Экспериментальная оценка сложности обратной разработки сервера TCP/IP с применением нейронных сетей

Зуб А.В., Кейно П.П.

***Аннотация* — В статье рассматривается задача количественной оценки сложности обратной разработки бинарного TCP/IP-сервера при отсутствии исходного кода и формальных спецификаций протокола. Предложена формализованная модель процесса реверс-инжиниринга, основанная на представлении анализируемого программного обеспечения в виде ориентированного графа вызовов и совокупности функциональных компонент обработки сетевых сообщений. Введён интегральный показатель сложности, агрегирующий временные, структурные и семантические параметры анализа.**

Проведено экспериментальное исследование влияния ассистивного инструмента, основанного на нейросетевой классификации функций, на трудоёмкость анализа бинарного кода. Показано, что предварительная структуризация пространства поиска и приоритизация функций для детального исследования позволяют снизить суммарное время анализа примерно на 30 % при сохранении сопоставимого уровня полноты восстановления протокольной логики.

Научная новизна работы заключается в формализации сложности процесса обратной разработки сетевого сервера, введении интегрального показателя трудоёмкости анализа и экспериментальной оценке влияния нейросетевой поддержки на снижение структурной и семантической неопределённости при исследовании бинарных программ.

***Ключевые слова* — бинарный анализ, граф вызовов, нейронные сети, обратная разработка, оценка сложности, системный анализ, TCP/IP-сервер.**

I. Введение

Рост числа закрытых сетевых сервисов и широкое распространение проприетарных протоколов обмена данными обусловливают необходимость анализа программных систем при отсутствии исходного кода и технической документации. Обратная разработка TCP/IP-серверов применяется при аудите информационной безопасности, исследовании вредоносного программного обеспечения [4], восстановлении форматов сетевых протоколов и обеспечении межсистемной совместимости [3]. В условиях широкого использования закрытого программного обеспечения задачи анализа бинарного кода становятся ключевыми при исследовании сетевых приложений, серверных систем и распределённых сервисов.

Современные программные системы характеризуются высокой сложностью архитектуры, большим количеством взаимосвязанных модулей и использованием оптимизирующих компиляторов, что существенно затрудняет их анализ без исходного кода. В процессе реверс-инжиниринга исследователь вынужден работать с дизассемблированным или декомпилированным представлением программы, которое не содержит семантической информации о структуре исходной системы. Несмотря на развитие инструментов дизассемблирования и декомпиляции, процесс обратной разработки остаётся трудоёмким и во многом эвристическим [6, 7, 8].

В существующих исследованиях основной акцент делается на разработке методов автоматического анализа бинарного кода, включая классификацию функций, восстановление сигнатур, поиск точек входа и идентификацию структур данных. В последние годы активно развиваются подходы, основанные на применении методов машинного обучения и нейронных сетей для анализа бинарных программ, что позволяет частично автоматизировать процесс исследования и снизить объём ручной работы аналитика [11, 12]. Однако большинство работ ориентировано на повышение точности отдельных этапов анализа, тогда как количественная оценка сложности самого процесса обратной разработки остаётся недостаточно формализованной.

Существующие метрики сложности программного обеспечения, такие как цикломатическая сложность, глубина вложенности или характеристики графа управления, разработаны для анализа исходного кода и не учитывают специфику реверс-инжиниринга, при котором исследователь работает с бинарным представлением программы и ограниченной наблюдаемостью системы. В результате отсутствуют унифицированные методы измерения трудоёмкости анализа и оценки эффективности ассистивных инструментов, применяемых при исследовании бинарных приложений.

Таким образом, возникает противоречие между ростом сложности программных систем и отсутствием формализованных методов количественной оценки сложности их обратной разработки. Особенно актуальной является задача оценки влияния современных ассистивных средств, включая нейросетевые модели, на снижение трудоёмкости

Статья получена 15 марта 2026.

