УДК 004.056 DOI: 10.17587/it.24.594-600

М. А. Стюгин, канд. техн. наук, e-mail: styugin@gmail.com,
В. В. Золотарев, канд. техн. наук, e-mail: amida.2@yandex.ru,
Н. Ю. Паротькин, канд. техн. наук, e-mail: nyparotkin@yandex.ru,
Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнева, г. Красноярск

Подход к защите информационных систем от уязвимостей, не выявленных на стадии проектирования системы[1]

Рассмотрена проблема защиты информационных систем от уязвимостей, не выявленных на стадии проектирования системы. Показано, что основной способ защиты от подобных уязвимостей — защита информационных систем и процессов от их исследования со стороны злоумышленника. В работе предложена формальная постановка задачи защиты от исследования, выраженная в получении неразличимых информационных систем по целевой функции и побочным каналам утечки информации. Доказывается, что неразличимая обфускация позволяет получить неразличимость по побочным каналам утечки данных. Предложен также способ приведения систем к неразличимости по целевой функции. Доказана работоспособность метода на примерах удаленных атак инъекции кода в приложение.

Ключевые слова: *защита от исследования, неразличимость, информационная безопасность, инъекция программного кода, неразличимая обфускация*

Введение

Проблема, обозначенная в заголовке данной статьи, столь фундаментальна и всеобъемлюща, что, найдя для нее решение, мы смогли бы раз и навсегда решить проблему информационной безопасности (ИБ) любых информационных систем. Действительно, если мы взглянем на все крупнейшие инциденты по защите информации за последний год, то обнаружим, что все они основаны на уязвимостях "нулевого дня", когда уязвимость найдена злоумышленником, но у служб безопасности было еще только 0 дней для ее обработки, анализа и устранения. Значительно растет рынок систем защиты, предназначенных для борьбы с "неизвестным". Сюда можно отнести, например, системы ATP (Advanced Threat Protection) или SIEM (Security Information and Event Management). Задача первых — анализировать поведение вредоносного ПО, которое не может быть идентифицировано как вредоносное путем сигнатурных или эвристических методов, в то время как задача SIEM — обнаружить неизвестные инциденты за счет корреляции "странных" событий в машинных данных (например, пользователь авторизовался под своей учетной записью в офисе, в то время как по данным системы учета рабочего времени находился в отпуске). Но все это, безусловно, не дает каких-либо гарантированных показателей защищенности информационной системы.

В то время как инженеры наращивают базы корреляции событий ИБ и латают дыры в системах, основываясь на анализе последних инцидентов и обновлений базы CVE, научные исследования все более ориентируются на разработку методов построения систем защиты, которые давали бы возможность защититься от неизвестных уязвимостей, не выявленных на стадии проектирования. Решение такой задачи в области ИБ сравнимо с мечтой о построении "единой теории поля" в физике. Поэтому было бы логично придумать ей столь же запоминающееся название. Назовем ее *единой*

[1] Работа поддержана Российским фондом фундаментальных исследований по проекту № 16-29-09456 офи_м.

теорией построения неразличимых информационных систем. Почему "неразличимых"? Здесь мы сталкиваемся с проблемой информационной асимметрии между защитником и атакующим [1]. Атакующий относительно защитника действует в "завтрашнем дне", где располагает большей информацией о возможных способах атак на систему и имеет неограниченное время на ее исследование и анализ. Преодолеть эту информационную асимметрию можно только за счет защиты информационной системы от исследования.

В данной статье мы приведем краткий обзор трендов в этой области и определим, как задача построения неразличимой информационной системы может быть решена на примере конкретной модели.

Обзор исследований

Публикации нескольких последних лет явно отражают рост интереса к проблеме защиты информационных систем от исследования в целях повышения безопасности. Защищая систему от исследования, мы затрудняем поиск уязвимостей в ней, при этом сами уязвимости не устраняя. Здесь можно выявить несколько различных подходов, одним из которых является борьба с монокультурой программного обеспечения (software monoculture) [2]. Допустим, злоумышленник получил в свое распоряжение одну из копий программного обеспечения и нашел в ней уязвимость, позволяющую реализовать внедрение и запуск исполняемого кода. Злоумышленник может сделать вывод, что все остальные копии данной программы с текущей версией также имеют данную уязвимость. Поскольку таких копий ПО могут быть тысячи и даже миллионы, он может написать скрипт, автоматизирующий процесс эксплуатации уязвимости, и нанести вред огромному числу пользователей и информационных систем. Для того чтобы этого не произошло, можно сделать так, чтобы каждая новая копия ПО была уникальна.

