Использование телеметрии Windows для закрепления в системе

Дата публикации: 9 июня, 2020 год
Оригинал
Автор: Christopher Paschen

Сегодня мы поговорим о методе закрепления в системе, который основывается на волшебной технологии - телеметрии, в Windows версиях последнего десятилетия. Процесс описываемый здесь применим к Windows начиная с 2008R2/Windows 7 по 2019/Windows 10.

Данная техника требует прав локального админа (возможность писать в ветку реестра HKLM) и ее результаты не видны в автозагрузке.

TLDR:
Если вы в танке, то вот список того, что вам надо знать:

1. Убедитесь, что целевой компьютер имеет активное подключение к сети

2. Добавьте ключ с любым именем в **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\TelemetryController**

3. Внутри этого ключа создайте переменную “Command” с типом *Reg_sz* и установите значение равное **.exe** файлу, который вы хотите запустить

4. Создайте DWORD ключи для Maintenance, Nightly, Oobe и установите их равными 1 (Nightly требует запуска раз в 24 часа)

5. Наслаждайтесь! Файл должен периодически запускаться, с помощью планировщика заданий Windows

6. Вы можете проверить с помощью команды **schtasks /run /tn “\Microsoft\Windows\Application Experience\Microsoft Compatibility Appraiser”** или запустив вручную задание в gui планировщика.

В качестве визуального примера, данные модификации реестра запустят notepad.exe под SYSTEM, когда соответствующее задание в планировщике станет активным