Зуб А.В., студент Московского Авиационного Института, Москва, Россия (e-mail: a.esseker.a@gmail.com),

П.П. Кейно, доцент кафедры №316 Московского Авиационного Института, Москва, Россия (e-mail: science@blockset.ru).

анализа и уменьшение объёма ручного исследования кода.

Объект исследования – процесс обратной разработки бинарного TCP/IP-сервера.

Предмет исследования – количественные характеристики сложности анализа и влияние нейросетевой поддержки на их изменение.

Цель работы – разработка формализованной модели оценки сложности обратной разработки TCP/IP-сервера и экспериментальная проверка влияния нейросетевого ассистивного анализа на снижение трудоёмкости процесса.

Для достижения цели решаются следующие задачи:

- Формализация структуры TCP/IP-сервера как сложной программной системы.
- Разработка интегрального показателя сложности анализа.
- Разработка методики экспериментального сравнения режимов анализа.
- Проведение сравнительного эксперимента и интерпретация результатов.

II. Обзор существующих подходов

Обратная разработка программного обеспечения представляет собой совокупность методов статического и динамического анализа бинарного кода, включающих построение графа вызовов, идентификацию точек входа, анализ сетевых обработчиков и реконструкцию протоколов. Статический анализ предполагает исследование исполняемого файла без его запуска, тогда как динамический анализ основан на наблюдении поведения программы во время выполнения. На практике эффективный реверс-инжиниринг требует комбинирования этих подходов, поскольку статический анализ позволяет получить глобальное представление о структуре программы, а динамический — уточнить семантику отдельных фрагментов кода [6–9].

Программные системы рассматриваются как иерархические структуры с высокой связностью и частично наблюдаемым поведением. При отсутствии исходного кода исследователь работает с низкоуровневым представлением программы, в котором отсутствуют имена функций, комментарии и информация о типах данных. Это существенно увеличивает трудоёмкость анализа и приводит к необходимости восстановления логической структуры программы на основе косвенных признаков. Классические метрики сложности программного обеспечения, такие как цикломатическая сложность, глубина вложенности, характеристики графа управления и информационная энтропия, ориентированы на анализ исходного кода [10] и не учитывают специфику реверс-инжиниринга при отсутствии семантической информации.

В последние годы активно развиваются методы автоматизации анализа бинарного кода, направленные на сокращение объёма ручной работы исследователя. К таким методам относятся автоматическое построение графов вызовов, восстановление сигнатур функций, обнаружение библиотечных вызовов, идентификация криптографических примитивов и анализ вредоносного программного обеспечения [5–9]. Однако большинство существующих инструментов ориентировано на поддержку отдельных этапов анализа и не предоставляет средств количественной оценки сложности процесса обратной разработки в целом.

Современные исследования анализа бинарных программ всё чаще используют методы машинного обучения и нейронных сетей [11, 12]. Такие подходы позволяют выполнять автоматическую классификацию функций, определять их вероятное назначение, восстанавливать структуру программных модулей и выявлять повторяющиеся шаблоны кода. Использование нейросетевых моделей особенно эффективно при анализе больших бинарных систем, содержащих тысячи функций и сложные графы вызовов, где ручной анализ становится крайне трудоёмким.

Несмотря на достигнутые успехи, большинство работ в данной области направлено на повышение точности распознавания отдельных элементов программы или на ускорение отдельных этапов анализа. В то же время практически отсутствуют исследования, посвящённые формализации сложности самого процесса реверс-инжиниринга и количественной оценке влияния ассистивных инструментов на трудоёмкость анализа. Не разработаны универсальные показатели, позволяющие сравнивать различные режимы анализа, учитывать объём ручной работы и оценивать эффективность применения автоматических и нейросетевых методов.

Следовательно, сохраняется исследовательский пробел в части количественной оценки сложности реверс-инжиниринга сетевых серверов и измерения вклада нейросетевой поддержки в её снижение. Разработка формализованной модели сложности анализа и экспериментальная проверка влияния ассистивных инструментов представляют собой актуальную научную задачу, имеющую как теоретическое, так и практическое значение.

