СЕКЦИЯ 2. СОВРЕМЕННЫЕ ВЫЗОВЫ В ОБЛАСТИ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.49

Агапова В.О.
Белорусский государственный университет
информатики и радиоэлектроники, Минск

Научный руководитель:
Лихачевский Д.В.
Белорусский государственный университет
информатики и радиоэлектроники, Минск

АНАЛИЗ МЕТОДОВ СКРЫТИЯ ВРЕДОНОСНЫХ ПРИЛОЖЕНИЙ

Аннотация. В статье рассматриваются методы, которыми вредоносные приложения стремятся ухитриться и уйти от обнаружения в *sandbox*-средах и антивирусных системах. Приведен анализ основных техник, выявлены их преимущества и ограничения. Рассмотрены возможные подходы к противодействию таким механизмам.

Ключевые слова: вредоносное посещение, *sandbox*, антивирусная защита, эвазия, кибербезопасность.

Введение

Современные атаки на информационные системы зачастую осуществляются через внедрение вредоносных приложений. Для их обнаружения и исследования широко используются *sandbox*-среды и антивирусы. Тем не менее, авторы вредоносных ПО создают техники, позволяющие уклоняться от их обнаружения.

Основная часть

Техники уклонения от *sandbox'ов* и антивирусов можно условно разделить на несколько категорий:

Обнаружение виртуальной среды

Многие *sandbox'ы* используют виртуальные машины (*VM*) для изоляции исполняемого кода. Вредоносные приложения анализируют признаки виртуализации, такие как наличие специфических драйверов (например, *VBoxGuest.sys*, *vmhgfs.sys*), виртуальных сетевых адаптеров (*vmxnet3*, *virtio*) или неестественные конфигурации оборудования (необычно маленький объем памяти, один процессор, отсутствие физических устройств) [5].

Расширенные методы обнаружения *VM* включают:

Определение задержек выполнения инструкций, характерных для виртуализации (например, через *RDTSC*, *RDPMC*, *CPUID*) [3].

Считывание специфических *CPUID*-инструкций (*Hypervisor present bit*, *hypervisor vendor ID*).

Проверку нестандартных размеров кэш-памяти (*L3 cache*) и особенностей структуры *ACPI*-таблиц.

Сканирование имен *BIOS'ов* и производителей оборудования на наличие упоминаний о виртуализации (например, *VMware*, *VirtualBox*, *QEMU*) [5].

Анализ специфических процессов и сервисов, характерных для *VM* (например, *vboxservice.exe*, *vmtoolsd.exe*).

Новейшие техники:

Использование побочных каналов (*side-channel analysis*) для определения наличия гипервизора [3].

Атаки через чтение аномалий работы *TLB* (*Translation Lookaside Buffer*) и кэш-памяти.

Некоторые вредоносные программы даже динамически меняют свое поведение, например, активируются только в случае отсутствия признаков виртуализации. Более того, в некоторых случаях осуществляется активная имитация нестабильной работы программного обеспечения, чтобы спровоцировать *sandbox* на преждевременное завершение анализа. Часто также применяются техники контроля температуры *CPU* и частоты отклика устройств ввода, поскольку виртуальные среды не всегда корректно эмулируют физические параметры реального оборудования.

Проверка времени выполнения

Sandbox-окружения ограничены по времени анализа (обычно от нескольких секунд до 5 минут), чтобы минимизировать ресурсы.

Классические техники:

Использование задержек (*sleep*, *NtDelayExecution*), нередко с модификацией параметров таймера для обхода простого пропуска времени анализа (*sleep-patching*) [2].

Измерение реального времени через *RDTSC*, *GetTickCount*(), *QueryPerformanceCounter*().

Расширенные методы:

Использование техник *sleep obfuscation*: выполнение коротких периодов сна, перемежаемых безвредной активностью [1].

Завуалированные задержки через ресурсоемкие задачи: перебор данных, бесконечные циклы с условием выхода в зависимости от времени.

Environmental Keying: активация вредоносного кода только по достижении определенного времени, даты или события (например, запуск только в пятницу вечером). Дополнительно, некоторые вредоносные программы анализируют изменения системных часов в процессе выполнения, чтобы выявить попытки искусственного ускорения времени *sandbox'ом*. Применяются также таймеры на базе внешних событий, таких как отклик сетевого оборудования или появление новых процессов пользователя, что затрудняет форсированную эмуляцию активности.

Ожидание пользовательского взаимодействия

Sandbox'ы часто не моделируют реальное поведение пользователя.

Традиционные методы:

Проверка наличия движения мыши (*GetCursorPos*, *GetLastInputInfo*) [2].

Ожидание ввода текста или нажатия клавиш (*GetAsyncKeyState*, *SetWindowsHookEx*).

