We run Powershell scripts on a Windows machine, and use Event Tracing for Windows (ETW) logs to collect corresponding logs. Event Tracing for Windows (ETW) provides a mechanism to trace and log events that are raised by user-mode applications and kernel-mode drivers. ETW is implemented in the Windows operating system and provides developers a fast, reliable, and versatile set of event tracing features.

ETW providers can raise events and can publish them to the Windows Event Log or can write their events to an ETW session, which gets written to a trace file or delivered to real-time consumer. Events are entities that describe interesting occurrences within the system and are defined by a set of attributes that are determined by the ETW providers.

We use four ETW providers to collect logs: process, thread, network, and registry. These logs are used to generate a graph with nodes and edges. The nodes are divided into five categories: "File", "Registry", "Network", "Process", and "Thread". The edges correspond to the features/events occured between two nods. The following are the events and their associated properties for the four providers we use.

Attributes which are common in all providers are:

* Size : Total number of bytes of the event. **Size** includes the size of the header structure, plus the size of any event-specific data appended to the header. On input, the size must be less than the size of the event tracing session's buffer minus 72 (0x48).
* HeaderType
* Flags
* EventProperty
* ThreadId : The TID of the thread that logged the event.
* ProcessId : The PID of the process that logged the event.
* TimeStamp : A 128-bit value which contains the time the event was logged in 100ns units since Jan 1, 1601.
* ProviderId : GUID’s of provider
* Id : A 16-bit integer that identifies the event type within the provider.
* Version : The 8-bit integer that is the version of the event type. An event’s

structure is defined by the provider, the event id and the version.

* Channel : An integer identifying the channel to which the event should be directed.
* Level : An integer between 0 and 255 that is used to indicate the severity or importance of the event. Lower values are considered of greater criticality. Event providers and viewers can be configured log or show events based on event levels. Windows predefines levels 0 through 5 as Log Always, Critical, Error, Warning, Informational and Verbose.
* Opcode : An integer in the range 0-255 that identifies the operation pertaining to which the event is logged. Providers may define opcodes in the range 11-238. The remaining are predefined by Windows with specific intended semantics. Providers may use these as well when writing events. When a provider uses tasks, it can define per-task opcodes as well. These task level opcodes follow the same range restrictions
* Task : An integer that identifies a specific logical component within the event provider
* Keyword : A bit mask used for grouping events of a similar type. The semantics of each bit is defined by the provider. Keywords can be used to control which events are written by a provider.
* KernelTime : The elapsed execution time for kernel-mode instructions in 100ns units
* UserTime : The elapsed execution time for user-mode instructions in 100ns units.
* ActivityId : The optional *ActivityId* argument tags the event with an identifier that other events may refer to as their related activity. This argument can be **NULL** to stand for whatever activity identifier is current set for the calling thread.
* Task Name : A human-readable, localized string corresponding to the integer task field.

Kernel-PROCESS provider attributes are:

Task Name:

* CPUBASEPRIORITYCHANGE/ CPUPRIORITYCHANGE
  + OldPriority
  + NewPriority
  + ProcessID
  + ThreadID
* IMAGELOAD/ IMAGEUNLOAD
  + ImageBase : Base address of the application in which the image is loaded
  + ImageSize : Size of the image being loaded.
  + ImageCheckSum
  + **TimeDateStamp :** Time and date that the image was loaded or unloaded
  + DefaultBase : Default Base Address
  + ImageName : Path to the executable file of the process
* IOPRIORITYCHANGE
  + ProcessID
  + ThreadID
  + OldPriority
  + NewPriority
* JOBSTART/ JOBTERMINATE
  + Container ID
  + Job ID
  + StatusCode
* PAGEPRIORITYCHANGE
  + ProcessID
  + ThreadID
  + OldPriority
  + NewPriority
* PROCESSFREEZE
  + FrozenProcessID
  + CreateTime
* PROCESSSTART
  + ProcessID
  + ProcessSequenceNumber
  + CreateTime
  + ParentProcessID : Unique identifier of the process that creates this process. Process identifier numbers are reused, so they only identify a process for the lifetime of that process.
  + ParentProcessSequenceNumber
  + SessionId : Unique identifier that an operating system generates when it creates a new session. A session spans a period of time from log on until log off from a specific system
  + ProcessTokenElevationType
  + ProcessTokenlsElevated
  + MandatoryLabel : (new to Win10) In addition to each objects Discretionary Access Control List (permissions on a file) Windows also enforces Mandatory Integrity Control (MIC) over object access attempts which compares the object's integrity label to the the integrity level on the process trying to access the object.  This field documents the integrity of the process which is determined from the user integrity level and the file integrity level of the EXE.
  + ImageName
  + ImageCheckSum
  + TimeDateStamp
  + PackageFullName : Specifies the full name of the AppX package that contains the application to be prelaunched in debug mode
  + PackageRelativeAppId : Specifies the application ID of the application within the AppX package that is prelaunched. The application ID is found in the package manifest file.
* PROCESSSTOP
  + ProcessID
  + ProcessSequenceNumber
  + CreateTime
  + ExitTime
  + ExitCode : Exit status of the stopped process.
  + TokenElevationType: This is useful for detecting when users running under User Account Control consent to running a program with admin authority. Token elevation is about User Account Control
  + HandleCount : Count of used handles.Handles are data structures that represent open instances of basic operating system objects applications interact with, such as files, registry keys, synchronization primitives, and shared memory.
  + CommitCharge : In computing, **commit charge** is a term used in [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) [operating systems](https://en.wikipedia.org/wiki/Operating_system) to describe the total amount of [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) of all processes that must be backed by either [physical memory](https://en.wikipedia.org/wiki/Physical_memory) or the [page file](https://en.wikipedia.org/wiki/Paging)
  + CommitPeak : The maximum number of pages that were simultaneously in the committed state since the last system reboot.
  + CPUCycleCount : The highest-resolution, with the frequency of the CPU, but may be unreliable depending on the CPU model and the thermal and power modes.
  + ReadOperationCount : Number of read operations performed
  + WriteOperationCount : Number of write operations performed
  + ReadTransferKiloBytes : Amount of data read
  + WriteTransferKiloBytes : Amount of data write
  + HardFaultCount
  + ImageName
* PSDISKIOATTRIBUTE
  + Job ID
  + DiskIoAttribution
* PSIORATECONTROL
  + Job ID
  + IORateControl
  + MaxIops
  + MaxBandwidth
  + MaxTimePercent
  + ReservationIops
  + ReservationBandwidth
  + CriticalReservationIops
  + CriticalReservationTimePercent
  + SoftMaxIops
  + SoftMaxBandwidth
  + SoftMaxTimePercent
  + ControlFlags
  + VolumeName
  + StatusCode
* THREADSTART / THREADSTOP
  + ProcessID
  + ThreadID
  + CycleTime (Only associated with THREADSTOP)
  + StackBase : Base address of the thread's stack
  + StackLimit : Limit of the thread's stack
  + UserStackBase : Base address of the thread's user-mode stack.
  + UserStackLimit : Limit of the thread's user-mode stack
  + : Memory address at which the trace starts
  + Win32StartAddr : Starting address of the function to be executed by this thread
  + TebBase : Thread environment block base address
  + SubProcessTag : Identifies the service if the thread is owned by a service; otherwise, zero.
* THREADWORKONBEHALFUPDATE
  + OldWorkOnBehalfThreadID
  + NewWorkOnBehalfThreadID

Kernel-FILE provider attributes are:

Task Name:

