ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Исследование методов сбора печатаемых символов

Отчёт подготовили _____ Н.Н. Ефанов

подпись

Отчёт принял _____ А.А. Чапчаев

подпись

Москва/Долгопрудный/Симферополь, 2016

Оглавление

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

ССВ-1	Система стенографического ввода
СДШВ-1	Система динамически шифруемого ввода
ОСВ	Оператор системы ввода
ФС	Файловая система

ВВЕДЕНИЕ

В современном мире информационных технологий проблема перехвата и анализа данных от устройств пользовательского ввода является важнейшей задачей класса взаимодействия с пользователем, ввиду необходимости построения эффективных средств защиты конфиденциальных данных и мониторинга активности пользователя в ходе работы с электронным устройством [1-3, 5-6]. Подавляющее большинство ключевых данных (тексты, номера, пароли, адреса в сети Интернет) вводится с различных клавиатур в виде последовательностей кодов символов. Средства для перехвата, хранения и зачастую использования введенных данных в отечественной литературе называются клавиатурными шпионами, однако нами будет использован заимствованный термин “кейлоггеры”.

“Кейлоггеры(от англ. “Keylogger”, правильное прочтение “ки-логгер”)” или клавиатурные снифферы – это программное обеспечение либо аппаратное устройство для регистрации действий пользователя через устройства ввода. Обычно клавиатурные снифферы используются для шпионажа за поведением пользователя как клавиатурный шпион или вирусная программа, либо в целях администрирования и контроля за поведением пользователя в гос. учреждениях и корпорациях. Кейлоггеры позволяют регистрировать ввод пользователя с клавиатуры, а также логгировать манипуляции мышью и даже снимать скриншоты/скринкасты с экрана в некоторых случаях.

Представленная работа состоит из двух частей. Первая часть посвящена классификации и анализу существующих решений подобного рода по механизмам внедрения, особенностям реализации и работы, а также по сложности обнаружения и “лечения” системы после внедрения. Во второй части предлагается построение эффективной системы ввода данных, препятствующей достижению целей кейлоггинга для десктопных ОС Linux (с возможностью сборки под альтернативные целевые архитектуры, например, ARM) и Windows, а также для мобильной

платформы (Android), с указанием преимуществ и недостатков предлагаемого решения.

1 Обзор современных кейлоггеров

Клавиатурные шпионы можно разделить на 2 категории: программные и аппаратные.

1.1 Програмные кейлоггеры

Рассматривая программное обеспечение для регистрации действий пользователя, следует условно выделить 5 основных групп. Условность выделения заключается в существовании сложных решений, затрагивающих множество подсистем ПО. К примеру, в решениях на основе виртуализационных окружений существует несколько подтипов, относящихся скорее к модификации ядра гостевой (хостовой) ОС, чем к механизмам гипервизора. Также следует отметить, что механизмы колец защиты и виртуальной памяти, существующие в современных ОС, делают неэффективными и зачастую невозможными “наивные” низкоуровневые методы перехвата вроде подмены векторов прерываний и прямого чтения из буфера клавиатуры в ОЗУ. О таких методах кратко упомянуто в разделе “Аппаратные кейлоггеры”. С другой стороны, ряд механизмов позволяет осуществить перехваты в ядре или подложную «тонкую» виртуализацию атакуемой системы [3].

1.1.1 На основе виртуализации

1.1.1.1“Чистая” гипервизорная виртуализация

Подобные клавиатурные шпионы являются частью устанавливаемого ПО, работающего под управлением ОС, которая в случае “чистой” виртуализации остается нетронутой. Гостевая ОС не имеет сведений о перехвате виртуализированного устройства ввода, следовательно, поиск внедрения и возможных утечек информации из-под гостя не приводит к каким-либо результатам, ибо последние происходят на уровне хостовой ОС. Широко известна техника внедрения “Blue Pill”, основанная на аппаратной виртуализации с перехватом

запущенной хостовой ОС «тонким» гипервизором с последующей обработкой последним нужных операций ввода/вывода и системных вызовов.

Данные кейлоггеры фактически работают как компонент гипервизора, либо являются частью окружения ВМ.

Внедрение: вместе с устанавливаемым виртуализационным ПО.

Особенности: не затрагивает операционную систему. Данные виды кейлоггеров могут также работать с информацией с экрана и других устройств ввода, что делает небезопасными также и экранные клавиатуры. Техника “Blue Pill” применима лишь для систем с поддержкой процессором аппаратной виртуализации (AMD-V и Intel-VTx – инструкции), что ограничивает применение данного метода для ряда мобильных устройств (например, для некоторых ARMv7 без аппаратной виртуализации).

Обнаружение/лечение: часто сигнатуры подобных кейлоггеров могут не содержатся в базах антивирусных программ, за исключением некоторых вирусов. Неизвестные кейлоггеры требуют тщательного обследования системы в ручном режиме. Утверждается о 100% необнаруживаемости[4] руткита из перехваченной ОС: виртуализованный гость не способен определить свой статус.

1.1.1.2Паравиртуализация

В данном подклассе следует выделить:

- паравиртуализационные решения, где кейлоггер может быть введён в систему модификацией гостевой ОС, либо как в п.1.1.1.1.
- контейнерную виртуализацию, где кейлоггер может работать, в том числе, на хостовом ядре, подвергая результатам своей деятельности все запущенные контейнеры. К счастью, в большинстве случаев работа с контейнерами осуществляется через защищённое ssh-соединение, что тем

не менее не исключает утечки зашифрованного трафика.

Внедрение: вместе с устанавливаемым ПО.

Особенности: специфичность/зависимость от виртуализационного окружения. Многие промышленные контейнерные решения базируются на частично переработанных старых версиях ядра Линукс, что может лежать в основе альтернативных способов построения эффективных ядерных кейлоггеров, как в п. 1.1.2.

Обнаружение/лечение: методы, свойственные решениям на основе ядра и API операционных систем (группы решений 1.1. 2 и1.1. 3).

1.1.2 На основе ядра ОС

Работающие в пространстве ядра кейлоггеры отличаются уровнем внедрения и являются незаметными для пользовательских приложений. Такие кейлоггеры часто создаются как руткиты [5] и предназначаются для внедрения в ядро системы с целью получения доступа к системным вызовам и обработчикам аппаратных средств. Также могут быть созданы фильтр-драйверы клавиатуры, получающие доступ к любой информации, набранной на клавиатуре и переданной в операционную систему [6]. Помимо этого, подобные программы практически невозможно удалить, не имея прав администратора. В работе [2] описан ряд способов внедрения кейлоггера уровня ядра ОС Windows. Некоторые методы не применяются широко, ввиду сложности или неактуальности (например, обработка IRQ1 – PS/2 клавиатуры практически вышли из использования), однако теоретически реализуемы.

1.1.2.1Ядро Линукс

Ядро Линукс представляет собой монолитное ядро с поддержкой загружаемых модулей, поэтому наиболее очевидным способом реализации кейлоггера является

модуль ядра, перехватывающий или модифицирующий системную или драйверную функцию, либо системный вызов. В ходе анализа актуальных методов встраивания и перехвата, были выделены следующие способы внедрения кейлоггинга:

- Перехват функции ядра
- Перехват системного вызова
- Модификация таблицы системных вызовов
- Модификация диспетчера системных вызовов: пре- и постобработка контекста потока
- Подмена обработчика TTY-очереди

1.1.2.1.1. Перехват системных вызовов.

Схема инициирования системных вызовов для 32 и 64 битных ядер с поддержкой 32-битного режима представлена на рисунке 1.