Решение этой проблемы заключается в рандомизации исполняемого кода (instruction-set randomization, ISR) [3, 4]. Основоположники технологии ISR утверждают, что сама идея взята из биологии, где организмы реагируют на множественные угрозы окружающей среды путем генетических изменений. По аналогии, механизм ISR изменяет основные исполняемые инструкции таким образом, что встраиваемый посторонний код внутри приложения не приводит к корректному исполнению.

Другим популярным подходом диверсификации программного обеспечения, который получил большую популярность последние несколько лет, — это мутация исполняемого кода [5].

Автоматическая мутация программ позволяет также осуществлять автоматическое исправление ошибок. Как правило, такое автоматическое исправление ошибок заключается в применении различных паттернов структур программного кода (проверка дополнительных условий, преобразования циклов и пр.) [6].

Популярным методом в области защиты от неизвестных уязвимостей является также технология Moving Target Defense (MTD). Суть технологии заключается в защите системы от исследования путем ее непрерывного изменения. На сегодняшний день в публикациях насчитывается более 150 различных техник в области MTD. Наиболее известные методы MTD — это рандомизация адресного пространства (Address Space Layout Randomization — ASLR) [7], рандомизация данных [8], перемешивание адресации локальной сети [9], защита от DDoS атак [10], защиты от XSS (cross-site scripting) атак на сайт [11] и пр.

В данной статье мы попробуем формализовать задачу получения неразличимых информационных систем, когда обозначенные выше методы могут быть частным случаем достижения неразличимости.

Формализация задачи

Любая формализация вынуждает нас перейти от абстрактной всеобъемлющей задачи к некой более узкой проблеме. Первый шаг к такому переходу — это формализация самого понятия "информационная система". В рамках данной статьи будем предполагать, что информационная система — это некоторый черный ящик, имеющий вектор входных значений x и результирующий вектор работы системы y. Вектор y состоит из совокупности трех векторов (y_1, y_2, y_3): y_1 — это значения, возвращаемые как результат воздействия x на систему, y_2 — это вектор изменения состояния системы и y_3 — это побочный результат выполнения алгоритма, который может быть выражен, например, во временных задержках возврата значения y_1, побочных электромагнитных излучениях и пр.

По-другому y_3 будем называть "отпечатком" работы алгоритма. На x мы не накладываем каких-либо ограничений, так он может включать в себя вектор изменения состояний системы y_2. В целом такая конструкция не меняет дальнейших выводов.

Обозначим также $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$ и $\mathbf{X}$ соответствующие множества, которым принадлежат данные вектора.

Целевая функция системы может быть выражена как

$$f{:}\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2.$$

Мы также можем выделить полезную функцию системы на основе полезного подмножества входных векторов $\mathbf{X}_u \subseteq \mathbf{X}$:

$$f{:}\mathbf{X}_u \to \mathbf{Y}_1 \times \mathbf{Y}_2.$$

Полезная функция системы — это то, ради чего данная система была создана. Она единственная должна выполняться корректно и правильно. Очевидно, что все, что находится за пределами полезной функции, должно быть неразличимо для остальных участников системы, которые могут генерировать произвольные значения входа из множества $\mathbf{X}$.

Добавим в систему уязвимости, выраженные как подмножество значений входа $\mathbf{X}_{vul} \subseteq \mathbf{X}$, которые на выходе формируют нежелательный для нас результат $(\mathbf{Y}_1 \times \mathbf{Y}_2)_{attack}$:

$$f{:}\mathbf{X}_{vul} \to (\mathbf{Y}_1 \times \mathbf{Y}_2)_{attack}.$$

При этом $\mathbf{X}_{vul} \cap \mathbf{X}_u = \varnothing$. Также введем функцию побочных каналов утечки информации:

$$f_{sc}{:}\mathbf{X} \to \mathbf{Y}_3.$$

На основе предложенного формализма мы можем ввести два свойства неразличимости информационной системы:

- *неразличимость по целевой функции* — итеративный доступ к функции $f{:}\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$ с наблюдаемыми значениями $\mathbf{Y}_1 \times \mathbf{Y}_2$ дает информации о значениях множества $\mathbf{X}_{vul}$ не больше, чем в случае, когда $\mathbf{Y}_1 \times \mathbf{Y}_2$ являются ненаблюдаемыми с точностью до пренебрежимо малых величин;
- *неразличимость по побочным каналам утечки информации* — итеративный доступ к функции $f_{sc}{:}\mathbf{X} \to \mathbf{Y}_3$ с наблюдаемыми значениями $\mathbf{Y}_3$ дает информации о значениях целевой функции $f{:}\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$ не больше, чем в случае, когда $\mathbf{Y}_3$ являются ненаблюдаемыми с точностью до пренебрежимо малых величин.