III. Формализация модели TCP/IP-сервера

A. Представление сервера как сложной программной системы

При отсутствии исходного кода бинарный TCP/IP-сервер рассматривается как сложная программная система [1, 2] с частично наблюдаемым поведением, доступным через дизассемблирование, декомпиляцию и динамическую трассировку (1).

$$G = (V, E) \tag{1}$$

где V — множество функций бинарного модуля, E — отношения вызова между ними.

Помимо графовой структуры система характеризуется семантическими компонентами, соответствующими функциональным подсистемам обработки сетевых сообщений. В рамках исследования выделяются: сетевой контур; парсер протокола; диспетчер команд; обработчики команд; подсистема управления состоянием соединения; вспомогательные функции.

Данная декомпозиция используется как рабочая модель анализа бинарного кода.

B. Критерии достижения результата обратной разработки

Мы фиксируем, что «сервер восстановлен» на прикладном уровне, если получено:

- описание структуры пакетов (заголовок, длина, идентификатор команды, поля);
- список команд протокола и их назначение;
- соответствие «команда - обработчик»
- понимание ключевых состояний (например: до/после авторизации, активная сессия, ошибки, таймауты);
- минимальный набор тестов, подтверждающий корректность реконструкции.

C. Процесс обратной разработки как итеративная идентификация структуры

Процесс анализа интерпретируется как последовательность итераций, направленных на снижение структурной и семантической неопределённости системы.

Каждая итерация включает:

- Выделение потенциальных элементов сетевого контура.
- Локализацию границы парсера входных данных.
- Идентификацию механизма диспетчеризации команд.
- Формирование гипотез о структуре сообщений.
- Классификацию обработчиков.
- Верификацию гипотез средствами динамического анализа.

Таким образом, обратная разработка может быть представлена как процесс поэтапного сужения множества допустимых интерпретаций поведения системы.

D. Источники сложности анализа

Сложность реверс-инжиниринга обусловлена рядом факторов:

- большим размером множества функций V
- высокой связностью графа вызовов;
- отсутствием семантических идентификаторов;
- оптимизациями компилятора и возможной обфускацией;

Ключевой трудностью является необходимость сужения множества функций, требующих детального исследования.

Если обозначить через F полное множество функций бинарного модуля, а через $F_a \subseteq F$ — подмножество функций, подвергнутых детальному ручному анализу, то задача аналитика состоит в минимизации мощности множества $|F_a|$ при сохранении полноты реконструкции.

E. Ассистивный нейросетевой анализ

В предлагаемом подходе используется ассистивный инструмент анализа, основанный на нейросетевой модели трансформерного типа, предназначенной для обработки программного кода и дизассемблированных представлений функций.

Модель обучена на корпусе исходных текстов на языке C/C++, а также на наборах дизассемблированного кода, полученного из исполняемых файлов, с целью формирования представлений, позволяющих классифицировать функции по их предполагаемой роли в программной системе. В качестве входных данных использовались восстановленные функции, инструкции ассемблера и сигнатуры вызовов, полученные средствами дизассемблирования.

Нейросетевая модель применяется в ассистивном режиме и используется для:

- предварительной классификации функций по функциональной роли;
- выявления потенциальных точек входа в обработку сетевых сообщений;
- обнаружения повторяющихся шаблонов кода;
- формирования гипотез о назначении фрагментов программы;
- приоритизации функций для детального ручного анализа.

Важно отметить, что модель не принимает окончательных решений о семантике кода. Все выводы, полученные с использованием нейросетевой поддержки, подлежат обязательной проверке средствами статического и динамического анализа.