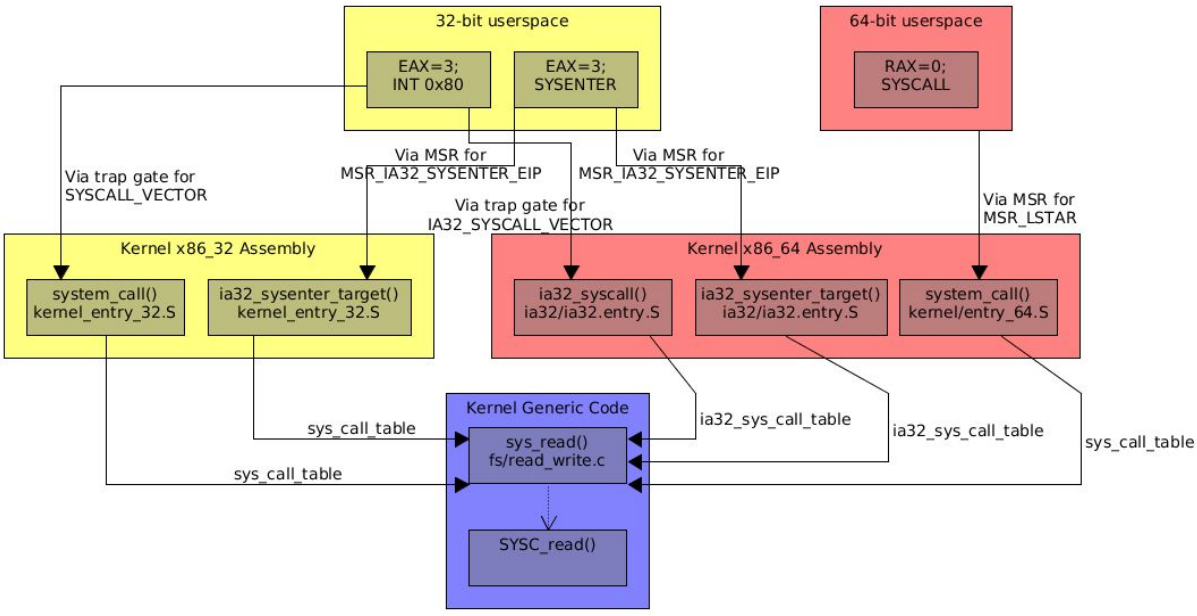

Рисунок 1 – Инициирование системных вызовов на 32 и 64-битной архитектуре

Из диаграммы видно, что для 64-битного ядра существует 2 таблицы системных вызовов: sys_call_table и ia32_sys_call_table, а диспатчинг производится

различными диспетчерами: “реальный” 64-битный режим SYSCALL, или эмуляции INT 80h, SYSENDER, либо SYSCALL32.

До версии ядра Линукс 2.5.41 наиболее очевидным способом перехвата вводимых символов был перехват системных вызовов Read/Write с последующей фильтрацией, однако такой метод неэффективен с точки зрения производительности: данные системные вызовы – основа операций ввода-вывода и наиболее часто используемые асинхронные операции, доступные из пространства пользователя. Для ядер версии выше 2.5.41 таблица системных вызовов явно не экспортируется, однако существует ряд методов поиска [9]. Опишем одну из схем внедрения (необходимы собранное ядро и исходники модуля кейлоггера):

1) найти адрес sys_call_table в /boot/System.map и передать его, например, в Makefile модуля кейлоггера:

```
#bash
grep sys_call_table /boot/System.map-$(uname -r) |awk '{print $1}' >>
kernlog/Makefile
```

2) для ядра версии выше 2.6 - получить возможность записи в область памяти с таблицей (по умолчанию - read-only):

   Пусть единичное значение нулевого бита CR0 данной архитектуры служит для включения “Protected Mode” (Intel), тогда для снятия блокировки записи, изменения таблицы и возвращения блокировки нужно выполнить действия:

```
//C
write_cr0(read_cr0() & (~0?10000));
/* Изменить таблицу */
write_cr0 (read_cr0() | 0?10000);
```

   Метод является SMP-неустойчивым: при разрешении записи в область векторов прерывания на SMP-системе возможны негативные последствия, вплоть до переноса процесса на соседнее ядро, поэтому авторы работы не рекомендуют использование метода в промышленных целях. Реализация в

виде функций опубликована на github-странице одного из авторов [17]. Существует альтернативный SMP-безопасный метод, заключающийся в маппинге посредством vmap() read-only страницы на страницу, запись на которой разрешена, с последующим вызовом stop_machine() - “метод временных отображений” [11, 18]. С реализацией данного способа разрешения записи можно ознакомиться на github-странице одного из авторов [18].

3) Остановимся подробнее на вопросе модификации диспетчера системных вызовов с пре- и постобработкой контекста потока. Данный способ внедрения является наиболее гибким по построению. Ниже приведен пример такой модификации (jmp на пре- и постобработчик, псевдокод):

```
system_call:
            swapgs
            ...
            jmp     service_inj
            mov     %rax,0x20(%rsp)
            ...
            sysret

service_inj:
            ...
            void KeylogTraceEnter(struct pt_regs *);
            ...
            call sys_call_table[N](args)
            ...
            void KeylogTraceLeave(struct pt_regs *);
            ...
             jmp back
```

Поиск адресов диспетчеров, исходя из диаграммы 1, осуществляется одним из 4 способов:

- INT 80h, чтение вектора таблицы IDT
- SYSENTER, чтение содержимого регистра MSR с номером MSR_IA32_SYSENTER_EIP
- SYSCALL32, чтение содержимого регистра MSR с номером

MSR_CSTAR

- SYSCALL, чтение содержимого регистра MSR с номером MSR_LSTAR

По найденным адресам по умолчанию можно только читать, что приводит к необходимости разрешения записи посредством 2.

1.1.2.1.2. Перехват системных вызовов через LSM

В ядра версий 2.5.41-2.6.4 интегрирован новый механизм LSM (Linux Security Modules), предоставляющий API для хуков системных вызовов, однако с 2.6.24 версии он не экспортируется, что заставляет прибегать к методам 1.1.2.1.1, 1.1.2.1.3.

1.1.2.1.3. Перехват обработчика TTY и драйвера устройства

Несмотря на гибкость вышеописанных перехватов 1.1.2.1.1 и 1.1.2.1.2, прямая реализация методов может иметь большие накладные расходы. Например, перехват всех READ/WRITE не является оптимальным решением, т.к. данные системные вызовы происходят очень часто :на них организуется весь файловый (а значит, и пайповый, сокетный) ввод-вывод.

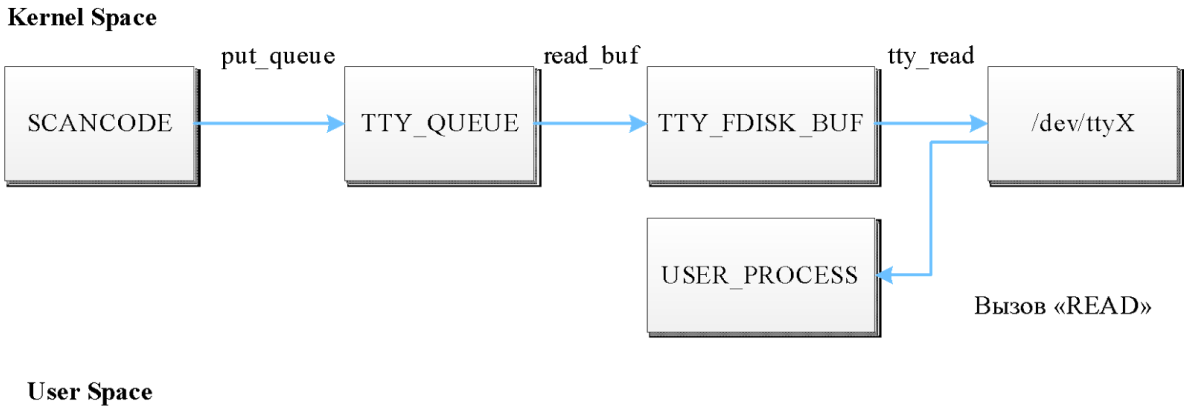

Рисунок 2 – Последовательность ввода с клавиатуры в Linux

На рисунке 2 схематично изображена последовательность клавиатурного ввода в Linux, от считывания скан-кода символа до передачи в пользовательский процесс. Отсюда следует вывод о 4-х возможных местах перехвата:

1) Функция handle_scancode() драйвера клавиатуры-для перехвата «сырых» сканкодов.
2) Функция put_queqe() подсистемы ввода для перехвата символов, вставляемых в TTY-очередь.
3) Функция receive_buf() для перехвата передачи из TTY-очереди в буфер логического TTY-устройства.
4) Функция tty_read() для связывания TTY с VFS (/dev/ttyX).