Неразличимая информационная система — это такая система, которая обладает свойствами неразличимости по целевой функции и побочным каналам утечки информации.

Сделаем небольшие пояснения. Неразличимость по целевой функции означает, что злоумышленник путем отправки значений $x \in \mathbf{X}$ и получения от системы ответов (y_1, y_2), где $y_1 \in \mathbf{Y}_1$ и $y_2 \in \mathbf{Y}_2$, не получает какой-либо информации о любом из элементов $x_{vul} \in \mathbf{X}_{vul}$, если ранее он этой информацией не обладал. Неразличимость по побочным каналам утечки информации означает, что наблюдение значений $y_3 \in \mathbf{Y}_3$ как результата работы системы не может быть злоумышленником сколько-нибудь осмысленно интерпретировано для того, чтобы получить информацию о целевой функции, выполняемой системой. Понятие "точность до пренебрежимо малых величин" означает, что в обоих случаях мы не можем найти вероятностно-полиномиальный алгоритм по прогнозированию неизвестных значений множества $\mathbf{X}_{vul}$ (в случае неразличимости по целевой функции) или ненаблюдаемых ранее значений функции $f{:}\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$ (в случае неразличимости по побочным каналам) с вероятностью больше, чем пренебрежимо малая величина $\alpha(\lambda)$, неполиномиально быстро уменьшающаяся от некоторого параметра λ, характеризующего уровень стойкости системы ко взлому.

Пример неразличимости по целевой функции: путем отправки GET и POST запросов на сайт с различным составом и содержанием переменных злоумышленник не может определить, какие из запросов могут спровоцировать инъекцию кода, т. е. найти хоть один элемент $x_{vul} \in \mathbf{X}_{vul}$.

Пример неразличимости по побочным каналам утечки информации: злоумышленник отправил несколько сообщений в некоторую информационную систему, где на них была сгенерирована цифровая подпись. При этом он анализировал все побочные каналы утечки информации: задержки времени, изменение напряжения питания информационной системы, электромагнитное излучение и пр. Вся полученная информация никак не повлияла на способность злоумышленника спрогнозировать значение цифровой подписи для ранее не подписанного сообщения m.

Получение неразличимости по побочным каналам утечки информации

Интуитивно можно предположить, что, не зная алгоритм работы системы, мы не сможем интерпретировать информацию, полученную от нее по побочным каналам утечки информации.

Проведем далее формальное доказательство этого факта.

Допустим, у нас есть два различных алгоритма C_0 и C_1, выполняющих различные функции $f:\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$, и также различные по функциям $f_{sc}:\mathbf{X} \to \mathbf{Y}_3$. Таким образом, мы имеем на каждый из алгоритмов его уникальный "отпечаток", выраженный в функциях $f_{sc}^{(C_0)}$ и $f_{sc}^{(C_1)}$.

Предположим далее, что мы не знаем выполняемую системой функцию $f:\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$, но можем наблюдать значения функции $f_{sc}:\mathbf{X} \to \mathbf{Y}_3$. Таким образом, мы можем видеть "отпечатки" алгоритма $f_{sc}^{(C_0)}$ и $f_{sc}^{(C_1)}$.

Допустим, у нас решена задача неразличимости функции $f:\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$ по ранее полученным значениям входа и выхода. Иными словами, по полученным значениям $(x_1, y_1^{(1)}, y_1^{(2)})$, $(x_2, y_2^{(1)}, y_2^{(2)})$, ..., $(x_n, y_n^{(1)}, y_n^{(2)})$ для наблюдателя, выполняющего только полиномиально-вероятностные алгоритмы, задача вычисления корректного значения $(x_k, y_k^{(1)}, y_k^{(2)})$ для раннее неиспользованного входа x_k является недостижимой с точностью до пренебрежимо малых величин.