Формально использование ассистивной нейросетевой поддержки приводит к уменьшению мощности множества функций, подвергаемых детальному ручному анализу (2):

$$\left|F_a^{(NN)}\right| < \left|F_a^{(manual)}\right| \qquad (2)$$

где F — полное множество функций бинарного модуля; F_a — подмножество функций, подвергнутых детальному ручному анализу; $F_a^{(manual)}$ — множество функций, которые пришлось анализировать вручную в традиционном режиме; $F_a^{(NN)}$ — множество функций, анализируемых вручную при использовании нейросетевой поддержки.

Таким образом, нейросетевая модель рассматривается как механизм снижения структурной неопределённости системы до начала детального анализа.

IV. Методика оценки сложности обратной разработки

A. Общая схема эксперимента

Для количественной оценки влияния нейросетевой поддержки была разработана экспериментальная методика сравнения двух режимов анализа:

- R_0 — традиционный режим (без нейросетевых инструментов);
- R_1 — ассистивный режим (с использованием нейросетевой классификации функций).

Во всех остальных условиях параметры анализа были идентичны: использовались одни и те же инструменты дизассемблирования, среда выполнения и уровень оптимизации компиляции. Это обеспечивало сопоставимость результатов.

B. Определение измеряемых параметров

Для количественной оценки сложности вводятся три группы показателей.

- T — суммарное время анализа до достижения прикладного уровня восстановления протокольной логики.
- Измеряется в астрономических часах.
- F_a — количество функций, подвергнутых детальному ручному анализу.
- H — количество сформированных и проверенных гипотез о назначении функций или структур данных.

- Гипотеза считается учтённой, если она требовала проверки средствами статического или динамического анализа.

C. Нормализация показателей

Поскольку величины T, F_a и H имеют различную размерность, для построения интегрального показателя используется нормализация относительно традиционного режима (3):

$$\tilde{T} = \frac{T}{T_0}, \tilde{F_a} = \frac{F_a}{F_{a0}}, \tilde{H} = \frac{H}{H_0}, \quad (3)$$

где индекс 0 соответствует режиму R_0.

Таким образом, в традиционном режиме нормированные показатели равны 1.

D. Интегральный показатель сложности

Интегральная оценка сложности определяется как взвешенная сумма нормированных показателей (4):

$$C = \alpha\tilde{T} + \beta\tilde{F_a} + \gamma\tilde{H}, \quad \text{где } \alpha, \beta, \gamma \geq 0, \quad \alpha + \beta + \gamma = 1. \quad (4)$$

В рамках настоящего исследования использовались равные веса (5):

$$\alpha = \beta = \gamma = \frac{1}{3} \quad (5)$$

Такой выбор обусловлен отсутствием априорных оснований считать временные, структурные или семантические факторы более значимыми при оценке сложности обратной разработки. Использование равных коэффициентов позволяет рассматривать все компоненты как равноправные и избежать искусственного усиления влияния одного из параметров на интегральную оценку.

Подобный подход соответствует практике построения интегральных показателей в задачах системного анализа, когда отсутствуют объективные данные для задания различной значимости факторов.

E. Критерий эффективности ассистивного режима

Ассистивный режим R_1 считается эффективным, если выполняются условия:

$\tilde{T} < 1$ (снижение времени анализа);

$\tilde{F_a} < 1$ (снижение времени анализа);

полнота восстановления протокольной логики не ниже уровня режима R_0.

Дополнительно анализируется снижение интегрального показателя C.

F. Контроль влияния эффекта обучения

В целях минимизации влияния эффекта обучения аналитика режимы R_0 и R_1 применялись к различным алгоритмическим подсистемам сервера.

Анализировались функционально сходные, но структурно независимые алгоритмы обработки протокольных команд, сопоставимые по следующим характеристикам:

- количеству вовлечённых функций;
- глубине графа вызовов;
- числу формируемых гипотез о назначении функций;
- объёму реконструируемой логики.

Таким образом, второй режим не представлял собой повторный анализ ранее исследованной структуры, а применялся к новой подсистеме с сопоставимой алгоритмической сложностью.

Это позволило интерпретировать различие во временных и структурных показателях как результат использования ассистивного инструмента, а не как следствие накопленного опыта аналитика.