Методы 1-2 являются платформо-зависимыми, ввиду перехвата драйверных функций, однако позволяют осуществлять перехват всей введенной информации, вне зависимости от назначения, будь то консоль или ввод для X-сервера, что является преимуществом. Для реализации перехвата данных функций можно использовать широко известный метод перехвата Сильвио [10,12], или метод временных отображений [11, 16]:

1.1.2.1.4. Собственный обработчик прерываний от клавиатуры

Альтернативным и наиболее прямым способом перехвата является перехват соответствующего прерывания с последующим чтением портов 0x60 (data) и 0x64 (status). Данный механизм не является достаточно унифицированным, ввиду аппаратной зависимости и предварительной неопределённости источника ввода. Одним из возможных способов решения является мониторинг /proc/interrupts с целью получения текущих обработчиков соответствующих номеров прерываний, анализа и установки своих.

1.1.2.1.5. Выводы: ядро Linux

Проведённый обзор методов кейлоггинга в ядре Линукс позволяет сделать вывод о возможности построения эффективных средств сбора информации на уровне ядра, однако необходим root-доступ и, возможно, пересборка ядра по

частично отредактированной конфигурации. Минусом также является необходимость пересборки модуля ядра под конкретную ОС.

1.1.2.2Ядро Windows

Укажем несколько базовых и официально документированных способов построения ядерных кейлоггеров Windows, после чего продемонстрируем альтернативные способы (рис. 3).

1.1.2.2.1. Фильтр-драйвер драйвера класса KbdClass

Кейлоггер перехватывает запросы IRP_MJ_READ установкой фильтр-драйвера поверх устройств \Device\KeyboardClass0 (KbdClass-драйвер). Такая фильтрация позволяет отследить коды нажатых и отпущенных клавиш. Гарантируется перехват всех нажатий и невозможность обнаружения из пространства пользователя, однако драйвер легко обнаруживается штатными системными средствами. Метод официально документирован в DDK[13].

1.1.2.2.2. Драйвер фильтр драйвера i8042prt

Устаревший метод, основанный на обработке прерывания IRQ1. Работает только с PS/2 – клавиатурой, что является недостатком в наше время. Метод официально документирован в DDK.

1.1.2.2.3. Модификация таблицы обработки системных запросов KbdClass-драйвера

Метод основан на подмене точки входа IRP_MJ_READ в таблице диспатчинга драйвера, либо на перехвате IRP_MJ_DEVICECONTROL. Методу присущи свойства 1.1.2.2.1 и 1.1.2.2.2.

1.1.2.2.4. Модификация таблицы KeServiceDescriptorTableShadow

Посредством изменения точки входа NtUserGetMessage/PeekMessage в таблице

системных сервисов KeServiceDescriptorShadow (win32k.sys) можно получить информацию о нажатии клавиши перехватчиком после завершения GetMessage или PeekMessage в каком-либо из пользовательских потоков. Метод сложен в реализации, однако эффективен, ввиду сложности обнаружения. Для реализации метода необходимо найти адрес таблицы системных сервисов. На данный момент это наиболее часто встречающийся в литературе и наиболее эффективный способ ядерного кейлоггинга.

1.1.2.2.5. Модификация кода NtUserGetMessage/PeekMessage

Непосредственная модификация функций NtUserGetMessage/PeekMessage – относительно сложный и эффективный метод перехвата. Модификацию можно осуществлять, к примеру, методом сплайсинга, модифицируя код «на лету».

1.1.2.2.6. Подмена KbdClass – драйвера

Прямая подмена драйвера в стеке драйверов представляется наиболее очевидным способом перехвата клавиатуры, однако возникают сложности связанные с определением клавиатуры, используемой в системе, характерные для метода 1.1.2.1.4 Linux.

1.1.2.2.7. Обработчик прерывания

Метод аналогичен методу 1.1.2.1.4 Linux, с присущими последнему аппаратно-зависимостью и сложностью прямого взаимодействия с конкретным обработчиком прерывания.

1.1.2.2.8. Выводы: Ядро Windows

Следует отметить, что кейлоггеры на основе методов 1.1.2.2.1-1.1.2.2.4 детектируются системой проактивной защиты Kaspersky Internet Security, что ставит под сомнение возможность злоумышленного внедрения данных решений. Методы 1.1.2.2.6 и 1.1.2.2.7 сложны и, по мнению авторов доклада, не актуальны для

использования на практике.

1.1.2.3Общие выводы по разделу

Внедрение: как руткит, может иметь вид модуля ядра Linux, драйвера устройства ввода/фильтр-драйвера для Windows. Однако прямая подстановка драйвера относительно легко определима, поэтому на практике применяются более сложные механизмы. Основной метод внедрения - подмена записей в таблице системных вызовов, подмена или модификация самих системных вызовов. В Windows - таблицы системных сервисов KeServiceDescriptorTableShadow. В Linux задача осложняется тем, что с версии ядра 2.6 таблица не экспортируется. Альтернативный вариант поиску таблицы - использование VFS.

Особенности: получает прямой доступ к аппаратным средствам/вызовам/драйверам ввода в случае фильтрующего типа [5]. Может иметь интерфейс, передающий функциональность в пространство пользователя (IOCTL и др).

Обнаружение/лечение: трудно обнаруживается пользовательским ПО и в некоторых случаях не может быть обнаружен и удален без администраторских прав/отключения модуля ядра/пересборки ядра.

1.1.3 На основе API ОС (в пространстве пользователя)

Программы на основе API зачастую [2] пользуются методом, называемым хукинг [3, 5 - 6] или перехват сообщений специального вида, который позволяет регистрировать события клавиатуры в системе. Хукинг внутренних функций и системных вызовов ядер был описан в разделе 1.1.2. В пространстве пользователя также существует возможность перехвата необходимых кейлоггеру событий.

Windows

В Windows API подобный перехват может осуществляться через функции

GetAsyncKeyState(), GetKeyState(), GetForegroundWindow(), либо прямым хукингом процесса csrss.exe [7], в котором находится очередь аппаратного ввода, а также методами из работы [2]. Перехват событий Windows кратко описан в разделе 1.3 “Реализация Кейлоггера под управлением JVM на языке JAVA”.

Linux

В Linux наиболее распространённым методом является перехват событий X-сервера с использованием соответствующего API библиотеки Xlib (XCB). Как альтернатива - ptrace(...)[8] к процессу, контекст которого перехватывается. Также существует множество способов перехвата выставлением переменной окружения LD_PRELOAD и подменой библиотечных вызовов. В случае, если права пользователя, под которым запускается кейлоггер, позволяют читать из соответствующего символьного устройства, можно ограничиться определением устройства ввода из доступных в /dev/input/ с помощью IOCTL (EVIOCGNAME) и последующим периодическим опросом. Хукинг X-сервера кратко описан в разделе “Реализация Кейлоггера средствами JVM на языке JAVA” данной работы.

Внедрение: запуск программы на целевом компьютере с соответствующими правами.

Особенности: использует API операционной системы и ее внутренние события.

Обнаружение/лечение: методы мониторинга и защиты критических компонент пространства пользователя [3, 7 – 8].

В качестве примеров внедрения приложена схема обработки ввода ОС Windows [2] (на схеме показано подключение клавиатуры через PS/2, с целью покрытия методов 1.1.2.2.2 и 1.1.2.2.6).