Расширенные техники:

Анализ естественности движений мыши: криволинейность, скорость, ускорение.

Ожидание нескольких разных типов взаимодействия: клик + перетаскивание + прокрутка + закрытие окна.

Проверка фоновых приложений и их активности (проверка наличия популярных приложений: *Chrome*, *Teams*, *Skype*, антивирусных процессов).

Слежение за состоянием окон: минимизация, разворачивание окон и фокус ввода. Также вредоносные программы могут измерять скорость и ритм набора текста, определяя, являются ли нажатия клавиш результатом реальной деятельности человека или автоматизированного скрипта. В некоторых случаях вредоносный код ожидает многократных взаимодействий с разными окнами, что трудно воспроизвести в *sandbox'ax* за короткий период анализа.

Шифрование, упаковка и полиморфизм

Шифрование частей кода, полиморфизм и метаморфизм позволяют избегать статического анализа.

Методы:

Полиморфизм: изменение структуры кода без изменения функциональности при каждом запуске.

Метаморфизм: полное переписывание кода, создание новых функций с тем же поведением [1].

Упаковка с помощью известных или самописных пакеров (*UPX*, *Themida*, собственные крипторы).

Использование стеганографии: сокрытие вредоносного кода внутри изображений, документов и других медиафайлов.

Новые тренды:

Fileless Malware: использование легитимных системных инструментов (например, *PowerShell*, *WMI*, *Living-off-the-Land Binaries – LOLBins*) без записи файлов на диск.

Атаки через многослойную упаковку: вредоносный код зашифрован несколько раз с использованием разных алгоритмов и методов упаковки. При этом некоторые современные обфускаторы могут динамически изменять сигнатуры *API*-вызовов и использовать методы шифрования с одноразовыми ключами (OTP), что практически исключает возможность статической декомпиляции. Нередко также применяется внедрение промежуточных виртуальных машин внутри основного кода, усложняющее процесс анализа.

Использование легитимных процессов

Классические методы:

Process Hollowing: создание процесса, замена его кода вредоносным [2].

DLL Injection: внедрение своей библиотеки в адресное пространство другого процесса.

Parent Process Spoofing: подмена родительского процесса для маскировки активности.

Современные методы:

AtomBombing: внедрение кода без изменения памяти процесса напрямую через атомарные таблицы *Windows*.

Process Doppelgänging: создание процесса с использованием *NTFS*-транзакций для запуска вредоносного кода без фактического сохранения исполняемого файла на диск [4].

Heaven's Gate: переход между 32-битным и 64-битным режимами исполнения в *Windows* для обхода системных ограничений.

Callback Injection: использование легитимных функций *Windows* (например, *CreateRemoteThread*, *QueueUserAPC*) для внедрения кода.

Дополнительно:

Инжекция через сетевые стеки (например, внедрение через фильтры сетевого драйвера).

Использование *Trusted Installer* сервисов для запуска вредоносного кода с привилегиями *SYSTEM*. Все чаще встречаются сложные многоступенчатые схемы, где вредоносный код сначала захватывает управление над менее защищенными пользовательскими процессами, а затем через цепочку внедрений переходит к системным службам. Используется также методика внедрения через COM-объекты, когда вредоносный код регистрирует себя как легитимный обработчик событий в системе.

Список использованных источников:

1. You I., Yim K. Malware Obfuscation Techniques: A Brief Survey. – 2010.

2. Egele M., Scholte T., Kirda E., Kruegel C. A Survey on Automated Dynamic Malware-Analysis Techniques and Tools. ACM Computing Surveys, 2012.

3. Dinaburg A., Royal P., Sharif M., Lee W. Ether: Malware Analysis via Hardware Virtualization Extensions. Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS), 2008.

4. Moser A., Kruegel C., Kirda E. Exploring Multiple Execution Paths for Malware Analysis. –Proceedings of the 2007 IEEE Symposium on Security and Privacy.

5. Ugarte-Pedrero X., Santos I., Bringas P.G., Alvarez G. Countering virtualization detection techniques. Computers & Security, 2015.

Agapova V.O.
Belarusian State University of
Informatics and Radioelectronics, Minsk

Scientific supervisor:
Likhachevsky D.V.
Belarusian State University of
Informatics and Radioelectronics, Minsk

ANALYSIS OF METHODS FOR HIDING MALICIOUS APPLICATIONS

Abstract. The article examines the methods by which malicious applications attempt to evade detection in sandbox environments and antivirus systems. An analysis of the main techniques is presented, highlighting their advantages and limitations. Possible approaches to counteracting these mechanisms are considered.

Keywords: malicious application, sandbox, antivirus protection, evasion, cybersecurity.