Такая неразличимость по функции не обязательно будет сохраняться при наблюдении также значений $f_{sc}:\mathbf{X} \to \mathbf{Y}_3$. Например, злоумышленник может зафиксировать задержки в работе вычислителя, основанные на возведении в степень больших чисел. Зная принцип построения алгоритма цифровой подписи, он может по этим "отпечаткам" вычислить ключ и, тем самым, получить функцию $f:\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$, после чего генерировать новые пары значений $(x_k, y_k^{(1)}, y_k^{(2)})$.

Каждый алгоритм C_0 может быть модифицирован таким образом, что функция $f:\mathbf{X} \to \mathbf{Y}_1 \times \mathbf{Y}_2$ останется неизменной, но при этом функция $f_{sc}:\mathbf{X} \to \mathbf{Y}_3$ поменяется. Таким образом, мы можем получить множество алгоритмов C_0^0, C_0^1, ..., C_0^n, выполняющих одинаковую функцию, но с различными "отпечатками" $f_{sc}^{(C_0^0)}$, $f_{sc}^{(C_0^1)}$, ..., $f_{sc}^{(C_0^n)}$. Злоумышленник может получить все данные о соответствии алгоритмов и их "отпечатков".

Теорема 1. Если к алгоритму C информационной системы применить неразличимую обфускацию $iO(C)$, то такая система становится неразличимой по побочным каналам утечки информации.

Докажем это утверждение от противного. После того как мы обфусцировали алгоритм $iO(C)$, мы получили также новый "отпечаток" $f_{sc}^{(iO(C))}$. Если этот новый "отпечаток" не имеет какой-либо связи с "отпечатком" системы до обфускации, то на основании полученной по нему информации мы не сможем получить какую-либо информацию об используемом алгоритме. Предположим, что это не так. Возьмем два исходных алгоритма C_0 и C_1 равной функциональности, но с различными "отпечатками" $f_{sc}^{(C_0)}$ и $f_{sc}^{(C_1)}$. Тогда существует такой вероятностно-полиномиальный алгоритм D (Distinguisher — различитель), для которого выполняется следующее равенство:

$$\left|\Pr[D(f_{sc}^{(iO(C_0))})] - \Pr[D(f_{sc}^{(iO(C_1))})\right| > \alpha(\lambda),$$

где $\alpha(\lambda)$ — пренебрежимо малая величина, а λ — параметр стойкости обфускации, определяющий класс исходных алгоритмов C_0 и C_1; Pr — обозначение вероятности события. Далее из очевидной связи "отпечатков" $f_{sc}^{(iO(C_0))}$ и $f_{sc}^{(iO(C_1))}$ и исходных алгоритмов $iO(C_0)$ и $iO(C_1)$ следует существование вероятностно-полиномиального алгоритма D', для которого выполняется условие

$$\left|\Pr[D'(iO(C_0))] - \Pr[D'(iO(C_1))]\right| > \alpha(\lambda).$$

Данное утверждение противоречит основному свойству неразличимой обфускации [12]. Таким образом, мы доказали, что информационная система, к алгоритму которой была применена неразличимая обфускация, становится неразличимой по побочным каналам утечки информации.

Получение неразличимости по целевой функции

Если принять во внимание, что $\mathbf{X}_{vul} \cap \boldsymbol{X}_{\boldsymbol{u}} = \emptyset$, то может возникнуть предположение, что достаточно просто ограничить множество входных параметров диапазоном $\boldsymbol{X}_{\boldsymbol{u}}$ и тем самым сделать систему безопасной. Безусловно, есть случаи, для которых такая логика применима. Однако для большинства описанных информационных систем диапазон $\boldsymbol{X}_{\boldsymbol{u}}$ столь огро-

мен, что валидировать его не представляется возможным. Примером может быть интернет-сайт, в котором есть форум для общения пользователей. Допустим, поле ввода комментария на форуме ограничено длиной 500 символов. Тогда все возможные входные данные для поля комментария ограничены значением 256^{500}. Здесь могут быть и значения из пространства $\mathbf{X}_{vul}$, опасные для системы, например инъекции SQL-запросов, XSS и пр. Таким образом, нужно как-то ограничить "бесконечность" — пространство $\boldsymbol{X_u}$ — с учетом того, что мы не можем знать все его значения. Свойство неразличимости по целевой функции требует, чтобы за пределами $\boldsymbol{X_u}$ не были предсказуемы такие значения $\mathbf{X}_{vul}$, для которых мы получили бы $f{:}\mathbf{X}_{vul} \rightarrow (\mathbf{Y}_1 \times \mathbf{Y}_2)_{attack}$.