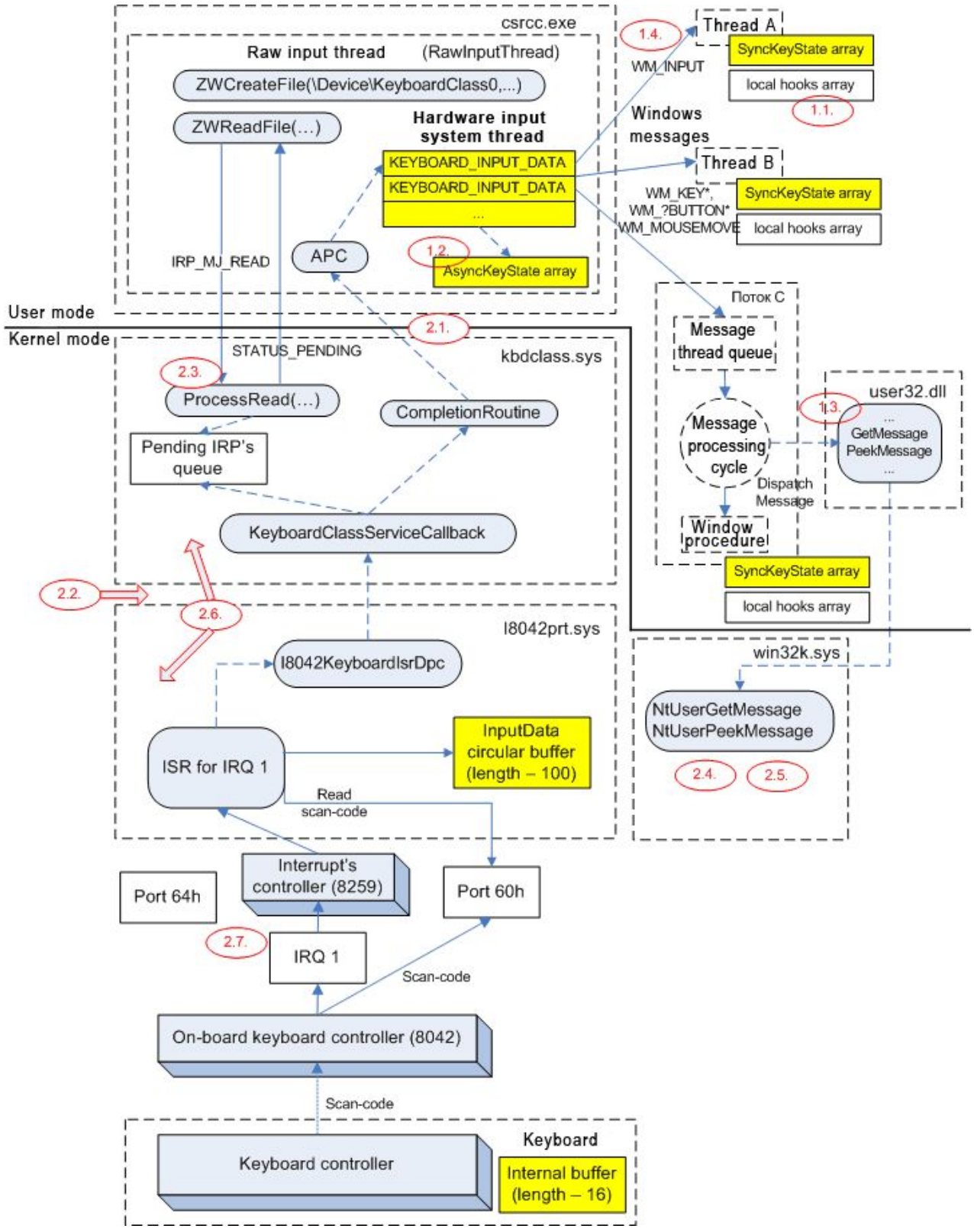

Рисунок 3 – Обработка ввода ОС Windows. Красным обозначены участки возможного внедрения. 1.X - пространство пользователя. 2.X - пространство ядра [2]

1.1.3.1Анализаторы пакетов и перехватчики данных с веб-форм

Данные виды клавиатурных снифферов позволяют перехватывать HTTP POST запросы с целью извлечения незашифрованных паролей, либо перехватывать submit событий веб-форм.

Внедрение: как часть программного обеспечения, либо как внедоносный javascript или надстройки над браузерами.

Особенности: исполнение на клиенте в контексте браузера, эксплуатация его возможностей.

Обнаружение/лечение: вредоносный js-код можно обнаружить при помощи средств веб-разработчика в браузере.

1.1.3.2На основе инъекции в память

(MitB)-based кейлоггеры выполняют свою функцию за счет изменения таблицы памяти, связанной с браузером или другими системными функциями [7]. Этот метод осуществляется при помощи внесения изменений в таблицы памяти или инъекции непосредственно в память, что может быть использовано вредоносным ПО для обхода системы контроля аккаунтов Windows.

Внедрение: вместе с троянами и другим вредоносным ПО.

Особенности: подменяет содержимое таблиц памяти.

Обнаружение/лечение: при помощи антивирусных программ и сканеров.

Также следует отметить, что некоторые кейлоггеры могут быть оснащены механизмами обратной связи и удаленного управления, что позволяет им администрироваться по сети и пересылать собранную информацию через http или ftp протокол, на удалённый сервер или на почту. Также информация может передаваться посредством беспроводной сети при помощи установленной аппаратной системы Bluetooth, Wi-Fi, WiMAX или CSD, GPRS, EDGE, EV-DO,

HSPA.

1.1.4 Общие выводы

Приведённый выше обзор мотивирует к разработке пассивных методов защиты, устойчивых к перехвату ввода на любом уровне. Естественной является идея построения преобразующего антикейлоггера, так, чтобы финальная последовательность символов имела смысл лишь на этапе работы с целевым приложением. Идея ввода символов без нажатий на клавиши также представляет интерес. Описание разработанных программных методов защиты приведено в разделе “Практические предложения” данной работы.

1.2 Аппаратные кейлоггеры

Аппаратные логгеры не зависят от программного обеспечения и используют аппаратные средства компьютерных систем.

1.2.1 В качестве прошивки и микропрограмм

Данные логгеры позволяют считывать информацию на уровне BIOS. При этом аппаратный кейлоггер должен иметь функциональные коды программных клавиатурных шпионов.

Внедрение: как прошивка для BIOS или других микроконтроллеров.

Особенности: малый размер, труднообнаружимы. Эксплуатация обработки прерывания 9h или 15h,16h. Прямой доступ к физической памяти (эксплуатация адресов 0:041A — 0:42C).

Обнаружение/лечение: диагностика/перепрошивка контроллеров.

1.2.2 На основании встроенных элементов

В качестве данного кейлоггера может выступать серия устройств,

размещенных между клавиатурой и компьютером в местах разъема кабеля.

Внедрение: в качестве внешнего устройства.

Особенности: может выступать в роли переходника клавиатуры между разъемами разных типов PS/2, USB, COM.

Обнаружение/лечение: путем внешнего осмотра и обследования используемых переходников.

1.2.3 Снифферы беспроводных устройств

Данные снифферы позволяют собирать пакеты данных, передаваемые беспроводными устройствами. Подобные радиоклавиатуры могут быть взломаны т.к. либо не используют шифрование для ускорения обмена данными, либо используют однобайтовый шифровальный ключ. Сложнее дело обстоит с bluetooth-клавиатурами т.к. обмен шифровальными ключами таких клавиатур происходит лишь в момент подключения устройства.

Внедрение: перехват осуществляется посредством радиоантенны на расстоянии до 40 м. на частоте 27 Гц [20]

Особенности: Для восстановления шифровального ключа требуется около 30 нажатий [20].

Обнаружение/лечение: не использовать радиоклавиатуру в общественных местах.

1.2.4 Накладные клавиатурные кнопки

Некоторые злоумышленники используют для достижения своих целей накладные клавиатурные кнопки, размещенные поверх настоящих. Широко известны случаи перехвата PIN-кодов банковских карт при помощи таких клавиш в банкоматах.

Внедрение: физический контакт с устройством.

Особенности: отдельное независимое устройство

Обнаружение/лечение: осмотр рабочего места / клавиатуры.

1.2.5 Акустический кейлоггер

Акустический криптоанализ может быть использован для мониторинга звуков, созданных во время введения символов с клавиатуры[22]. Каждая клавиша на клавиатуре может делать различные тональные сигналы, позволяя при этом считать введенные символы используя статистические и частотные методы анализа.

Такие устройства способны реагировать на изменение акустических тонов клавиш, определять время между нажатиями клавиш на клавиатуре и записывая иную контекстную информацию, которая может быть использована для определения языка. Единственный недостаток такого метода считывания данных – большой объем записи (как минимум 1000 символов) необходимый для корректного распознавания и расшифровки данных.

Внедрение: посредством “жучков” или в зоне слышимости.

Особенности: не требует взаимодействия со сканируемой системой.

Обнаружение/лечение: невозможно

1.2.6 Использование электромагнитных излучений

При помощи аппаратных клавиатурных шпионов можно перехватить электромагнитные излучения, исходящие от клавиатуры на расстоянии до 20 метров без необходимого подключения дополнительных устройств к клавиатуре. Недостаток использования – дорогостоящее оборудование.

Внедрение: посредством специализированного оборудования

Особенности: не требует взаимодействия со сканируемой системой.