V. Экспериментальная оценка

A. Ограничения эксперимента

Полученные результаты следует интерпретировать с учетом ряда ограничений экспериментальной постановки. Эксперименты проводились одним исследователем, что может влиять на индивидуальные особенности стратегии анализа.

Полученные результаты получены на примере анализа бинарного TCP/IP-сервера с типичной архитектурой сетевых приложений. Рассматриваемый класс систем широко используется в сетевых сервисах и серверных приложениях, что позволяет рассматривать полученные оценки как репрезентативные для задач анализа сетевых протоколов.

B. Воспроизводимость эксперимента

Все эксперименты проводились в идентичных условиях анализа бинарного кода с использованием стандартных инструментов дизассемблирования и декомпиляции. Для каждого режима анализировались сопоставимые участки программы, содержащие обработчики сетевых команд и связанную инфраструктуру. Методика измерения времени и количества восстановленных строк кода сохранялась неизменной во всех сериях экспериментов.

C. Характеристики экспериментального объекта

В качестве экспериментального объекта использовался бинарный TCP/IP-сервер, реализованный на языке C++, скомпилированный в среде Microsoft Visual Studio 2010 для операционной системы Windows Server. Сервер использует объектно-ориентированную архитектуру с иерархией классов, виртуальными методами и обработчиками сетевых команд.

Программа характеризуется следующими параметрами:

- около 3000 функций в бинарном модуле;
- 255 команд прикладного протокола с однобайтовым идентификатором;
- наличие диспетчеризации команд через таблицы обработчиков;
- использование структур с частично неизвестными полями;
- отсутствие исходного кода и формальной спецификации протокола.

Цель анализа заключалась в восстановлении протокольной логики до уровня идентификации структуры пакетов, механизма диспетчеризации команд и ключевых состояний соединения.

D. Результаты традиционного режима R_0

Усредненные значения традиционного режима:

- время анализа: $T_0 = 43$ч;
- средняя скорость восстановления кода: 41 строка/ч;

- характерны возвраты к ранее исследованным участкам кода;
- наблюдалось значительное число повторных проверок гипотез.

Количество функций, подвергнутых детальному анализу, принято за базовое значение:

$$F_{a0.}$$

Количество проверенных гипотез:

$$H_0$$

E. Результаты ассистивного режима R_1

Усредненные значения ассистивного режима:

- время анализа:

$$T_1 = 30\text{ч};$$

- средняя скорость восстановления кода: 60 строк/ч;
- количество возвратов к ранее исследованным участкам минимизировано;
- ускорена идентификация диспетчера и структур классов.

F. Сравнение режимов R_0 и R_1

На рисунке 1 представлено сравнение режимов R_0 и R_1 по времени аналитической реконструкции в пяти независимых экспериментах. Во всех сериях наблюдается устойчивое снижение времени выполнения при использовании режима R_1 по сравнению с базовым режимом R_0.

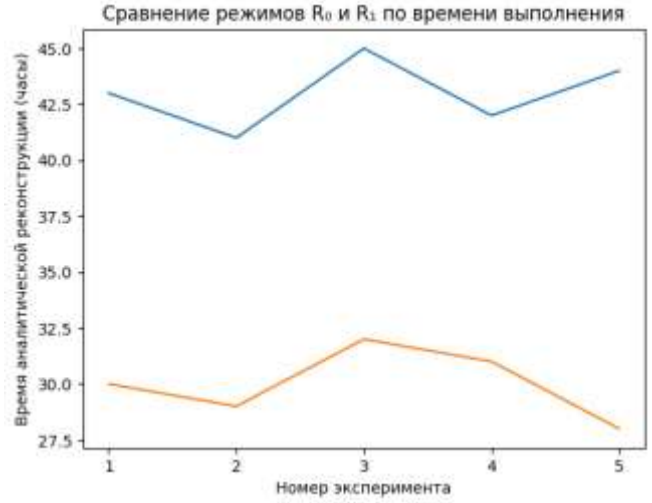

Рис. 1: Сравнение времени реконструкции кода

Численные результаты проведённых экспериментов приведены в таблице 1. В таблице представлены значения времени аналитической реконструкции для каждого эксперимента в режимах R_1 (ассистивный анализ) и R_0 (традиционный анализ).

