Статья Побег из песочниц с помощью одного бита - флага ловушки

Побег из песочниц с помощью одного бита - флага ловушки

Unit 42 обнаружил определенный единственный бит (флаг ловушки) в регистре процессора Intel, которым могут злоупотреблять вредоносные программы, чтобы избежать обнаружения песочницей. Вредоносное ПО может определить, выполняется ли оно на физической или виртуальной машине (ВМ), отслеживая реакцию ЦП после установки этого единственного бита.

Песочница - это популярный метод, используемый для определения вредоносности образца. Песочница анализирует поведение двоичного файла при его выполнении в контролируемой среде. Чтобы преодолеть проблему анализа большого количества двоичных файлов с ограниченными вычислительными ресурсами, виртуальные машины используются для создания песочниц. Чтобы избежать обнаружения, вредоносная программа попытается определить, выполняется ли она на физической или виртуальной машине. Когда вредоносная программа обнаруживает, что выполняется на виртуальной машине, она прекращает выполнение или предоставляет поддельные результаты, чтобы скрыть свои настоящие намерения.

Некоторые из наиболее распространенных методов уклонения включают вредоносное ПО, выполняющее различные проверки системы в среде, в которой оно выполняется. Например, вредоносные программы часто ищут аномальное разрешение экрана, размер жесткого диска и физической памяти. Песочницы могут создавать контрмеры, например возвращать вредоносной программе ложную информацию во время этих проверок.

В этой статье рассказывается, как вредоносные программы могут обнаруживать различия в поведении ЦП на виртуальной или физической машине с помощью всего лишь одного бита в регистре ЦП.

Клиенты Palo Alto Networks защищены от семейств вредоносных программ с помощью аналогичных методов обхода в песочнице с помощью Cortex XDR или межсетевого экрана нового поколения WildFire и Threat Prevention.

Пошаговый режим - флаг ловушки (TF)

Чтобы обнаружить использование виртуальной машины в песочнице, вредоносное ПО может проверить поведение ЦП после включения флага ловушки.

Флаг прерывания (TF) - это 8-й отдельный бит в регистре EFLAGs архитектуры процессора Intel x86. Если TF включен до выполнения одной инструкции, ЦП вызовет исключение (пошаговый режим) после завершения инструкции. Это исключение останавливает выполнение ЦП, чтобы обработчик исключений мог проверить содержимое регистров и области памяти. Прежде чем разрешить выполнение кода, ЦП также должен очистить TF.

Чтобы определить, используется ли виртуальная машина, вредоносная программа может проверить, было ли одношаговое исключение доставлено в правильную инструкцию ЦП, после выполнения определенных инструкций (например, CPUID, RDTSC, IN), которые вызывают выход из виртуальной машины с включенным TF. Во время выхода из виртуальной машины гипервизор, также известный как монитор виртуальной машины (VMM), будет имитировать воздействие физического процессора, с которым он сталкивается.

Следующая последовательность инструкций объясняет поведение ЦП после включения TF на физической машине.

```
.text:00401068          pushf
.text:00401069          or      dword ptr [esp], 100h
.text:00401070          popf
.text:00401071          rdtsc
.text:00401073          nop
.text:00401074          nop
```


Первые три инструкции включают бит TF в регистре EFLAGs ЦП. RDTSC выполняется с включенным TF. На физическом компьютере исключение будет доставлено первой инструкции без операции (NOP) (0x00401073). Обратите внимание, что исключение возникло в инструкции сразу после выполнения инструкции с включенным TF.

```
.text:00401068          pushf
.text:00401069          or      dword ptr [esp], 100h
.text:00401070          popf
.text:00401071          rdtsc           ; TF Enabled
.text:00401073          nop             ; exception
.text:00401074          nop
```


Выполнение той же последовательности инструкций в виртуальной машине даст другой эффект. В виртуальной машине выполнение RDTSC приведет к выходу из виртуальной машины. Гипервизор будет выполнять свои обычные задачи по имитации поведения инструкции RDTSC. Однако реализация гипервизора с неправильной эмуляцией TF приведет к игнорированию TF и выполнение кода продолжится до первой инструкции NOP. Во время выполнения первой инструкции

NOP TF все еще включен, поскольку TF не обрабатывается гипервизором. Это приводит к возникновению исключения во второй инструкции NOP (0x00401073). Правильная реализация потребует, чтобы гипервизор внедрил исключение отладки после эмуляции инструкции, которая вызвала выход виртуальной машины и очистку TF.

```
.text:00401068          pushf
.text:00401069          or      dword ptr [esp], 100h
.text:00401070          popf
.text:00401071          rdtsc           ; TF Enabled
.text:00401073          nop             ; TF Enabled
.text:00401074          nop             ; exception
```


В качестве метода обхода песочницы вредоносное ПО будет использовать обработчик исключений в дополнение к приведенной выше последовательности инструкций, чтобы проверить, в какой инструкции возникло исключение. В следующем разделе описывается реальный пример семейства вредоносных программ, которое использовало эту технику для обхода песочниц.