Обнаружение/лечение: невозможно

1.2.7 Оптическое наблюдение

Хотя в классическом понимании этот способ не является кейлоггингом, он все же используется для перехвата введенных паролей и PIN кодов. Возможны также экзотические случаи использования тепловизора/ИК-сканера высокого температурного разрешения (~0,05K) для получения температурной карты клавиатуры [10] после введения информации. Разность нагрева поверхности клавиш даёт потенциальную возможность восстановить введённую последовательность символов.

Внедрение: стратегически правильно расположенная веб-камера, скрытая камера или другое устройство видеонаблюдения.

Особенности: по сути является самым простым и наименее криминальным методом слежения.

Обнаружение/лечение: осмотр потенциально опасных камер наблюдение и слепой ввод с закрытой чем-либо клавиатурой.

1.2.8 На основе вещественных доказательств

Данный способ применяется для перехвата паролей и кодов преимущественно введенных с цифровой клавиатуры. При вводе кода на клавиатуре остаются отпечатки пальцев, позволяющие, зная длину кода и использованные символы, подобрать пароль полным перебором всех комбинаций(brute force attack).

Внедрение: прямой физический контакт.

Особенности: требует доступ к устройству ввода для снятия вещественных доказательств.

Обнаружение/лечение: использование при вводе перчаток.

1.2.9 Перехват с сенсоров смартфонов

Исследователи [19] продемонстрировали возможность перехвата ввода с клавиатуры посредством акселерометра смартфона находящегося рядом на столе с анализируемым устройством. Акселерометр смартфона может идентифицировать вибрации от нажатия клавиш при печати и, анализируя полученный с акселерометра сигнал, преобразовать их в читаемый текст с 80% точностью. Принцип основан на анализе событий типа “нажатие пары клавиш” и сопоставления их с раскладкой клавиатуры.

Внедрение: путем размещения на столе с машиной смартфона с акселерометром

Особенности: приближенное распознание текста, отталкиваясь от манеры печати пользователя.

Обнаружение/лечение: осмотр рабочего места.

1.3 Реализация Кейлоггера под управлением JVM на языке JAVA

В языке JAVA мониторинг событий с клавиатуры и мыши производится при помощи имплементации интерфейсов awt.event.KeyListener и awt.event.MouseListener. Однако такой подход не позволяет отлавливать глобальные события клавиатуры вне java приложения. В случаях когда нужно отлавливать глобальные события клавиатуры и мыши, либо отлавливать события не используя элементы графического интерфейса, на помощь приходит JNI и библиотека JNativeHook.

JNI(Java Native Interface) - стандартный механизм для вызова из под управления JVM кода, написанного на C/C++ или Ассемблере и скомпонованного в виде динамических библиотек, позволяет не использовать статическое связывание, даёт возможность вызова функций C/C++ из программы JAVA и наоборот.

JNativeHook - библиотека, позволяющая слушать глобальные события клавиатуры и мыши на Java, что невозможно обычными средствами. Используя Java Native Interface, она получает доступ к глобальным событиям клавиатуры посредством перехвата низкоуровневых событий и доставки их в Java-приложение, однако имеет платформозависимую реализацию. Скачать библиотеку и прочитать дополнительную информацию можно по ссылке [14].

Библиотека позволяет перехватывать следующие виды событий:

- Key Press Events
- Key Release Events
- Key Typed Events
- Mouse Down Events
- Mouse Up Events
- Mouse Click Events
- Mouse Move Events
- Mouse Drag Events
- Mouse Wheel Events

В дополнение к глобальным событиям, она позволяет получать и системные свойства, предоставляемые нативной библиотекой. Однако получение этих свойств может не гарантироваться на некоторых архитектурах.

- jnativehook.key.repeat.rate
- jnativehook.key.repeat.delay
- jnativehook.button.multiclick.iterval

- jnativehook.pointer.sensitivity
- jnativehook.pointer.acceleration.multiplier
- jnativehook.pointer.acceleration.threshold

Для корректной работы библиотеки система должна соответствовать следующим требованиям (см. таблицу 1).

Таблица 1 – Требования и зависимости решения на Java

System	Apple	Windows	Linux
1) RAM - 256 MB 2) Java 1.5 - 1.8	1) OS X 10.5 - 10.10 2) i586, amd64 3) Enable Access for Assistive Devices	1) Windows 2000-10 2) i586, amd64	1) X11 2) i586, amd64, arm6j, armv7 3) libX11.so.6, libXt.so.6, libXtst.so.6, libXext.so.6, libXdmcp.so.6, libXau.so.6, libICE.so.6, libSM.so.6, libxcb.so.1, Libc.so.6, Libdl.so.2, libuuid.so.1, libXinerama.so.1

Перехват глобальных событий мыши при помощи этой библиотеки реализуется посредством имплементации интерфейса org.jnativehook.keyboard.NativeKeyListener и регистрации его в глобальном мониторе org.jnativehook.GlobalScreen.

Простейший способ перехвата событий клавиатуры в JAVA можно осуществить даже в консольном приложении. Для этого достаточно зарегистрировать NativeHook при помощи вызова GlobalScreen.unregisterNativeHook();

Затем при помощи вызова метода GlobalScreen.addNativeKeyListener зарегистрировать реализацию интерфейса NativeKeyListener как слушателя клавиатуры. При реализации в связке со Swing, в силу того, что его копоненты потоконебезопасны, следует использовать GlobalScreen.setEventDispatcher(new SwingDispatchService());

1.3.1 JavaScript

В JavaScript можно “слушать” события клавиатуры в браузере, назначая слушатели событий keypress, keyup, keydown. Для “прослушивания” событий мыши можно использовать слушатели “onmousedown”, “onmousemove” для определения событий клика и положения курсора мыши. Определение кода нажатой клавиши происходит по event.which или event.keyCode в зависимости от используемого браузера.

Отслеживание событий клавиатуры в браузере может быть использовано для перехвата вводимой на сайте информации или логгирования поведения пользователя, например, для бот-контроля деятельности посетителей.

Создание подобных скриптов существенно зависимо от версии используемого браузера и операционной системы, однако, для упрощения процесса разработки, можно использовать несколько библиотек для работы с клавиатурными событиями: JavaScript Shortcuts Library, OpenJS, Keycode, Keypress и др.

Также следует упомянуть о разработке исследователей колумбийского университета [21], которые смогли восстановить информацию о нажатых клавишах и кликах мышью в веб-браузере по содержимому кэш-памяти центрального процессора компьютера. По их сведениям, эксплойт эффективен на компьютерах с новыми моделями процессоров производства Intel, такими как Core i7, в браузерах, поддерживающих HTML5, что составляет около 80% всех компьютеров. Суть метода заключается в измерении времени, требуемого для доступа к кэш-памяти последнего уровня (кэш L3 разделяется всеми ядрами процессора), и сравнении этого времени с действиями пользователя.

2 Практические предложения

В результате проведенного выше обзора было принято решение разработать методы позволящие совершать ввод текста стандартными средствами и с обходом угроз, создаваемых некоторыми из описанных выше кейлоггеров. Рассматривая создаваемые существующими кейлоггерами угрозы, было принято решение выделить следующую модель угроз для разработки методов и прототипов для защиты от наиболее часто встречаемых кейлоггеров.

1.4 Модель угроз

Раздел “модель угроз” описывает возможные сценарии угроз информационных систем, реализованных в процессе исследования, а также возможные и реализованные способы их ликвидации.