Параметр T_{1i} соответствует времени выполнения i-го эксперимента в режиме R_1, а T_{0i} — времени выполнения того же эксперимента в базовом режиме R_0.

Полученные данные демонстрируют устойчивое снижение времени анализа при использовании ассистивного режима во всех сериях эксперимента.

Таблица I: Результаты экспериментального сравнения режимов R_0 и R_1

Номер эксперимента	T_{1i}	T_{0i}
1	30	43
2	29	41
3	32.5	45
4	32	42.5
5	27.5	43

Относительное снижение времени:

$$\tilde{T} = \frac{T_1}{T_0} = \frac{30}{43} \approx 0.7 \quad (6)$$

Таким образом, применение режима R_1 обеспечивает сокращение временной компоненты процесса обратной разработки примерно на 30% по сравнению с базовым режимом.

G. Интегральная оценка

При использовании нормализации из раздела 4 получаем (7):

$$\tilde{T} \approx 0.7, \quad \tilde{F_a} < 1 \quad \tilde{H} < 1 \quad (7)$$

При равных весах из (5) интегральная оценка сложности принимает вид (8):

$$C_1 = \frac{1}{3}\left(\tilde{T} + \tilde{F_a} + \tilde{H}\right) \quad (8)$$

Поскольку все три нормированные компоненты в режиме R_1 меньше единицы, выполняется (9):

$$C_1 < C_0 = 1. \quad (9)$$

Экспериментальные данные свидетельствуют о снижении интегральной сложности анализа примерно на 30-40%, что соответствует уменьшению временных и структурных затрат при сохранении полноты реконструкции.

H. Интерпретация результатов

Полученные результаты показывают, что ускорение обусловлено не механическим увеличением скорости чтения кода, а следующими факторами:

- предварительной структуризацией графа вызовов;
- приоритизацией функций для детального анализа;
- сокращением числа гипотез, требующих проверки;
- снижением когнитивной нагрузки при анализе иерархических структур.

При этом полнота восстановления протокольной логики в обоих режимах оставалась сопоставимой, что позволяет интерпретировать снижение времени как реальное уменьшение сложности анализа, а не как снижение качества реконструкции.

VI. Перспективы дальнейших исследований

Перспективным направлением дальнейших исследований является расширение экспериментальной базы за счет анализа других типов программных систем, включая распределенные сервисы и микросервисные архитектуры. Кроме того, представляет интерес исследование влияния различных типов ассистивных инструментов на отдельные компоненты сложности процесса обратной разработки.

Отдельным направлением может являться использование полученных экспериментальных результатов в задачах обучения нейросетевых моделей, предназначенных для поддержки анализа бинарного

кода. Полученные количественные оценки могут рассматриваться как ориентир (референсный уровень) эффективности, позволяющий оценивать степень улучшения, достигаемого при применении нейросетевых методов автоматизации реверс-инжиниринга.

VII. Заключение

В работе предложена формализованная модель оценки сложности обратной разработки бинарного TCP/IP-сервера в условиях отсутствия исходного кода и формальных спецификаций протокола. Предложенный подход основан на представлении анализируемой программной системы в виде ориентированного графа вызовов и совокупности функциональных подсистем обработки сетевых сообщений, что позволяет рассматривать процесс реверс-инжиниринга как задачу поэтапного снижения структурной и семантической неопределённости.

В ходе исследования получены следующие результаты.

