Зачастую помогают деобфускаторы. Для интерпретируемых языков (за исключением perl) обычно не требуются или легко самопишутся.

Для JavaScript — JSBeautifier.

Для шарпа один из лучших — de4dot.

Парсеры "исполняемых" форматов

Бинари для современных ОС содержат не только машинный код, но и информацию о том, как его необходимо правильно загружать в память. В никсах используется ELF, в винде - PE, в MacOS/gayOS - MachO.

Большинство из инструментов - библиотеки. standalone инструменты нужны больше для внешнего осмотра, все декомпиляторы/дизассемблеры/лифтеры либо пользуются чужими либами, либо имеют свою либу внутри себя.

Универсальные

- LIEF - в большинстве случаев тебе нужен именно он. Либа с широким функционалом. Жаль только, что с нуля делать бинарники не умеет, только редактировать существующие.

PE

- PE Tree
- **PETools** — тулза некоего NeoX.
- LordPE - (умеет фиксить SizeOfImage в пебе).

ELF

- libelf - умеер и в парсинг, и в конструирование. LGPL v3
- pyelftools и статьи (1 2 3 4) от Илая Бендерски

Динамическая инструментация

Инструменты из этой группы позволяют автоматически навесить на чужой код свои обработчики. Полезно, например, для логгирования, некоторые инструменты из той группы сделаны поверх инструментов из этой группы.

- DynamoRIO
- frida
- systemtap - для ядра Linux.

Символьное выполнение

Ты накладываешь условие на желаемое состояние программы, система вычисляет входные данные, необходимые, чтобы его достичь. Разумеется, без каких либо гарантий, задача вообще неразрешима.

- angr

Патч-анализ и дифферы для control flow graph

Вендор выпустил патч, а что в нём - не сказал. Не порядок.

- DarunGrim Проприетарь.
- Polypyus

Логгирование поведения

Иногда нам нужно создать свою программу, взаимодействующую с определённой библиотекой с известным API, интерфейсом ядра ОС, устройством или с сетевым сервисом, обычно с сайтом. Тут незаменимы следующие инструменты:

Логгирование библиотечных вызовов

- ltrace для никсов
- drltrace для Windows

Логгирование общения по сети

- Wireshark
- mitmproxy
- Telerik Fiddler ("бесплатная" проприетарь, но на шарпе, см. пункт "Декомпиляторы")

Логгирование USB

- USBPcap (Windows)
- usbmon (Linux)

Логгирование API OS

- strace
- Microsoft Sysinternals Process Monitor

Реверсинг файловых форматов

- Kaitai Struct - язык описания. Позволяет быстро и интуитивно понятно прототипировать парсеры, в почти реальном времени визуализировать результат и повторять до полного понимания структуры формата
- Для сложных случаев:
  - Hobbits - продвинутая визуализация бинарей через цифровую обработку сигнала.
  - Veles - просто красивые картинки, построенные по всяким эвристикам, которые типа помогают видеть структуру в бинари, а на деле не помогают никак. По крайней мере мне не зашло.

Аппаратный реверсинг

- осциллограф с подключением к компу
- устройства для работы с интерфейсами и сетями: UART, JTAG, SPI, I2C, CAN, Wi-Fi, Bluetooth, LoRa, ZigBee
- glitch-kit

SDR

Позволяют слушать и слать произвольные радиосигналы. Необходимы для реверсинга беспроводных протоколов.

- rtl-sdr - дешёвый USB-донгл
- nexmon - monitor mode + SDR из Wi-Fi чипов в телефонах и Raspberri Pi
- hackrf, bladerf, rad1o - платы на основе FPGA

FPGA

Позволяет делать многое из вышеперечисленного. По сути микросхема, определяемая прошивкой. Обычно используется для быстрой обработки сигнала.

Прочее

RSATool — GUI-программа для прокачивания скила по RSA. Автор — эгоист из ТМГ (илита сцены, ни одного нюка). Для факторизации не годна, лучше юзать msieve.

Архив даташитов и документация

Без этих вещей вы никогда не узнаете, что процессор Nippon-Denso NP648976 на самом деле жалкий клон Motorola 6803 с 16K ROM на борту. А когда узнаете, поймёте, что япошки никогда ничего сами не придумывали, а только умело пиздили.

Вещи, затрудняющие анализ

Для борьбы с реверсингом авторы используют всяческие трюки, чтобы их творения не были доступны анализу. Чем выше жадность и хреновее функционал, тем обычно более параноидальна защита, которая в клинических случаях может даже начать мстить (удалять файлы и т. п.), если почувствует, что к её попке приближается некто с отладчиком. Интересно, что глючащая защита может сработать там, где кряком и не пахло, и тогда простому пользователю пиздецъ.