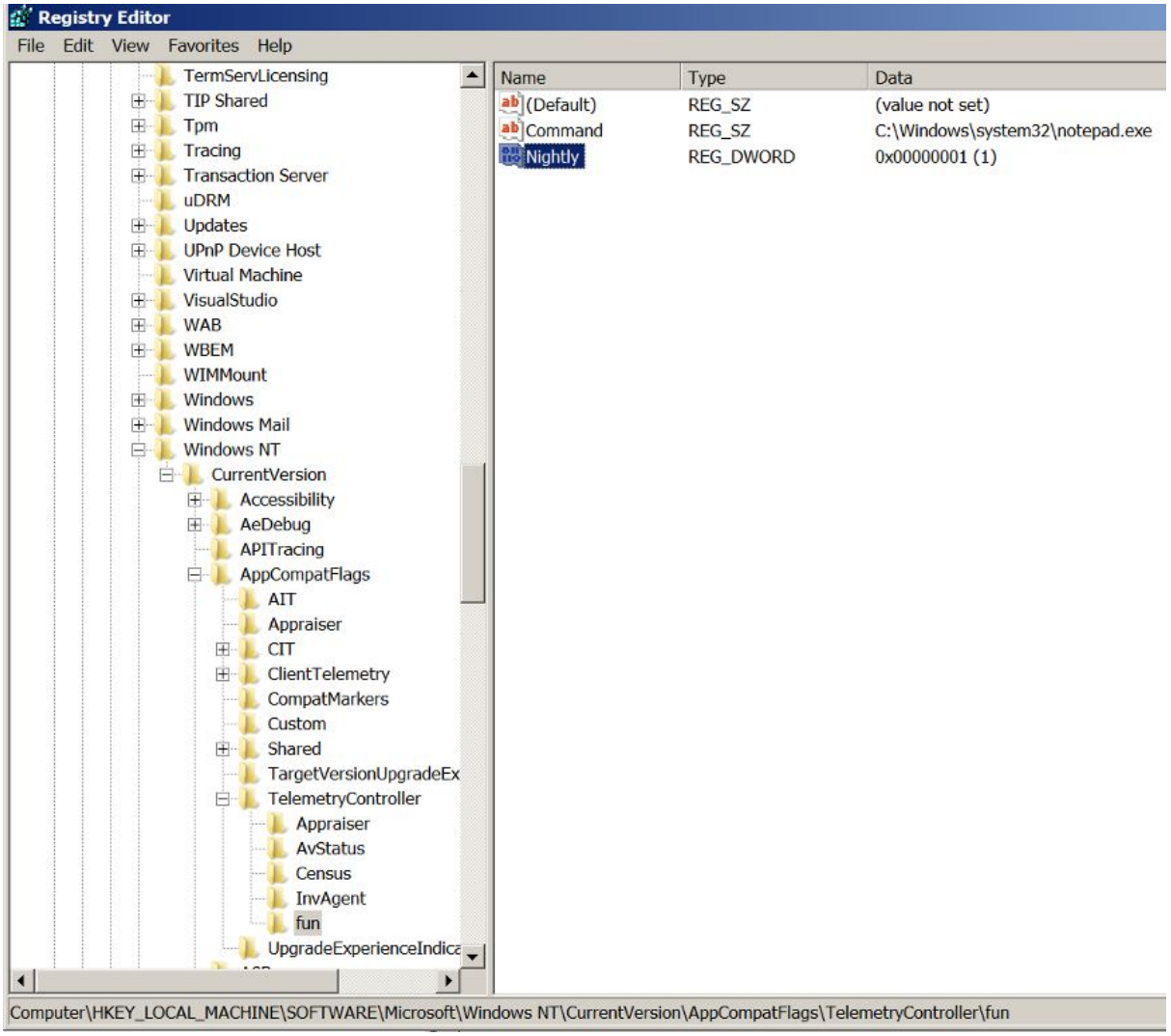


Рисунок 1 - Закрепляемся, с помощью RegEdit.exe

Как и почему

Так, давайте посмотрим что же произошло и почему ваше задание выполняется с правами SYSTEM. Во-первых, **c:\windows\system32\CompatTelRunner.exe** - это по всей видимости бинарник, предназначенный для запуска разных заданий телеметрии. Сам бинарник особо не собирает данные. Похоже, что он проверяет статистику ОС, убеждается в наличии сети и потом запускает различные команды, для непосредственно сбора телеметрических данных. Можете называть его менеджером телеметрии.

Бинарник работает гибко и берет команды для запуска из реестра. Проблема в том, что он запускает любую команду, не проверяя путь в ней или тип. На первый взгляд - это серьезная проблема в безопасности, но я так не считаю. Если вы уже получили админские права, которые нужны для запуска этого процесса, то у вас уже имеется куча опций для эксплуатации.

Когда **CompatTelRunner.exe** запускается (текущая версия - Май 2020), идет проверка успешности выполнения нескольких условий перед тем, как начать работу с телеметрией.

Одно из следующих условий должно быть выполнено:

1. ОС является Windows 10/Server 2019

2. ОС - клиентская версия винды

3. **HKEY_LOCAL_MACHINE \Software\Microsoft\Windows\CurrentVersion\Policies\DataCollection\CommercialDataOptIn** - это DWORD не равный нулю

Интересно то, что эти проверки были добавлены в определенный момент, после релиза Windows server 2016. До обновления **CompatTelRunner.exe** их не было, и этот бинарник мог запускать команды из реестра вне зависимости от версии Windows.

После проверок, в зависимости от наличия или отсутствия параметров командной строки выбирается режим запуска программы. Существует три режима запуска соответствующие некоторым условиям.

Если параметр командной строки указывает на DLL или функцию, то **CompatTelRunner.exe** проверяет их по списку доверенных. Далее он запускает DLL провайдер и завершается. Если DLL или имя функции отсутствуют, то идет дальнейшее определение режима запуска.

Второй режим запуска (OOBE) активен, если существует ключ **HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompatFlags\TelemetryController\Oobe** и параметр -maintenance не передан. Данный ключ удаляется, после проверки его наличия.

Если *-maintenance* передан, то мы проверяем, можем ли мы запускать и входить в режим под номером 0. В таком случае, проверяется выставлен ли **HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompatFlags\TelemetryController\TestAllowRun** в REG_DWORD не равный нулю или проверка пропускается.

Проверяются следующие условия:

1. Режим энергосбережения не должен быть активным. Если он активный, то проверка всегда завершается неудачей.

2. Проверка успешна, если питание компьютера подключено

3. Если проверки были провалены четыре раза ранее, то мы проверяем состояние батареи. Если оно неизвестно или заряд батареи больше 5%, или батарея заряжается, то проверки проходят успешно.

