0.1. *Лебедев Р.К., Ситнов В.Е.* Метод создания исполняемого файла Linux без машинного кода при помощи ROP

В отсутствие сетевых взаимодействий машинный код полностью определяет поведение программ и библиотек, поэтому он подвергается анализу как в рамках обратной разработки, так и в рамках проверки программы на вредоносность, например, по сигнатурам. В связи с этим возникает вопрос: может ли существовать программа, не содержащая машинного кода, но выполняющая некоторые полезные действия?

Один из наиболее известных ответов на этот вопрос был дан исследователями информационной безопасности, открывшими подход ROP (Return Oriented Programming) [1], традиционно используемый при эксплуатации переполнения стека. До появления технологий защиты NX-bit и DEP [2] полезная нагрузка атак размещалась прямо в памяти в виде машинного шелл-кода. Когда это стало невозможным, был изобретен данный подход, суть которого состоит в сборке кода из уже загруженных в память программы фрагментов кода (например, библиотек), также называемых «гаджетами», каждый из которых заканчивается инструкцией RET, откуда и происходит название метода.

В данной работе ROP был применен для генерации исполняемых файлов ОС Linux, совершенно не содержащих машинного кода. Был реализован ROP-кодогенератор для компиляторной инфраструктуры ELVM [3], включающей компилятор языка Си 8cc, используемой обычно для эзотерических языков программирования. Для этого был отобран базис из распространенных гаджетов, реализующих необходимые инструкции: загрузку и запись в память, арифметические операции и операции управления исполнением.

Как показывает практика, поиск гаджетов может быть очень затруднительным процессом, и даже этот базис не всегда может быть найден в загруженных библиотеках. В данной работе для решения этой проблемы предложен новый подход: поиск гаджетов во всех доступных в системе библиотеках и добавление их в список используемых исполняемым файлом, так как он находится под полным контролем.

Метод был реализован при помощи языка Python, ELVM и библиотеки LIEF. Для передачи исполнения на ROP-цепочку, размещенную в секции данных, был использован массив конструкторов DT_INIT_ARRAY и несколько ROP-гаджетов, вызываемых как конструкторы и реализующих загрузку нужного значения в указатель стека RSP. Программа генерируется в виде разделяемой библиотеки, которая начинает исполнение в момент своей загрузки, что позволяет избежать необходимости иметь разрешение файла на исполнение.

Тестирование метода осуществлялось против систем проверки целостности исполняемых файлов Linux IMA, DIGSIG, а также систем блокировки исполнения AppArmor и опции mount noexec. В результате тестирования выявлена абсолютная эффективность метода против всех упомянутых систем, что указывает на важность разграничения доверенного и недоверенного кода при их реализации.

Научный руководитель — д.т.н. Павский К. В.

Список литературы

[1] Shacham H. The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86) // Proceedings of 2007 ACM Conference on Computer and Communications Security. ACM, 2007. P. 552–561.

[2] Litchfield D. Buffer Underruns, DEP, ASLR and improving the Exploitation Prevention Mechanisms (XPMs) on the Windows platform. [Электронный ресурс]. URL: https://www.nccgroup.com/media/olvmmhpa/_xpms.pdf (дата обращения 19.09.2025).

[3] EsoLangVM Compiler Infrastructure. [Электронный ресурс]. URL: https://github.com/shinh/elvm (дата обращения 19.09.2025).