Продвинутые программы редко содержат паранойю, например, WinRar.

Выпуск эксклюзивно под анально огороженную платформу для демонстративного потребления - платформой не пользуются те, кого могла бы заинтересовать крякнутая прога или отреверсенная программа - касание технических деталей и/или использование софта, за который не заплачено, считается ниже достоинства пользователей данной платформы, как если бы генеральный директор компании с доходом миллион в неделю собственноручно занимался мытьём туалета у себя дома вместо того, чтобы нанять профессионала и заплатить ему, или ездил бы на автомобиле, который ремонтируют не в официальной дилерской мастерской. Любители сами поковыряться и нелюбители понтов просто используют другую платформу. Раз никому не нужно - нет смысла и реверсить.

Быдлокодинг. Как это ни удивительно, но быдлопроги зачастую обладают естественной защитой от анализа. Скажем, проект на Visual Basic, скомпилированный в P-code, читать гораздо труднее, чем продукт полноценного C-программера. Даже после декомпиляции. У профессиональных программистов есть представление, какой примерно должна быть архитектура для решения той или иной задачи. Архитектура строится из известных шаблонов (которые есть даже у тех, кто никогда о них не слышал, great minds think alike). У быдлокодеров понятия обо всём этом нет, поэтому их говнокод делает неожиданные и дикие для профессионала вещи и имеет ненужные зависимости между частями, что портит жизнь как самому быдлокодеру, так и всем остальным, вынужденным в разбираться в его коде. Также сложность представляет индусский стиль программирования — быдлокодер неумело пытается защитить своё творение путём искусственного раздутия кода, хотя тот и без всякой доработки может представлять собой такую HEX, что даже имея оригинальные исходники, хрен поймёшь, как оно работает - чтение такого листинга может вызывать рвоту.

Дзен-кодинг — использование хитросплетений паттернов проектирования и вообще кодогенерации шибко «умным» программистом приводит к созданию не менее отвратительного кода, который что в сорцах хуй проссышь, что в асме.

Пакеры сжимают программы, чтобы они занимали меньше места на диске и распаковывались в памяти. Per se защитой не являются, однако в нераспакованном виде анализировать сжатую программу невозможно. Для известных пакеров лучше всего найти анпакер, который вернёт программу в более-менее исходное состояние. Известные — UPX, ASPack, MEW, PECompact, NSPack.

Обфускация — превращение исходного/промежуточного/конечного (нужное подчеркнуть) исполняемого кода в кашу, которую невозможно читать. Из исходной инструкции обфускатор делает пять (а то и сотню, в случае ~~паранойи~~ пермутации), делающих то же, что и одна, плюс ещё 200, которые вообще ничего не делают, и 28 препятствующих эмулированию или отладке. Применяется как в «хороших» программах, чтобы перенести появление кряка, так и в «плохих», типа полиморфных вирусов, стремящихся оттянуть момент своей детекции — каждая следующая копия будет непохожа на предыдущую. Олсо пытается юзаться в илитной малваре типа рустока и прочей, в расчёте на то, что антивирусные вендоры будут сосать хуйцы. Кстати, есть такой сайтик http://www.ioccc.org, где проводится конкурсы на самую запутанную программу для UNIX. Победители конкурса — истинные шедевры кодописательства и кодозапутывания. Настоятельно рекомендуются к ознакомлению.

Антиотладка — приёмы для борьбы с отладкой. Для мелкософтовского виндувса достаточно `IsDebuggerPresent()`. Например, уныло посмотреть поле BeingDebugged в пебе, или посмотреть значение NtGlobalFlag. Это 2 самых говнометода, илита же предпочитает еблю с rdtsc (многопроцессорные системы сосут хуйцы), ядерные сплойты и парсинг физической памяти. На каждый такой приём существует антиприём, ну и так далее. И обычно это люто, бешено глючит, бсодит, и вообще, нормальные челы не отлаживают всякую хуиту на своих компах, а заставляют давиться этим калом эмуляторы: Virtual Box, VMWare, VirtualPC и т. д.

Краденые куски — часть кода программы выпиливается, а выполнение оригинального куска выносится вовне: в протектор, на удалённый сервер, донгл и прочая. Когда хацкер делает дамп, он получает не всю программу, а лишь её часть. Кошерным примером реализации этого метода являются токены RSA SecurID.

Виртуальные машины — часть кода программы транслируется в байткоды другого (не обязательно реального) процессора, и добавляется транслятор, который эти байткоды хавает и выполняет. Некоторые протекторы, такие как «Темида», превращают один грамм исходного быдлокода в 1-3 десятка кило нового говна.