Таких решений на сегодняшний день не существует, и, наиболее вероятно, они не могут быть найдены. В данной статье будет предложен метод, когда система может самомодифицироваться, приближаясь к состоянию неразличимости по целевой функции.

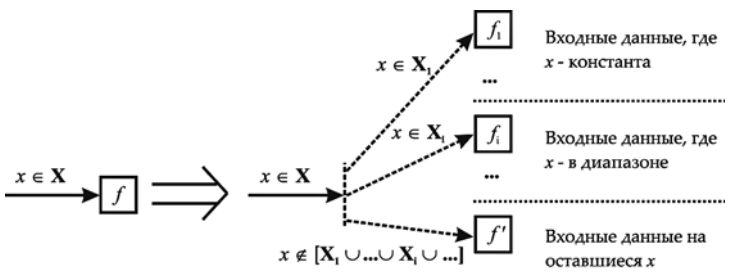
Рассмотрим информационную систему, реализующую некую функцию f на диапазоне входных данных $\mathbf{X}$. Допустим, это некий скрипт, выполняющийся на веб-сервере. Запустив скрипт в работу и анализируя его выполнение, мы можем в дальнейшем разделить алгоритм работы скрипта по различным диапазонам входных значений (см. рисунок).

Некоторый фиксированный набор значений на входе может повторяться многократно. Таким образом, мы можем выделить конечный набор переменных и сделать для них "усеченную" копию исходного скрипта. При этом, анализируя выполнение кода на ограниченном наборе входных переменных, мы можем обнаружить, что некоторые условные операторы или подпрограммы не выполняются, и удалить их из "усеченного" скрипта. Если в выполнении скрипта не участвуют внешние переменные, то мы можем просто сгенерировать таблицу заранее вычисленных значений выхода. Это будет в некотором роде кеш программы. Таким образом, мы получаем набор скриптов $f_1..f_{(i-1)}$ для обработки фиксированных значений входа.

Следующий набор скриптов $f_i..f_n$ выполняется на входных значениях, ограниченных в некотором диапазоне. Здесь мы можем проверить, что какая-то конкретная переменная является числом или имеет формат электронного адреса. Скрипты $f_i..f_n$ также являются "усеченным" вариантом исходного скрипта с функцией f с выбрасыванием всех неиспользуемых условных операторов и подпрограмм по заданным диапазонам входных значений. Кроме этого, в скрипты на таких "диапазонных значениях" мы также добавляем программную избыточность. Эта избыточность добавляет случайные или псевдослучайные прямые и обратные преобразования в переменные и смену формата переменных, т. е. значение, которое пользователь ввел в поле ФИО, конкатенируется со строкой "123". С этим же значением конкатенируются переменные при считывании полей из базы данных. Таким образом, логика приложения не нарушается, но добавленная псевдослучайная избыточность не дает возможности предсказать, как именно будет обработан запрос. Один из таких методов описан в работе. Здесь применимы методы, описанные в начале статьи.

Оставшиеся данные, которые нельзя валидировать в каких-либо диапазонах, попадают в копию скрипта с функцией f'. Здесь уже применяется не только избыточность, но и модификация данных, не приводящая к нарушению логики работы программы, а также мутация алгоритмов. Здесь необходимо анализировать активность и, в случае попадания активности в полезную функцию работы системы, расширять диапазон значений по скриптам $f_i..f_n$.

Физически все копии исходного скрипта могут быть разнесены по разным вычислительным средам в зависимости от степени доверия к ним. Эксплуатация уязвимостей в модулях $f_1..f_{(i-1)}$ невозможна. Нахождение и эксплуатация уязвимостей в модулях $f_i..f_n$ имеет пренебрежимо малую вероятность. Эксплуатация уязвимостей в модуле f' также невозможна, поскольку модуль представляет собой "песочницу" (все запросы в нем выполняются не на реальных, а на искусственно сгенерированных данных), но в него возможно попада-



Разделение функций работы скрипта по диапазонам

ние легальных запросов, что может привести к проблемам функционирования системы.