* CLEANUP/CLOSE
  + Irp : IO request packet. This property identifies the IO activity.
  + FileObject : Match the value of this pointer to the FileObject pointer value in a [FileIo\_Name](https://docs.microsoft.com/en-us/windows/win32/etw/fileio-name) event to determine the file involved in the I/O operation.
  + FileKey : To determine the directory name, match the value of this property to the FileObject property of a [FileIo\_Name](https://docs.microsoft.com/en-us/windows/win32/etw/fileio-name) event.
  + IssuingThreadId : The identifier of the issuing thread.
* CREATE/ CREATENEWFILE
  + Irp
  + FileObject : Identifier that can be used for correlating operations to the same opened file object instance between file create and close events.
  + IssuingThreadId : The identifier of the issuing thread
  + FileName : Pattern specified for directory enumeration/ Full path to the file, not including the drive letter
  + CreateOptions : Values passed in the CreateOptions and CreateDispositions(Specifies what to do, depending on whether the file already exists,) parameters to the NTCreateFile function. The options to be applied when creating or opening the file, as a compatible combination of the flags
  + CreateAttributes :
  + ShareAccess : Value passed in the *ShareAccess* parameter to the [NtCreateFile](https://docs.microsoft.com/en-us/windows/win32/api/winternl/nf-winternl-ntcreatefile) function.
* DELETEPATH
  + Irp
  + FileObject
  + FileKey
  + ExtraInformation : For FileDispositionInformation requests, this field contains the requested disposition. For FileEndOfFileInformation and FileAllocationInformation requests, this field contains the specified file size.
  + IssuingThreadId : The identifier of the issuing thread
  + InfoClass : Requested directory enumeration information class.
  + FilePath
* DIRENUM/DIRNOTIFY
  + Irp
  + FileObject
  + FileKey
  + IssuingThreadId
  + Length : Size of the query buffer, in bytes.
  + InfoClass : Requested directory enumeration information class.
  + FileIndex : File index from which to continue directory enumeration
  + FileName : Pattern specified for directory enumeration.
* FSCTL
  + Irp
  + FileObject
  + FileKey
  + IssuingThreadId
  + InfoClass
* NAMECREATE
  + Irp
  + FileObject
  + FileName
* NAMEDELETE
  + FileName
  + FileKey
* OPERATIONEND
  + Irp
  + ExtraInfo : For FileDispositionInformation requests, this field contains the requested disposition. For FileEndOfFileInformation and FileAllocationInformation requests, this field contains the specified file size.
  + Status
* QUERYEA/QUERYINFORMATION/QUERYSECURITY/SETDELETE/SETINFORMATION
  + Irp
  + FileObject
  + Filekey
  + ExtraInformation
  + IssuingThreadId
  + InfoClass
* READ/WRITE
  + ByteOffset : Starting file offset for the requested operation. Byte offset from the beginning of the physical disk.
  + Irp
  + FileObject
  + FileKey
  + IssuingThreadId
  + IOSize : Number of bytes requested.
  + IOFlags : IO request packet flags for this operation.
  + ExtraFlags

Kernel-REGISTRY provider attributes are:

Opcode/Task Name :

* 32( CreateKey) / 33 (OpenKey) Creates the specified registry key. If the key already exists in the registry, the function opens it.
  + BaseObject
  + KeyObject
  + Status : NTSTATUS value of the registry operation.
  + Disposition
  + BaseName
  + RelativeName
* 35 (QueryKey) Retrieves information about the specified registry key.
  + KeyObject
  + Status
  + InfoClass
  + DataSize
  + KeyName :
  + CapturedDataSize
  + CapturedData
* 38 (QueryValuekey) Retrieves the data associated with the default or unnamed value of a specified registry key.
  + KeyObject
  + Status
  + InfoClass
  + DataSize
  + KeyName
  + CapturedDataSize
  + CapturedData
  + ValueName = The full registry path of the key, beginning with a valid registry root, such as "HKEY\_CURRENT\_USER".
* 36 (SetValueKey) Sets the data for the specified value in the specified registry key and subkey.
  + KeyObject
  + Status
  + Type
  + DataSize
  + KeyName
  + CapturedDataSize
  + CapturedData
  + PreviousDataType
  + PreviousDataSize
  + PreviousDataCapturedSize
  + PreviousData
* 39 (EnumerateKey) / 40 (EnumerateValuekey) Enumerates the subkeys/value of the specified open registry key.
  + KeyObject
  + Status
  + Index : The subkey index for the registry operation (such as EnumerateKey).
  + InfoClass
  + DataSize
  + KeyName
  + CapturedDataSize
  + CapturedData
* 42 (SetInformationKey)
  + KeyObject
  + Status
  + InfoClass
  + DataSize
  + KeyName
  + CapturedDataSize
  + CapturedData
* 44 (CloseKey) Closes a handle to the specified registry key.
  + KeyObject
  + Status
  + KeyName

Kernel-NETWORK provider attributes are:

Opcode/Task Name :

* 10 (DataSent) / 11 (DataReceived) / 13 (Disconnectissued) / 14 (Dataretansmitted) /16(Reconnetattempted) /18(Protocolcopieddataonbehalfofuser) /42 (DatasentoverUDPprotocol) / 43 (DatareceivedoverUDPprotocol)
* 12 (ConnectionAttempted)/15 (Connectionaccepted)