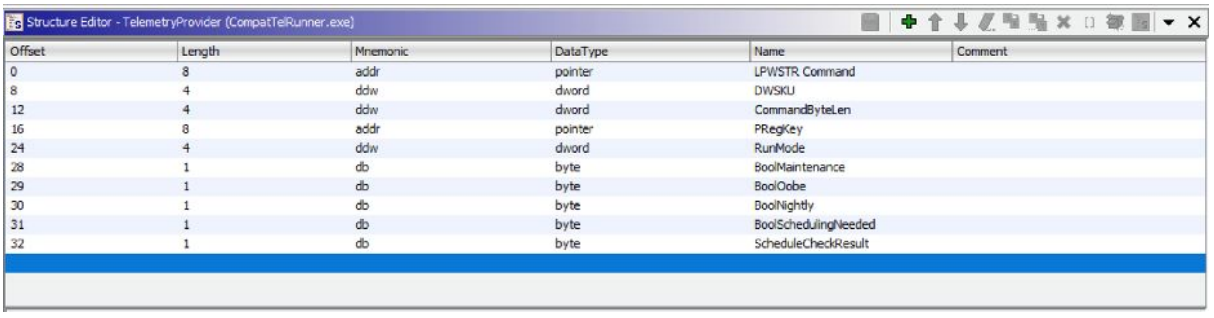
   После прохождения проверок, программа выставляет ключ реестра **RunsBlocked** в 0. В случае неудачи, **RunsBlocked** увеличивается на 1.

   Если параметры командной строки не были переданы, то **CompatTelRunner.exe** входит в режим 1 (Nightly).

   После определения режима запуска, происходит несколько проверок планировщика заданий. После, мы доходим до интересующего нас места, называемого RunTelemetry.

   Есть несколько дополнительных проверок, если наш режим работы равен нулю. Если они пройдены удачно или режим запуска не нулевой, то программа открывает **HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompatFlags\TelemetryController**.

   Все подключи реестра перечислены в этой папке и используются при вызове функции Initialize, для заполнения структур. С помощью дизассемблера я нашел то, что используется в структуре.



Offset	Length	Mnemonic	DataType	Name	Comment
0	8	addr	pointer	LPWSTR Command	
8	4	ddw	dword	DWSKU	
12	4	ddw	dword	CommandByteLen	
16	8	addr	pointer	PRegKey	
24	4	ddw	dword	RunMode	
28	1	db	byte	BoolMaintenance	
29	1	db	byte	BoolOobe	
30	1	db	byte	BoolNightly	
31	1	db	byte	BoolSchedulingNeeded	
32	1	db	byte	ScheduleCheckResult	

Рисунок 2 - Определение структуры

Эти поля заполняются из ключа реестра:

1. Command заполняет *LPWSTR* Command/CommandByteLen
2. Maintenance устанавливает **BoolMaintenance**
3. Nighlty устанавливает **BoolNightly**
4. Oobe устанавливает **BoolOobe**
5. Sku устанавливает **DWORD DWSKU**
6. SchedulingNeeded устанавливает **BoolSchedulingNeeded**

Для установки задания в планировщике, производится дополнительное чтение реестра, но для наших целей это не представляет интереса.

Теперь мы дошли до самой веселой части! Указанная команда загружается в буфер:

```
char command[520] = {0};
StringCchCatW(command, 260, L"%ls %ls%hs", this->CommandStr,
L"-cv", <Some Random looking string>);
```

В зависимости от режима запуска/использования планировщика, -oobe или -fullsync добавляются к командной строке. В конечном счете, команда передается в качестве второго параметра **CreateProcessW**, что эквивалентно выполнению команды в шелле.

Также стоит отметить, что **CompatTelRunner.exe** блокируется, пока не завершится запускаемый процесс, поэтому не забудьте принять это во внимание, когда будете добавлять программы в этот ключ.

Почему это важно?

Атакующие всегда ищут новые и интересные пути закрепления в сетях. На сегодняшний день, я не видел, чтобы подобный способ с телеметрией был задокументирован. Blue team будет полезно добавить данную ветку реестра в Sysmon или другие инструменты мониторинга. Red team сможет использовать новую технику в сетях клиентов. Способ также может использоваться в качестве тривиального повышения привилегий Admin -> System.

Сейчас я не могу назвать это багом в безопасности системы. Для модификации ключей реестра нужны админские права. Гораздо в большей степени, данный способ показывает мне наличие новых и интересных мест в Windows. Когда вы думаете, что все уязвимости уже найдены, то всегда найдется что-то новое.