Практическая реализация системы с неразличимой целевой функцией

Описанный в предыдущем разделе механизм трансформации программ по диапазонам входных данных может эффективно справляться с любыми удаленными атаками, ориентированными на ошибки разработчика и инъекции программного кода. Однако для получения действительно эффективных механизмов защиты стоит рассмотреть алгоритмы, осуществляющие такую кластеризацию скрипта по входным диапазонам. Здесь может использоваться современный аппарат системного анализа и искусственного интеллекта. Для разработки таких алгоритмов необходимо дополнительное исследование.

В рамках настоящей работы нам необходимо подтвердить корректность достижения результатов исследования. Здесь мы не будем использовать сложные алгоритмы кластеризации и ограничимся только тремя блоками f_1, f_2, f'.

Разработанная техническая реализация получала на вход скрипт и некий набор входных значений. На основе автоматического анализа входных значений строился так называемый валидатор на входе скрипта, который адресовал выполнение программы к блокам f_1, f_2 или f'. Входные значения для f_1 — это все значения, которые повторились хотя бы один раз. Входные значения для f_2 формировались автоматически по маскам диапазонов. В качестве примеров маски задавались форматы числа, электронной почты, строки без спецсимволов и даты. Соответственно, на вход f' попадало все, что не соответствовало валидации по двум другим блокам. В качестве смещения алгоритмов в блоке f_2 использовались методы, описанные в работах [11, 13]. Блок f' имел доступ к базе данных посредством только механизма data masking.

Для эксперимента был использован скрипт с заранее известными точками входа для атак SQL-инъекций и межсайтового скриптинга. Ни одного положительного сценария атаки после кластеризации системы найдено не было. Более того, в результате пентестов (penetration test) не удалось исследовать диапазоны валидации значений сайта, поскольку ни один из входящих запросов системой не блокируется, даже если запрос содержит признаки инъекции кода.

Заключение

В данной работе обозначен современный тренд построения систем безопасности на основе технологий, позволяющих защитить системы от внешнего исследования. Построена формальная модель неразличимой информационной системы. Данная модель позволяет решать задачи защиты информационных систем от исследования, когда множество входных значений может содержать недопустимые для системы данные, но при этом в силу своих размеров не может быть проверено на наличие уязвимостей. Доказано, что неразличимая обфускация делает алгоритм неразличимым по побочным каналам утечки информации. Разработан метод автоматического преобразования скрипта, позволяющий защитить алгоритм от исследования его работы за пределами полезной функции системы. Проведено практическое исследование корректности работы метода.

Список литературы

1. **Jajodia S., Ghosh A. K., Subrahmanian V. S.** at al. Moving Target Defense II. Application of Game Theory and Adversarial Modeling. Series: Advances in Information Security. Springer-Verlag New York, 2013. 203 p.

2. **Gherbi A., Charpentier R.** Diversity-based approaches to software systems security // Communications in Computer and Information Science. 2011. Vol. 259. P. 228—237.

3. **Chan A.** Automated Program Diversity Using Program Synthesis // Proceedings — 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2017. 2017. P. 156—159.

4. **Styugin M., Zolotarev V., Prokhorov A., Gorbil R.** New Approach To Software Code Diversification In Interpreted Languages Based On The Moving Target Technology // Proceedings of the 10th IEEE International Conference on Application of Information and Communication Technologies (AICT 2016). 2016. P. 1—5.

5. **Larsen P., Brunthaler S., Franz M.** Security through diversity // IEEE Security and Privacy. 2014. Vol. 12, Iss. 3. P. 28—35.

6. **GenProg:** A Generic Method for Automatic Software Repair / C. L. Goues и др. // IEEE Transactions on software engineering. 2012. Vol. 38, Iss. 1. P. 54—72.

7. **Chongkyung K., Jinsuk J., Bookhold C.** et al. Address Space Layout Permutation (ASLP): Towards fine-grained randomization of commodity software // Annual Computer Security Applications Conference (ACSAC). 2006. P. 339—348.

8. **Bhatkar S., Sekar R.** Data space randomization // Lecture Notes in Computer Science. 2008. Vol. 5137. P. 1—22.

9. **Carroll T. E., Crouse M., Fulp E. W.** et al. Analysis of network address shuffling as a moving target defense // 2014 IEEE International Conference on Communications. 2014. P. 701—706.

10. **Ma D., Xu Z., Lin D.** Defending blind DDoS attack on SDN based on moving target defense // Lecture Notes of the Institute for Computer Sciences, LNICST. 2015. Vol. 152. P. 463—480.