- Показана возможность интерпретации бинарного TCP/IP-сервера как сложной программной системы с частично наблюдаемым поведением, описываемой ориентированным графом вызовов и набором функциональных компонент, соответствующих обработке сетевых сообщений.
- Процесс обратной разработки формализован как итеративная процедура идентификации структуры системы, включающая выдвижение и проверку гипотез о назначении функций, структур данных и связей между ними.
- Предложен интегральный показатель сложности анализа, объединяющий временные, структурные и семантические характеристики процесса. Разработана методика нормализации параметров и сопоставления различных режимов анализа, позволяющая выполнять количественную оценку трудоёмкости обратной разработки.
- Проведено экспериментальное исследование на бинарном TCP/IP-сервере средней сложности (около 3000 функций и 255 команд протокола), скомпилированном на языке C++ для операционной системы Windows Server. В рамках эксперимента выполнено восстановление протокольной логики на уровне структуры пакетов, диспетчеризации команд и ключевых состояний соединения, что позволило получить количественные оценки параметров анализа.
- Экспериментальные результаты показывают, что использование ассистивного инструмента, основанного на нейросетевой классификации функций и предварительной структуризации пространства поиска, сопровождается снижением временных и структурных затрат на анализ по сравнению с традиционным режимом. В проведённых сериях экспериментов наблюдалось уменьшение времени анализа порядка 30%, при сохранении сопоставимого уровня полноты восстановления протокольной логики.

Научная новизна работы заключается в формализации сложности процесса обратной разработки сетевого сервера и предложении интегрального показателя, позволяющего количественно оценивать трудоёмкость анализа бинарных программ и сравнивать различные режимы исследования.

Теоретическая значимость работы состоит в расширении методов системного анализа программных систем за счёт введения формализованной модели сложности реверс-инжиниринга и методики экспериментальной оценки влияния ассистивных инструментов на процесс анализа.

Практическая значимость заключается в возможности применения предложенной методики при сравнении средств анализа бинарного кода, а также при оценке эффективности инструментов, использующих методы машинного обучения и нейронных сетей, в задачах аудита сетевых сервисов, исследования проприетарных протоколов и анализа исполняемых программ без исходного кода.

Ограничением проведённого исследования является использование одного экспериментального объекта и выполнение анализа одним исследователем, что может влиять на полученные количественные оценки. В связи с этим дальнейшие исследования целесообразно проводить на расширенной выборке программных систем различной архитектуры и с участием нескольких аналитиков, что позволит повысить статистическую достоверность результатов и уточнить предложенную модель оценки сложности.

Библиография

[1] Таненбаум Э. С., Уэзеролл Д. Компьютерные сети. 5-е изд. СПб.: Питер, 2012.

[2] Таненбаум Э. С., Остин Т. Архитектура компьютера. 6-е изд. СПб.: Питер, 2014.

[3] Стивенс У. Р. TCP/IP. Подробное руководство. Т. 1: Протоколы. СПб.: Питер, 2003.

[4] Руссинович М., Соломон Д., Ионеску А., Йосифович П. Внутреннее устройство Windows. 7-е изд. СПб.: Питер, 2021.

[5] Сикорски М., Хониг Э. Вскрытие покажет! Практический анализ вредоносного ПО. СПб.: Питер, 2013.

[6] Касперски К. Искусство дизассемблирования. СПб.: БХВ-Петербург, 2009.

[7] Касперски К. Образ мышления — дизассемблер IDA. М.: СОЛОН-Р, 2001.

[8] Эйлам Э. Реверс-инжиниринг: исследование программ без исходных текстов. М.: Вильямс, 2007.

[9] Юричев Д. Reverse Engineering для начинающих [Электронный ресурс] Режим доступа: https://yurichev.com/news/20200227_anniversary/12-Mar-2013/RE_for_beginners-ru.pdf

[10] Кнут Д. Э. Искусство программирования. Т. 1: Основные алгоритмы. 3-е изд. М.: Вильямс, 2013.

[11] Shin E. C. R., Song D., Moazzezi R. Recognizing Functions in Binaries with Neural Networks [Распознавание функций в бинарных файлах с использованием нейронных сетей] // Proceedings of the 24th USENIX Security Symposium (USENIX Security 2015). Washington, D.C., 2015. P. 611–626.