Протекторы — проги, которые защитят ваше творение, особо не заставляя думать — знай сиди и бросай формочки в ~~де~~билдере. Thinstaller, RCrypt, ASProtect, EXECryptor (привет Relayer), WL/Themida… много их. Могут содержать все вышеперечисленные пункты в невозбранном количестве и в самых невероятных сочетаниях, но обычно содержат баги в реализациях PRNG/метаморфов и прочей хуиты. Кстати, большая часть самых известных протов уже отошла в мир иной-не выдержали конкуренции с пиратами!

Обращения к взломщикам внутри исполняемого файла в виде ASCII-строк - и такое встречается часто! Начиная от *«злобные дяди хакиры, ни ламайте пожалуйста маю программку, нет денег на пиво»*® до всяких «yates stilL Here kinda Ooooh» и целых анекдотов про улитку в серьезных защитах. Действуют на хакеров-ломакеров сильнее, чем все вышеперечисленные приемы, заставляя последних давиться от хохота, тем самым оттягивая дату неминуемого взлома.

Немного о личных качествах самих реверсеров

> Вечер. Автобус. Едет. На первом сиденье сидят двое. У них коэф. интеллекта IQ=180:
>
> — Я вчера Гамлета в оригинале читал… Такое эстетическое наслаждение… На другом сиденье сидят ещё двое. У них IQ=140: — Я вчера посмотрел «Андалузского пса» и нашёл коррелят с ранними картинами Пикассо… На другом сиденье сидят двое. У них IQ=100: — Мы с другом час назад посмотрели «От заката до рассвета». Как там тёлка говорит тёлке, что… На другом сиденье сидят двое других. У них IQ=80: — Слышь, брат! Помнишь, какое пойло мы в мерсе

> пили, когда ноги какой-то б..яди торчали в окне… А на задней площадке стоят двое с IQ=40: — Ну вот, вскрываю прогу твоим дебаггером…

Анекдоты — анекдотами, но реверсивщики делают много хорошего. Помимо упомянутого участия в разработке всевозможных антивирусов, именно один из них создал неофициальный кодовый патч (UCP) для Master of Orion образца 2016 г, позволивший модить игруху сверх дозволенного разработчиками. Продукт продолжает расширяться и дорабатываться. Самое забавное, что даже по строгим заокеанским законам собственник не осудил кулибина, но и не заплатил за старания, а вместо этого тупо скомуниздил несколько решений в официальный патч. Вот тебе, бабушка, и авторские права!

Ресурсы

- WASM.ru — Windows Assembler Site, ориентирован в том числе на «Отладка и дизассемблирование программ» и «Защита от предыдущего пункта». Отличная подборка статей на разные околореверсные тематики, набор исходников и утилит (впрочем, не сильно приватных), а также форум, где можно получить или поделиться знаниями в области ассемблера. В Германии забанен ~~кровавой гэбней~~ TrendMicro. В настоящее время ушел в вечность. Появились псевдоподия основанные на слитых базах форума, короч внимания не достойны.
- exelab.ru — более узкоспециализированный форум.
- exetools — некогда был неплохой сайт, где тусило немало ИЛ. Сейчас анально огородились, без инвайтов не попасть, на ftp не зайти.
- woodmann/fravia — с чего начинается реверсинг, с картинки в твоем букваре… Самый натуральный букварь, на котором выросло поколение реверсеров конца 90-х.
- OpenRCE, r0 Crew — ещё комьюнити реверсеров, где местами можно найти интересные тулзы, скрипты и плагины.
- Еще один учебник для начинающих на русском
- w:International Journal of Proof-of-Concept — на инглише. Познавательный журнал.

См. также

- Умение разбираться в чужом коде
- ReactOS
- Матан

Китай

Adidas Brick Game Chery Made in China QNet Reverse Engineering TikTok Yao Ming Арбуэ Ашанбайк Баночка Боевые искусства Брюс Ли Будда Гаутама Го Даосизм Дзен Иероглиф Китай Китайские ватты Китайские пингвины Китайские чиновники Китайский айфон Китайский шрифт Коммуняки Мао Цзэдун Мозг обезьяны Неизвестный бунтарь Нонейм Очки надо? Пандемия Пейджер Пиратские игры девяностых Пол Пот Последнее китайское предупреждение Сверхдержава Силумин Суп из младенцев Тамагочи Тоталитарное искусство Узкоглазый Уханьский коронавирус Фейк Фэн-шуй Хуй Чай Шри Япутра Эльдорадо

Языки программирования

++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер Языки программирования

w:Обратная разработка uk.w:Зворотна розробка en.w:Reverse Engineering