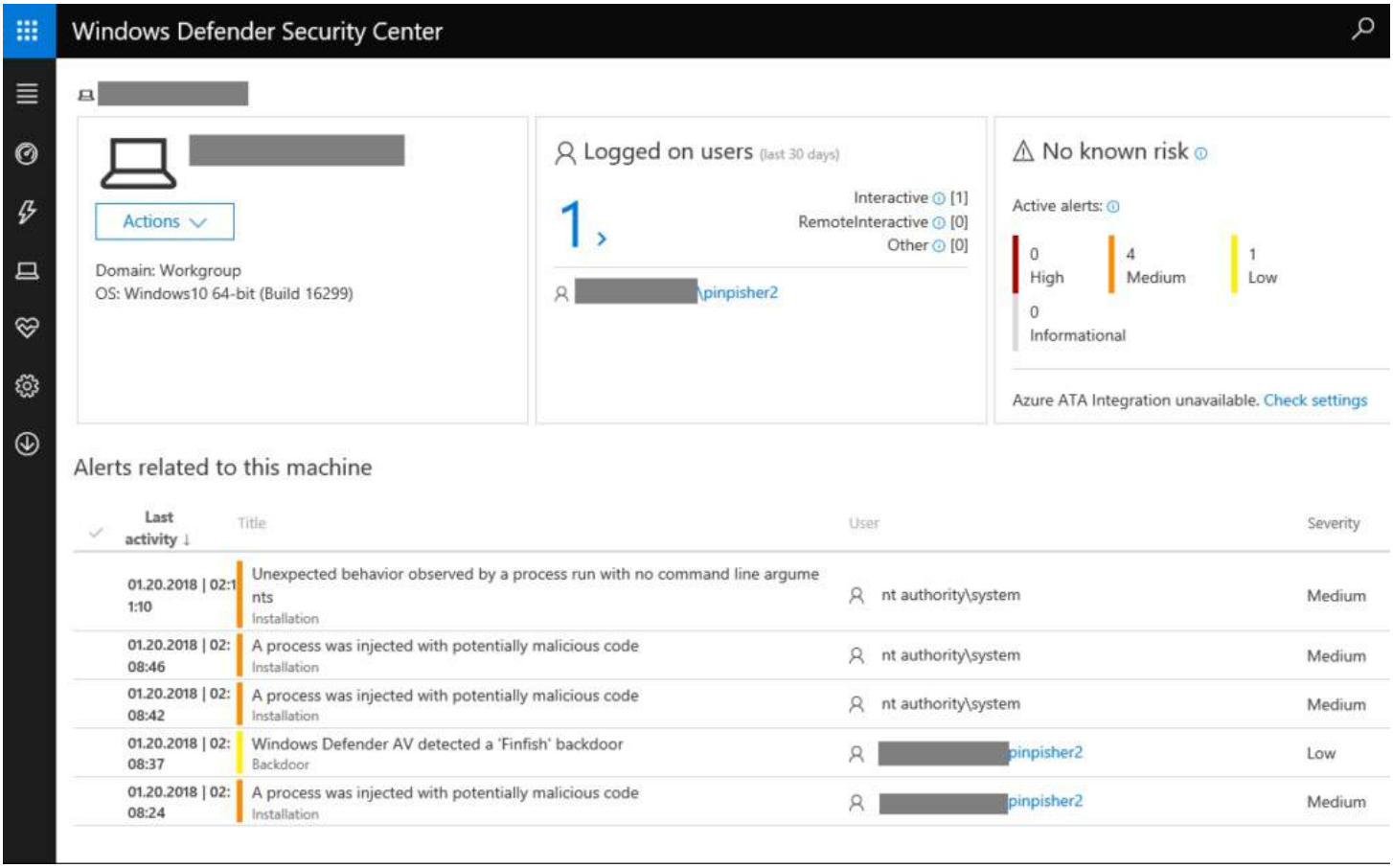

Figure 1. Generic Windows Defender ATP detections trigger alerts on FinFisher behavior

While our analysis has allowed us to immediately protect our customers, we'd like to share our insights and add to the growing number of published analyses by other talented researchers (listed below this blog post). We hope that this blog post helps other researchers to understand and analyze FinFisher samples and that this industry-wide information-sharing translate to the protection of as many customers as possible.

Spaghetti and junk codes make common analyst tools ineffective

In analyzing FinFisher, the first obfuscation problem that requires a solution is the removal of junk instructions and "spaghetti code", which is a technique that aims to confuse disassembly programs. Spaghetti code makes the program flow hard to read by adding continuous code jumps, hence the name. An example of FinFisher's spaghetti code is shown below.

```
loc_401FFD:

          push    eax
          jp      loc_401EE0
          jnp     loc_401EE0

loc_401EE0:

          push    ecx
          ja      loc_401B77
          jbe     loc_401B77

loc_401BC4:

          call    ReadDwordFromDataSect
          test    eax, eax
          jz      loc_401A0F
          jnz     loc_401A0F

loc_401A0F:

          jnz     loc_4020B1
          jz      loc_401EF4
          jnz     loc_401EF4
```

Figure 2. The spaghetti code in FinFisher dropper

This problem is not novel, and in common situations there are known reversing plugins that may help for this task. In the case of FinFisher, however, we could not find a good existing interactive disassembler (IDA) plugin that can normalize the code flow. So we decided to write our own plugin code using IDA Python. Armed with this code, we removed this first layer of anti-analysis protection.