[12] Rosenblum N., Zhu X., Miller B. P., Hunt K. Learning to Analyze Binary Code with Neural Networks [Анализ бинарного кода с использованием нейронных сетей] // Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2017). Atlanta, GA, 2017. P. 403–424.

Experimental Assessment of the Complexity of Reverse Engineering a TCP/IP Server Using Neural Networks

Artyom Zub, Pavel Keyno

***Abstract* — The paper addresses the problem of quantitative evaluation of the complexity of reverse engineering a binary TCP/IP server in the absence of source code and formal protocol specifications. A formalized model of the reverse engineering process is proposed, based on representing the analyzed software as a directed call graph and a set of functional components responsible for network message processing. An integral complexity metric is introduced that aggregates temporal, structural, and semantic parameters of the analysis.**

An experimental study of the influence of an assistive tool based on neural network function classification on the effort required for binary code analysis is conducted. The results show that preliminary structuring of the search space and prioritization of functions for detailed examination make it possible to reduce the total analysis time by approximately 30 % while maintaining a comparable level of completeness in protocol logic reconstruction.

The scientific novelty of the work lies in the formalization of the complexity of the reverse engineering process of a network binary TCP/IP server, the introduction of an integral metric for analysis effort, and the experimental evaluation of the impact of neural network–based assistance on reducing structural and semantic uncertainty during the analysis of binary programs.

***Keywords* — reverse engineering, system analysis, TCP/IP server, call graph, neural networks, complexity estimation, binary analysis.**

REFERENCES

[1] Tanenbaum, E.S., Weatherall, D. Computer Networks. 5th ed. — St. Petersburg: Piter, 2012. — 960 p.

[2] Tanenbaum, E.S., Austin, T. Computer Architecture. 6th ed. — St. Petersburg: Piter, 2014. — 816 p.

[3] Stevens, W.R. TCP/IP. The Definitive Guide. Vol. 1: Protocols. — St. Petersburg: Piter, 2003. — 672 p.

[4] Russinovich, M., Solomon, D., Ionescu, A., and Iosifovich, P. Windows Internals. 7th ed. — St. Petersburg: Piter, 2021. — 944 p.

[5] Sikorski, M., Honig, E. The Autopsy Will Reveal! Practical Malware Analysis. — St. Petersburg: Piter, 2013. — 800 p.

[6] Kaspersky K. The Art of Disassembling. — St. Petersburg: BHV-Petersburg, 2009. — 896 p.

[7] Kaspersky K. Way of Thinking — the IDA Disassembler. — Moscow: SOLON-R, 2001. — 480 p.

[8] Eilam E. Reverse Engineering: Studying Programs Without Source Code. — Moscow: Williams, 2007. — 592 p.

[9] Yurichev D. Reverse Engineering for Beginners [Electronic resource]. — Access mode: https://yurichev.com/news/20200227_anniversary/12-Mar-2013/RE_for_beginners-ru.pdf

[10] Knuth D. E. The Art of Computer Programming. Vol. 1: Basic Algorithms. 3rd ed. — Moscow: Williams, 2013. — 720 p.

[11] Shin E. C. R., Song D., Moazzezi R. Recognizing Functions in Binaries with Neural Networks // Proceedings of the 24th USENIX Security Symposium (USENIX Security 2015). — Washington, D.C., 2015. — P. 611–626.

[12] Rosenblum N., Zhu X., Miller B. P., Hunt K. Learning to Analyze Binary Code with Neural Networks // Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2017). - Atlanta, GA, 2017. - P. 403–424.

About of Authors

Artyom V. Zub, bachelor student, Moscow Aviation Institute, Moscow, Russia (e-mail: science@blockset.ru).

Pavel P. Keyno, Associate Professor, Department No. 316, Moscow Aviation Institute, Moscow, Russia (e-mail: science@blockset.ru).