1.4.1 Описание информационной системы

Таблица 2 – ССВ-1

Идентификатор	**Описание**	**Примечание**
АН-1	Анализатор ввода с устройства	Приводит события с ввода к внутренним событиям системы
РАСП	Распознаватель символа для списка событий ввода	Приводит список внутренних событий в однозначное соотношение с символом
СЛОВ-Ф	Словарь ввода. Файл, который содержит модель соотношения событий ввода и символов	Файловое представление словаря
СЛОВ-М	Словарь ввода. Модель представления словаря, загруженная в память приложения	Модель словаря в памяти
ТР	Транспортный модуль	Модуль передачи расшифрованного текста во внешний источник

Таблица 3 – СДШВ-1

Идентификатор	Описание	Примечание
ШВ	Шифрованный ввод	Представляет из себя нажимаемые на физической клавиатуры клавиши. Фактический ввод
ДШВ	Дешифрованный ввод	Закодированное сообщение или то, что подразумевалось для ввода. Получается путем преобразования ШВ внутри программы
ДШК-Ф	Файл с динамическим шифровальным ключом	Представляет из себя файл со списком раскладок ввода в JSON формате. Каждая раскладка ставится в соответствие с итерацией ввода
ДШК-Г	Сгенерированный шифровальный ключ в памяти	Модель раскладки в памяти процесса
СВ	Сессионный ввод на основе файлов ключей	Шифрование происходит на основе файла ключа .key и графического представления ключа, например, в pdf
РВ	Ввод в реальном времени со случайной генерацией раскладок	Генерация раскладки в реальном времени и выведения ее на экран
ТАЙМ	Таймер. Точка отсчета и длительность итераций	Момент запуска приложения и длительности итерации

1.4.2 Угрозы безопасности

Таблица 4 – ССВ-1

Идентификатор угрозы	Угроза	Затрагиваемые части	Результат реализации
У1.1	Перехват и логирование поведения мыши/тачпада.	АН-1, доступ к мыши	Само по себе без реализации других угроз не является достаточным для дешифрования текста
У1.2	Перехват скриншотов экрана	Доступ к экрану	Не позволяет перехватить текст
У1.3	Восстановление словаря на основе анализа	Злоумышленник должен знать	Позволяет восстановить словарь и

	вводимого текста. Требует реализации угрозы У1.1 и знания вводимого текста для восстановления словаря.	достаточное кол-во введенного контрольного текста или быть в сговоре с ОС	перехватывать вводимый далее текст
У1.4	Получение файла словаря и распознавания вводимых символов. Требует реализации угрозы У1.1 и доступа злоумышленника к ФС	АН-1, РАСП, СЛОВ-Ф	Позволяет полностью однозначно восстанавливать текст
У1.5	Получение модели словаря из памяти программы, например, при помощи получение дампа JVM	АН1, РАСП, СЛОВ-М	Позволяет полностью однозначно восстанавливать текст
У1.6	Перехват расшифрованного символа во время транспортировки во внешний модуль	ТР	Получает результат расшифровки

Таблица 5 – СДШВ-1

Идентификатор угрозы	**Угроза**	**Затрагиваемые части**	**Результат реализации**
У2.1	Перехват и логгирование ввода с клавиатуры	Доступ к клавиатуре	Перехватывает фактически ввод, но не шифрованное сообщение
У2.2	Перехват скриншотов экрана	ДШК-Г, РВ, ТАЙМ	Система уязвима к данной угрозе в режиме генерации ключа в реальном времени т.к. На экране отображается текущая раскладка
У2.3	Перехват PDF файла со словарем	СЛОВ	Получение словаря позволяет узнать список используемых при вводе раскладок, но для однозначного определения текста нужно знать текущую итерацию. Угроза может быть критичной при комбинированном использовании с У2.1 и У2.2.
У2.4	Перехват и распознавание	СЛОВ	То же, что для У2.3

	словаря из .key файла		
У2.5	Получение расшифрованного сообщения из памяти процесса, посредством дампа JVM	СЛОВ, ДШВ	Непосредственный перехват результата шифрования
У2.6	Перехват расшифрованного символа во время транспортировки во внешний модуль	ТР	Получает результат расшифровки

Таблица 6 – Заключение

Угроза	**Способы мер защиты**	**Реализовано/ Возможно**
У1.1	Исключение возможности перехвата событий мыши программными и аппаратными средствами Рассматривается как неизбежная угроза	-
У1.2	Исключение возможности перехватов содержимого экрана Рассматривается как неизбежная угроза	-
У1.3	Исключить вероятность сговора с ОСВ или получение контрольного текста достаточной величины. Является внешней угрозой и должно устраняться на уровне корпоративной политики	-
У1.4	Шифрование файла словаря при помощи AES шифрования	Реализовано
У1.5	Размещение модели словаря вне кучи JVM посредством unsafe	Реализовано
У1.6	Шифрование ТР посредством RSA шифрования или использование шифрованных протоколов	Реализовано
У2.1	Исключение наличия кейлоггеров Рассматривается как неизбежная угроза	-
У2.2	Исключение возможности перехватов содержимого экрана. В случае если это является неизбежной угрозой, то рекомендуется использовать ввод на основе шифрования ключами .key файл и PDF	-
У2.3	Генерировать .key и хранить PDF на защищенном устройстве или распечатать на бумаге Является внешней угрозой и должно устраняться на уровне корпоративной политики	-

У2.4	Хранение файла в зашифрованном AES алгоритмом виде	Реализовано
У2.5	Хранение ДШВ вне кучи JVM при помощи unsafe метода	Реализовано
У2.6	Шифрование ТР посредством RSA шифрования или использование шифрованных протоколов	Реализовано

1.5 Предлагаемые решения

1.5.1 ССВ - Система стенографического ввода

Разработанный авторами метод стенографического ввода позволяет совершать ввод на основе таблицы последовательностей 1-4 движений мыши по 4 направлениям в символы латинской клавиатуры + спец. символы.

Описание процесса преобразования текста

Пользуясь следующим списком событий, система распознает вводимый символ и инициирует отправление характеризующего символ списка событий в модуль анализатора:

- Касание (клик кнопки мыши).
- Черта (задается единичным вектором в одном из 4 направлений).
- Ожидание (временная пауза-разделитель между чертами; производится с нажатой клавишей).
- Завершение символа (событие происходит при отпускании нажатой клавиши или завершения касания сенсорного экрана).

Диаграмма процесса распознавания символа ССВ-ДИАГ-2 представлена на рисунке 4.

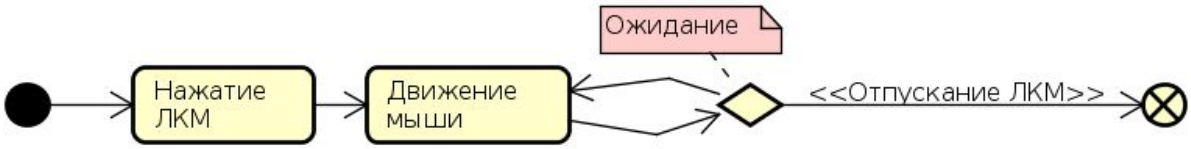

Рисунок 4 – Диаграмма процесса распознавания символа ССВ-ДИАГ-2

Создание списка событий

- Система слушает события с тачпада или манипулятора-мышь на наличие событий клика ЛКМ или касания сенсорного экрана. Появление такого события инициирует создание нового списка событий для распознавания символа
- Следующее событие обязательное для кодирования символа это черта - движение мышью с нажатой кнопкой мыши (выполняется не отпуская ЛКМ после предыдущего события) или движение по сенсорному экрану после касания. Для кодирования события составляется характеризующий его вектор для определения, которого берутся точки начала и конца траектории движения.

Черта может иметь одно из 4 направлений, к которым полученный вектор приводится округлением до ближайшего направления.

↑ (-44 - 44)° ↓ (135 - 224)°

→(45 - 134)° ←(225 - 314)°

- Событие может выполняться несколько раз, разделяемых событием ожидание. Событие ожидание характеризуется промежутком времени Δt, отсутствие движения, в течение которого инициирует событие Ожидание.
- Завершение символа. Данное событие инициируется отпусканием кнопки мыши или завершением касания сенсорного экрана.

Работа анализатора

Получая на вход список событий, анализатор проверяет его при помощи словаря стенографического ввода и ставит в соответствие списку событий: символ, функциональный символ, либо выдает сообщение о невозможности классификации.

Дополнительный варианты реализации (являются предложениями вариантов реализации прототипа и его усовершенствования. Не являются необходимыми для демонстрации метода и не реализовано в данной работе):

- Реализация распознавателя на основе изломов траектории без таймаута и задержек. В данном методе рассматривается траектория целиком и без задержек между чертами символа. Данный метод призван ускорить ввод символов 1 непрерывным движением без задержек.
- Реализация ввода на основе композитных символов. Совершенствуя метод можно составить словарь для символов состоящих из нескольких частей, разделенных нажатием/отпусканием кнопки мыши. Например символ похожий на i, j.

Анализ безопасности метода

Данный метод использует события манипулятора-мышь или касания сенсорного экрана, поэтому само поведение пользователя с устройством ввода может быть перехвачено, но без знания списка событий, алгоритма их распознавания и словаря, расшифровка может быть затруднительна.

