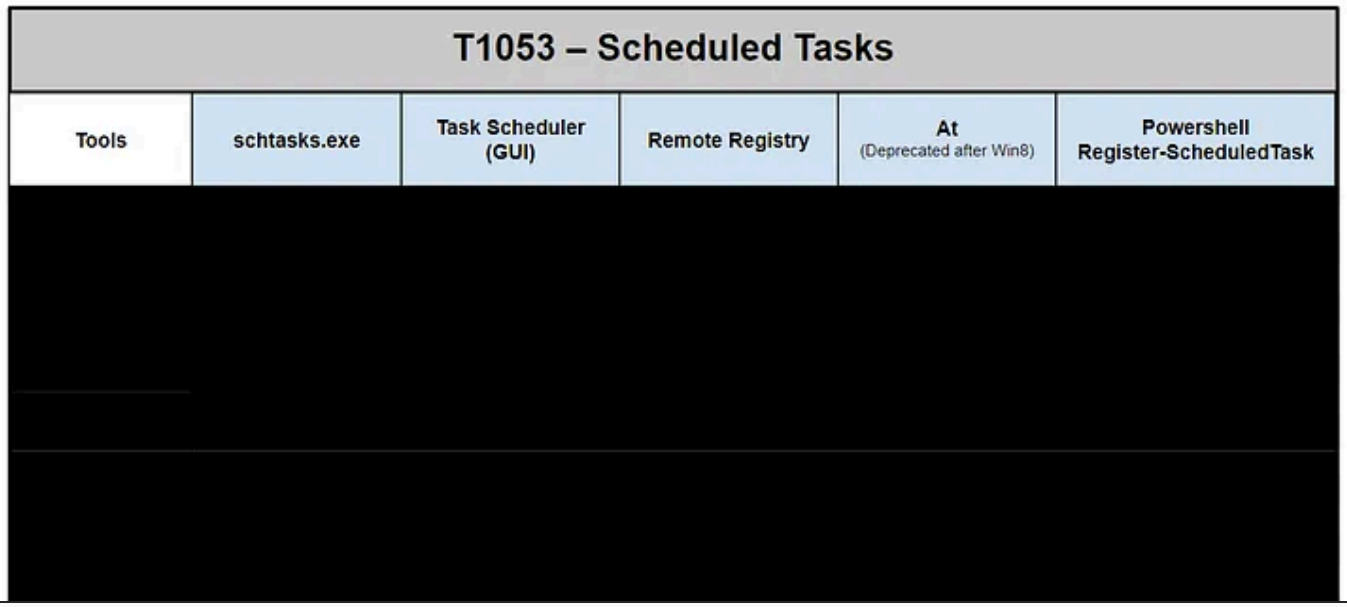
# Abstracting Scheduled Tasks

Jonathan Johnson · Follow

Published in Posts By SpecterOps Team Members · 9 min read · Mar 15, 2021

privilege escalation in some cases where the attacker can control the target of the trigger itself (i.e. the binary on disk which the scheduled task will execute) or if they can control a task which runs as a more privileged user.

Although this behavior has generally fallen out of favor for offensive use over the past number of years due to increased awareness and wide-scale deployment of detections, it is still actively used by numerous threat actors today, including in SUNSPOT, the implant used during the Solarwinds supply chain compromise. Our interest in this technique was revived after seeing it being actively used in ransomware campaigns with high degrees of success



boundary, is Remote Procedure Call (RPC) and scheduled tasks are no exception. Specifically, the Task Scheduler is backed by the Task Scheduler Remoting Protocol (MS-TSCH). This protocol is backed by three endpoints — ATSvc, SASec, and ITaskSchedulerService. The following screenshot shows RPC telemetry collected via Zeek where the `NetrJobAdd` method is invoked by a remote client and passed to the ATSvc endpoint on another host via its named pipe, `\\.\pipe\atsvc`.

```
index=Zeek source="/logs/zeek-logs/dce_rpc.log" endpoint=atsvc
```

```
PS C:\> Get-NtFile -Win32Path "\\.\pipe\atsvc" | select ServerProcessId

ServerProcessId
---------------
           5228


PS C:\> Get-WmiObject -Class Win32_Process | ? {$_.ProcessId -eq 5228} | select CommandLine

CommandLine
-----------
C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule


PS C:\> Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\Schedule\ | select DisplayName

DisplayName
-----------
@%SystemRoot%\system32\schedsvc.dll,-100
```

anything other than `0`, the DLL is loaded. After reviewing this function, we wanted to find the source of the value that is being checked.

If we look at the `JobStore` class, there is a function called `InitJobStore` which appears to populate a global instance of the `JobStore` (`m_pCommonStore`) with values. One of the values being populated is the string "`EnableAt`."

- Path ends with EnableAt
- Process name is svchost.exe

Sure enough, we can see `svchost.exe` querying the value in `EnableAt`.

We opened up the event's call stack and found that the call happens

Remember that offset that was checked when deciding whether to load `[redacted]`? Our original thought was that if `[redacted]` above is incremented to

We hit a bit of a sunk cost trying to chase down the condition for the load of `taskcomp.dll`, so we opted to operate with the knowledge that it is loaded into `schedsvc.dll` under *some* condition and started digging into `taskcomp.dll` to find out how it works.

Since we knew that the DLL was responsible for serving the named pipe, we simply searched for strings containing “atsvc” and found that the named pipe was created in the `StartRpcServer()` function.

When we looked at the cross references to this global variable, we found that

**Note:** The one exception is `NetrJobDel()` which doesn't check the global

WinDbg to check the value stored in the variable both when the registry key was set to `1` and `0`.

- The `NetrJob*()` RPC methods are used to interact with `taskcomp.dll`, so if the `EnableAt` value is set to `1`, correlating only to the execution of `at.exe` could miss executions
- Using `at.exe` to delete scheduled tasks is an edge case that should be covered as it could be considered benign but it subverts the `EnabledAt` restriction

## Conclusion