11. **Portner J., Kerr J., Chu B.** Moving target defense against cross-site scripting attacks // Lecture Notes in Computer Science. 2015. Vol. 8930. P. 85—91.

12. **Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.** Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits // 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. 2013. P. 40—49.

13. **Styugin M., Parotkin N.** Multilevel Decentralized Protection Scheme Based on Moving Targets // International Journal of Security and Its Applications. 2016. Vol. 10, N. 1. P. 45—54.

M. A. Styugin, Ph. D., e-mail: styugin@gmail.com,
V. V. Zolotarev, Ph. D., e-mail: amida.2@yandex.ru,
N. Y. Parotkin, e-mail: nyparotkin@yandex.ru,
Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russian Federation

Protection of Information Systems from Undetected Vulnerabilities

The problem of protecting information systems from vulnerabilities that were not detected at a system's development stage is considered. It is demonstrated that the main technique for protection from vulnerabilities of that kind is protecting information systems and processes from reconnaissance by an adversary. The formal definition of the protection's objective, which is establishing undistinguishable information systems with the target function and data leak side channels, is presented in the paper. Indiscernibility by target function means that attacker's ability to send data to the system and observe the results of its operation does not provide any information that can be used for exploiting the system's vulnerabilities. Indiscernibility by side channels means that observation of all external manifestations of the system's functioning, such as time delays, impact on the computing device, etc. do not provide any information on the value of the system's target function. The system's indiscernibility by target function can be established by separating the input value range and applying different processing logic for them. A modified algorithm for running in an untrusted computation environment should be implemented for non-standard input values. Proof is provided that indistinguishable obfuscation allows establishing Indistinguishability via data leak side channels. A method for providing a system's indistinguishability by the target function. The technique's effectiveness is demonstrated with instances of remote code injection attacks at an application.

Keywords: *protection from research, indistinguishability, code injection attack, indistinguishable obfuscation*

DOI: 10.17587/it.24.594-600

References

1. **Jajodia S., Ghosh A. K., Subrahmanian V. S.** at al. Moving Target Defense II. Application of Game Theory and Adversarial Modeling. Series: Advances in Information Security, Springer-Verlag New York, 2013, 203 p.

2. **Gherbi A., Charpentier R.** Diversity-based approaches to software systems security, *Communications in Computer and Information Science*, 2011, vol. 259, pp. 228—237.

3. **Chan A.** Automated Program Diversity Using Program Synthesis, *Proceedings — 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2017*, 2017, pp. 156—159.

4. **Styugin M., Zolotarev V., Prokhorov A., Gorbil R.** New Approach To Software Code Diversification In Interpreted Languages Based On The Moving Target Technology, *Proceedings of the 10th IEEE International Conference on Application of Information and Communication Technologies (AICT 2016)*, 2016, pp. 1—5.

5. **Larsen P., Brunthaler S., Franz M.** Security through diversity, *IEEE Security and Privacy*, 2014, vol. 12, iss. 3, pp. 28—35.

6. **GenProg:** A Generic Method for Automatic Software Repair / C. L. Goues et al, *IEEE Transactions on software engineering*, 2012, vol. 38, iss. 1, pp. 54—72.

7. **Chongkyung K., Jinsuk J., Bookhold C.** at al. Address Space Layout Permutation (ASLP): Towards fine-grained randomization of commodity software, *Annual Computer Security Applications Conference (ACSAC),* 2006, pp. 339—348.

8. **Bhatkar S., Sekar R.** Data space randomization, *Lecture Notes in Computer Science*, 2008, vol. 5137, pp. 1—22.

9. **Carroll T. E., Crouse M., Fulp E. W.** at al. Analysis of network address shuffling as a moving target defense, *2014 IEEE International Conference on Communications*, 2014, pp. 701—706.

10. **Ma D., Xu Z., Lin D.** Defending blind DDoS attack on SDN based on moving target defense, *Lecture Notes of the Institute for Computer Sciences, LNICST,* 2015, vol. 152, pp. 463—480.

11. **Portner J., Kerr J., Chu B.** Moving target defense against cross-site scripting attacks, *Lecture Notes in Computer Science*, 2015, vol. 8930, pp. 85—91.

12. **Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.** Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits, *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013, pp. 40—49.

13. **Styugin M., Parotkin N.** Multilevel Decentralized Protection Scheme Based on Moving Targets, *International Journal of Security and Its Applications,* 2016, vol. 10, no. 1, pp. 45—54.