Рассматривая реализации с анализатором на стороне клиента:

- В данном случае можно перехватить поведение мыши, но без знания алгоритма получения из них списка событий и словаря, данные расшифровать невозможно. Т.к. Программа выполняется локально, можно декомпилировать ее и выделить алгоритмы преобразований к списку событий и доступы к словарю.

- Перехват скриншотов экрана не позволяет вычислить вводимый текст.
- Для перехвата информации при стенографическом вводе может быть предпринята попытка анализа перехваченного поведения мыши и восстановление алгоритмов формирования событий и словаря, однако для такого рода атаки требуется чтобы пользователь совершил ввод некоторого заранее известного текста для расшифровки метода (длина текста требует оценки).

Рассматривая реализацию с анализатором на стороне сервера (клиент-серверная версия пример Web-front-side – Web-server-side), злоумышленники могут перехватить лишь поведение мыши, но доступа к исходникам и словарю получить невозможно. В таком случае для перехвата может быть реализован лишь сценарий с анализом поведения мыши для восстановления алгоритмов и словаря (см. выше).

1.5.2 СДШВ - Система динамически шифруемого ввода

Разработанный авторами на Java прототип представляет собой систему динамической генерации раскладки клавиатуры, в соответствии с которой вводимые с клавиатуры символы переводятся в конечный текст. Зашифрованная раскладка передаётся на целевое, либо стороннее устройство и отображается на экране в виде формы/html-файла, либо выводится на печать.

Принцип работы приложения:

1) Пользователь генерирует сессию шифрованного ввода.
   Генерация сессии подразумевает создание динамического шифровального ключа с раскладками клавиатуры и временем их активности относительно момента старта шифрованного ввода.
2) Шифрованная сессия сохраняется в защищенное локальное хранилище (для клиентских версий) или сохраняется на сервере для клиент-серверной

реализации.

3) Пользователю высылается документ с раскладками и таймингов в формате pdf. Например на почту (или другой защищенный от перехвата кейлоггером источник по выбору). В клиент-серверной реализации так же высылается ID сессии и ключ доступа к ее старту.

4) Пользователь выбирает сгенерированную сессию шифрованного ввода и запускает ее. При этом на экране отображается поле ввода и таймер, позволяющий, сверяясь с присланным ключом вводить информацию.

5) Шифрованный ввод с клавиатуры направляется в приложение (без преобразований) и там приводится к оригинальному тексту на основе времени нажатия каждой клавиши и шифровального ключа с раскладками и таймингом.

6) (*дополнительно) Десктопные реализации предполагают исследование на возможность инъекции результата расшифровки напрямую в поле ввода с фокусом для любого приложения. Такое исследование было проведено на примере инъекции в графическую систему X Window System.

Дополнительный варианты реализации (являются альтернативными предложениями вариантов реализации прототипа и его усовершенствования. Не являются необходимыми для демонстрации метода и не реализованы в данной работе):

- Отправка раскладки на email(как локально, так и с помощью специального сервера). Позволит открыть на другом устройстве без формирования файла на основном.
- Embedded-решение. Создание физического устройства-переходника для клавиатуры на базе простого одноплатного компьютера или FPGA, который заменит файл с раскладкой. Пользователю не нужно будет искать

символы в раскладке. При вводе обычного текста с клавиатуры, он будет преобразовываться к зашифрованному тексту, согласно динамическому шифровальному ключу. Для компьютера это будет выглядеть аналогично вводу пользователем зашифрованного текста, далее программа будет расшифровывать его, как в данном прототипе.

Анализ безопасности метода

Ввод осуществляется посредством клавиатуры, потому все нажатые клавиши могут быть перехвачены кейлоггерами любого вида, что однако еще недостаточно для восстановления оригинального ввода. Для полного восстановления оригинального текста требуется 3 объекта:

1) Зашифрованный ввод с клавиатуры(может быть перехвачен)
2) Файл с динамическим шифровальным ключом. Файл сохраняется в зашифрованном виде и для его расшифровки нужен RSA ключ. Либо должен быть перехвачен pdf/html файл с кодировками для ручного анализа поведения.
3) Точка отсчета. Момент, с которого запускается таймер для определения позиции в динамическом ключе шифрования. Для его перехвата злоумышленник должен однозначно определить момент запуска шифрованного ввода. В случае если этот момент не может быть перехвачен, расшифровать сообщение невозможно.

В случае с клиент-серверной версией декомпиляция или получение дампов памяти невозможно, т.к. на сервер отправляется лишь шифрованный ввод (как нажимались клавиши и время их нажатия), а результат расшифровки может быть перенаправлен с сервера в другой источник (пользователь получает лишь сообщение об удачной расшифровке), для такого случая перехват сообщения становится невозможен. В данной работе клиент серверная реализация приводится лишь в виде

обзора и не является обязательно для реализации.

Примечание: во всех случаях предполагается, что пользователь выбирает безопасный способ получения раскладок клавиатуры и на устройстве ввода его перехватить невозможно (открывает на другом устройстве или распечатывает письмо на надежном принтере).

1.5.3 Общие положения и особенности реализации

1.5.3.1Проблема хранения конфиденциальных данных в памяти JVM

В ходе работы предлагаемых прототипов возникает ситуация, когда возможна утечка информации через дампы JVM. Это связано с тем, что String в JAVA является immutable(неизменяемая) и final(финализированной), и хранится в пуле строк. Пулом строк называется набор строк, что хранится в памяти Java Heap. Из-за этого строка будет храниться там с момента создания и до тех пор, пока не будет удалена сборщиком мусора. Поэтому она может быть еще некоторое время доступна для похищения путем получения дампа JVM. Хотя пул строк содержит большое количество строк, и какая из них является декодированным сообщением не очевидно, вероятность отсеивания лишних строк и похищения сообщения остается.

В подобных случаях предпочтительнее использовать массив символов char[] вместо String, т.к. массив символов можно очистить после использования, но это не исключает похищения данных в случае своевременно сделанного дампа JVM. Либо если дампы памяти будут делаться достаточно часто. Для решения этой проблемы можно использовать прямую работу с памятью при помощи утилитарного класса sun.misc.Unsafe. Данный класс позволяет резервировать память вне кучи JVM используя вызовы native методов. Так, например, для резервирования памяти используется allocateMemory , который приводит к вызову си кода - os::malloc(sz, mtInternal).

1.6 Отбор кейлоггеров для тестирования представленных выше решений

1.6.1 Априорный анализ предлагаемых решений

В представленных выше решениях преобразование ввода осуществляется на уровне взаимодействия с пользователем: введённая последовательность либо синтезируется на базе стенографического ввода (см. 2.2.1 и скринкаст), либо является динамически изменённой на основании текущей раскладки. Перехватывать нужно не ввод, а результат его преобразования. Передача данных через транспорты закодирована RSA. Таким образом, перехват только «сырых» последовательностей введённых символов является неэффективным «по построению», тем самым отсекая угрозы со стороны ядерных кейлоггеров и низкоуровневых перехватчиков клавиатуры. Постулируется 100% неуязвимость относительно данных методов. Это побуждает к отбору высокоуровневых кейлоггеров пространства пользователя. С другой стороны, активная работа решений с графической оболочкой (ввод в формы, набор капчи, пароля и др.) целевой системы, которая (как предлагают авторы) может быть реализована средствами JavaNativeHook, представляет уязвимость с точки зрения повторного перехвата контекста целевого процесса и получения доступа к уже введённой преобразованной последовательности. На выходе, в случае успеха, будут уже готовые начальные символы.

С точки зрения пост-анализа полученных кейлоггерами данных, система также имеет уязвимости:

1) В случае задачи стенографического ввода – перехват событий мыши/тачпада.
2) В случае динамической раскладки – возможность снятия скриншота с экрана, на котором расположена клавиатура с текущей раскладкой.

На данном этапе работ авторами постулируется отсутствие злоумышленного реверс-анализа стенографического ввода: считается, что перехватить стенографический шифр можно лишь заполучив полную таблицу шифрования, поэтому упор защиты от пост-анализа делается на уязвимости типа 2. Тем не менее, планируется тестирование кейлоггеров, записывающих события мыши*.