Пример из реального мира

Lampion - это семейство вредоносных программ, нацеленных на пользователей в Португалии. Lampion использовала несколько системных проверок, чтобы избежать обнаружения песочницей. Один из методов - использование пошагового режима с TF, как обсуждалось в предыдущем разделе.

Lampion реализовала все свои системные проверки с помощью инструкций x86 и минимального количества вызовов Windows API. Это позволило семплам Lampion скрыть свое поведение от песочниц. Образцы Lampion прекратили бы свою работу, если бы вредоносное ПО определило, что оно выполнялось внутри виртуальной машины. Системные проверки также переплетаются с множеством методов защиты от реверс-инжиниринга, чтобы скрыть их от аналитиков.

На следующем снимке экрана показан фрагмент инструкций, скрытых в образце Lampion, который выполняет проверку системы.

```
007F0E0C                db 'HeapCreate',0
007F0E17 word_7F0E17    dw 0                    ; DATA XREF: .text:007F3FD5↓o
007F0E19                db 'GetLocalTime',0
007F0E26 word_7F0E26    dw 0                    ; DATA XREF: .text:007F4009↓o
007F0E28                db 'CreateDirectoryW',0
007F0E39 word_7F0E39    dw 0                    ; DATA XREF: .text:007F3D8D↓o
007F0E3B                db 'RegDeleteValueW',0
007F0E4B ; ---------------------------------------------------------------------------
007F0E4B                popf                    ; TF enabled!
007F0E4C                rdtsc                   ; Privileged instruction
007F0E4E                nop
007F0E4F                pushf
007F0E50                pushf
007F0E51                pusha
007F0E52                lea     esp, [esp+28h]
007F0E56                jnp     loc_7EE5F2
007F0E5C                push    6600F72Fh
007F0E61                pusha
007F0E62                jmp     loc_7F8CD7
007F0E62 ; ---------------------------------------------------------------------------
007F0E67 word_7F0E67    dw 0                    ; DATA XREF: .text:007F3EC5↓o
007F0E69                db 'OpenProcess',0
007F0E75
007F0E75 ; =============== S U B R O U T I N E =======================================
```


Ниже приведен псевдокод, демонстрирующий, как Lampion выполняет одну из проверок своей песочницы, активируя TF по инструкции, которая вызывает выход виртуальной машины.

```
Anti_sandbox_Check()
{
    try
    {
                    pushfd
                    or dword ptr[esp], 0×100
                    popfd
                    rdtsc
                    nop
                    pushf
                    pushf
    }
    catch ()
    {
        dwEIP = ExceptionInfo→ContextRecord→EIP;
        bByte = ReadByte(dwEIP);
        if bByte ≠ 0×90 ExitProcess();
    }
}
```


Инструкция сразу после инструкции RDTSC - NOP. Байт-код для инструкции NOP - 0x90. Обработчик исключений будет проходить структуру ContextRecord, чтобы найти адрес инструкции в регистре указателя расширенной инструкции (EIP), когда возникнет исключение. Затем инструкция сравнивается с байтом 0x90, и вредоносная программа завершает работу, если проверка не удалась.

На следующем снимке экрана показано что EIP = 0x7F0E4E, когда произошло исключение. На снимке экрана показано, что EIP = 0x7F0E4E за исключением.

Вредоносные программы против песочниц

На протяжении многих лет идет постоянная игра в кошки-мышки между авторами вредоносных программ, создающих методы уклонения для предотвращения эффективного анализа, и авторами песочницы, которые исследуют новые способы борьбы с этими уловками.

Это один из основных причин, который побудил нас в Palo Alto Networks создать собственный гипервизор для анализа вредоносных программ. Поскольку у нас есть полный контроль над программным стеком, включая уровень виртуализации, мы можем реагировать на новые и возникающие угрозы. В этом конкретном случае, как только мы определили проблему с неправильной эмуляцией флага ловушки, наша команда гипервизора смогла протестировать и развернуть исправление. С тех пор эта проблема уклонения была решена для любого образца вредоносного ПО, использующего эту технику.

Клиенты Palo Alto Networks дополнительно защищены от семейств вредоносных программ с помощью аналогичных методов обхода в песочнице с помощью Cortex XDR или межсетевого экрана нового поколения с подписками безопасности WildFire и Threat Prevention. Пользователи AutoFocus могут отслеживать обсуждаемые здесь вредоносные программы с помощью тега Lampion. Другие аналогичные методы обхода песочницы, основанные на злоупотреблении инструкциями или регистрами процессора Intel, не будут работать против WildFire.

Индикаторы компроментации

Сэмпл Lampion

EB3F2BE571BB6B93EE2E0B6180C419E9FEBFDB65759244EA04488BE7C6F5C4E2