1.6.2 Список тестируемых кейлоггеров

На основании модели угроз, был составлен список кейлоггеров для тестирования на Windows и Linux x86-64 архитектуры.

Таблица 7 – Кейлоггеры для тестирования на Windows и Linux x86-64 архитектуры

Windows 8.1, 10	
Puntoswitcher +	Позволяет осуществить перехват ввода клавиш в контексте активного приложения. Перехват преобразованного ввода в форму - вопрос тестирования
Refog Free Keylogger + KidLogger + Spyrix Free Keylogger +	Помимо основной функциональности, позволяют снимать скриншоты
*Refog-personal-monitor +	Помимо основной функциональности, позволяет отслеживать события мыши и снимать скриншоты
Linux Mint 17, CentOS 7	
iXKeylog https://github.com/magcius/keylog/	Осуществляют перехват X-сервера, что представляет собой угрозу с точки зрения глобальной инъекции ввода
Wireshark	Будет произведена дополнительная проверка на случай перехвата транспортов
LogKeys	Распространённый кросс-платформенный (*nix) кейлоггер с открытым исходным кодом

1.7 Результаты

1.7.1 Методология и результаты тестирования

С целью тестирования прототипов на вышеперечисленных ОС и кейлоггерах были развёрнуты ВМ под управлением VirtualBox. Минимальная конфигурация ВМ для тестирования содержит JRE (JDK), jar-файлы прототипов, PostgreSQL, Apache

Tomcat и тестируемые кейлоггеры. Тестируемое решение запускалось одновременно с кейлоггером, производился ввод последовательности символов, после чего разбирались логи кейлоггера и решения. Тест считался успешно пройденным, если результат из лога кейлоггера не совпадал с результатом из лога решения.

Таблица 8 – результаты тестирования методов на различных кейлоггерах.

	СДШВ, Windows 8.1	**ССВ, Windows 8.1**	**СДШВ, Windows 10**	**ССВ, Windows 10**
Refog Personal Monitor / Keylogger	текстовый тест пройден / тест на скриншот провален.	тест пройден	текстовый тест пройден / тест на скриншот провален.	тест пройден
Spyrix Keylogger	тест пройден	тест пройден	тест пройден	тест пройден
PuntoSwitcher	тест пройден	тест пройден	тест пройден	тест пройден
KidLogger	тест пройден	тест пройден	тест пройден	тест пройден
	СДШВ, CentOS 7	**ССВ, CentOS 7**	**СДШВ, Linux Mint**	**ССВ, Linux Mint**
logkeys	тест пройден	тест пройден	тест пройден	тест пройден
ixkeylog	тест пройден	тест пройден	тест пройден	тест пройден
WireShark (http)	тест пройден	тест пройден	тест пройден	тест пройден

Для тестирования Android-версии ССВ apk-сборка последней загружалась на телефон с Android 5.1 и Android 6.0, после чего производился тест с кейлоггерами из Play Market - “Input Events” и “Kids Logger”. Все тесты были успешными для ввода через клавиатуру в качестве сервиса и при вводе через активити.

1.7.2 Результаты для ОС - инъекции

В процессе исследования возможности прямой инъекции текста в поля приложений, реализованных на основе X11, Qt и пр., не было обнаружено общего безопасного метода инъекции текста в поле ввода с фокусом программным способом: модификация виртуальной консоли или кадрового буфера либо глобально затронет все графические подсистемы, что может служить источником для

перехвата кейлоггерами пространства пользователя, либо потребует изменения API (/dev/fbX для Linux), что лишит универсальности как предлагаемые методы, так и графическую подсистему. Предлагается компромиссное решение - подмена глобальной таблицы символов внутри конкретной графической оболочки, с учётом того, что кейлоггеры зачастую считывают не результат отображения по данной таблице, а входящее событие - источник. На базе предположения реализовано универсальное решение в контексте одной графической подсистемы - замена таблицы символов для X-сервера.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Andrew Schulman // The Extent of Systematic Monitoring of Employee E-mail and Internet Use, 2001. Электронный источник: http://www.diogenesllc.com/internetmonitoring.pdf.

2 Nikolay Grebennikov, “Keyloggers:how they work and how to detect them (Part 1)”, Securelist, AO Kaspersky Lab, 2007. Электронный источник: https://securelist.com/analysis/publications/36138/keyloggers-how-they-work-and-how-to-detect-them-part-1/

3 Nikolay Grebennikov, “Keyloggers: implementing keyloggers in Windows: part two”, Securelist, AO Kaspersky Lab, 2011. Электронный источник: https://securelist.com/analysis/publications/36358/keyloggers-implementing-keyloggers-in-windows-part-two/

4 “Blue Pill archived pages”, 2007. Электронный источник:

http://web.archive.org/web/20080418123748/http://www.bluepillproject.org/

5 Windows Internals / David Solomon, Mark Russinovich, Alex Ionescu. — Microsoft Press, 2012.

6 Зайцев О. Rootkits, SpyWare_AdWare, Keyloggers & BackDoors. Обнаружение и защита / Олег Зайцев. — Санкт-Петербург: БХВ-Петербург, 2006.

7 Электронный источник: http://resources.infosecinstitute.com/hooking-system-service-dispatch-table-ssdt

8 Дмитрий Пукаленко, “Современные rootkit-технологии в Linux”, 2012. Электронный источник: http://nobunkum.ru/ru/linux-rootkits#user-mode-ptrace.

9 Электронный источник: http://www.okbsapr.ru/kanner_2012_5.html

10 Электронный источник: http://althing.cs.dartmouth.edu/local/vsc07.html

11 Электронный источник: https://habrahabr.ru/company/securitycode/blog/245539/

12 Электронный источник: http://www.ouah.org/kernel-hijack.txt

13 Электронный источник: https://msdn.microsoft.com/en-us/library/windows/hardware/ff542278(v=vs.85).aspx

14 Электронный источник: https://github.com/kwhat/jnativehook

15 Электронный источник: https://gitlab.com/IdeaCreation/RMCPC

16 Электронный источник: https://drive.google.com/drive/folders/0B0D1av_FjVNGX1BzNFJaR3ZmUU0

17 Электронный источник: https://github.com/nefanov/kernel_tmp/blob/master/wp_CR0_bit.c

18 Электронный источник: https://github.com/nefanov/kernel_tmp

19 Электронный источник: http://www.berkeley.edu/news/media/releases/2005/09/14_key.shtml

20 Gregg Keizer, «Computerworld Россия», № 01, 2008. Электронный источник: http://www.osp.ru/cw/2008/01/4715364

21 Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan and Angelos D. Keromytis Computer Science Department, Columbia University, “The Spy in the Sandbox – Practical CacheAttacks in Javascript”, 1 Mar 2015.

22 Электронный источник: http://cs.tau.ac.il/~tromer/acoustic/ec04rump/

Приложение А.

Пояснительная информация

Ниже приводятся ссылки на демонстрации работы прототипов, репозиторий с исходными текстами разработанных прототипов [15,16], а также путь к материалам на предоставляемом диске в директории «Исследование методов сбора печатаемых символов».

СДШВ

Демонстрация работы прототипа – в скринкасте №2 :

https://drive.google.com/drive/folders/0B0D1av_FjVNGSVdlSDNXU3FUWTQ

Диск: Презентации/KPS-PC/*.mp4

Исходные коды прототипа в репозитории :

https://gitlab.com/IdeaCreation/RMCPC/tree/master/kps-pc

Диск: Sources/master/kps-pc

/release/kps-pc

/release-unsafe/kps-pc

/jni/kps-pc

Название в проекте: “KPS”

jar-сборка: kps-SNAPSHOT-0.0.1.jar

ССВ

Демонстрация работы прототипа – в скринкасте №1 :

https://drive.google.com/drive/folders/0B0D1av_FjVNGLThsSTVwV19XQ28

Диск: Презентации/VIS-PC/*..mp4

Исходные коды прототипа в репозитории :

https://gitlab.com/IdeaCreation/RMCPC/tree/master/vis-pc

Диск: Sources/master/vis-pc

/release/vis-pc

/release-unsafe/vis-pc

Название в проекте: “VIS”

jar-сборка: vis-SNAPSHOT-0.0.1.jar

Документация, инструкции адинистратора и пользователя:

Диск: Документация/

Хранилище виртуальных машин, на которых проводилось тестирование:

Диск: VM/