Implant Proposal

Command and Control Design:

Master Server Location

Master Server will be hosted on a VPS through a provider such as Linode (affordable options that may have free trials). In an ideal scenario, a bulletproof VPS would be used; however, bulletproof hosters are much less affordable. Any payment (if needed) for the server will be separated from any operator(s) by using prepaid cards.

Communication Medium/Proxy

As can be seen in the diagram below, commands will be uploaded to a pastebin, which will be encoded in a story of sorts. The implant will parse the inputs from the pastebin story and take action accordingly. Detailed command execution will be in another pastebin that's encoded within the story as well. This secondary pastebin will be different each time and will destruct after being accessed.

Files will be uploaded through file sharing sites that do not require accounts such as sendgb/swisstransfer–both of which have an email sharing function. When going from C2 to implant, the files will be accessed by link and have limited download count (1 unless situation calls for more). When exfiltrating files from the implant, it will be uploaded to the file sharing site and immediately sent to a burner email which the C2 server will then access and download the file. These emails have the option of being hardcoded before each implant or encoded within the pastebin story, most likely the latter option will be utilized.

Operator connection to the C2 server will be through TOR so that likelihood of operator being compromised is lessened.

In addition, the operator(s) may access the pastebin through TOR; however, this will not be the primary method of uploading commands and is reserved primarily for if the master server is compromised and the implant needs to be destroyed.

Communication Encoding

Encoding scheme will be done with keywords in the pastebin which will then be parsed by the implant. Additionally, any commands that cannot be encoded with keywords (such as the detailing pastebin for command execution), will be encoded with either base64 or with a password that is detailed in the primary pastebin.

Diagram of Communication(s)

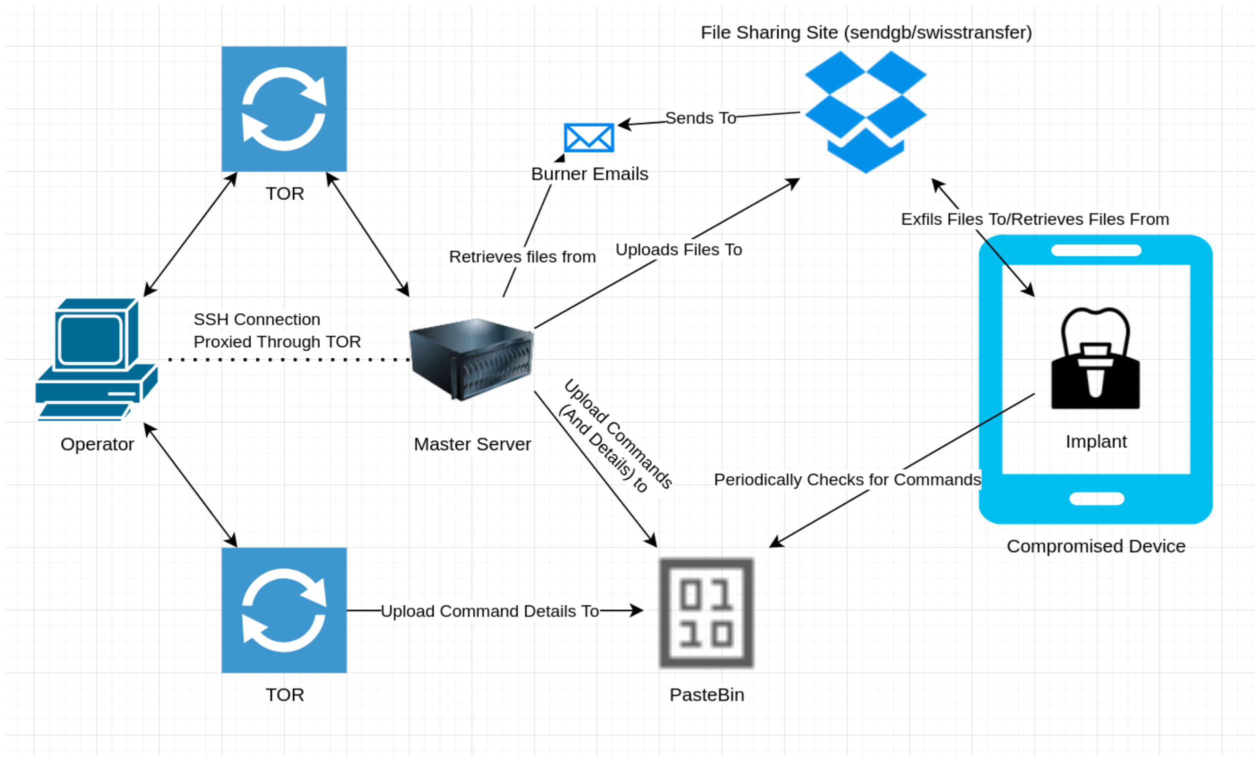

Diagram displaying the overall communication between systems

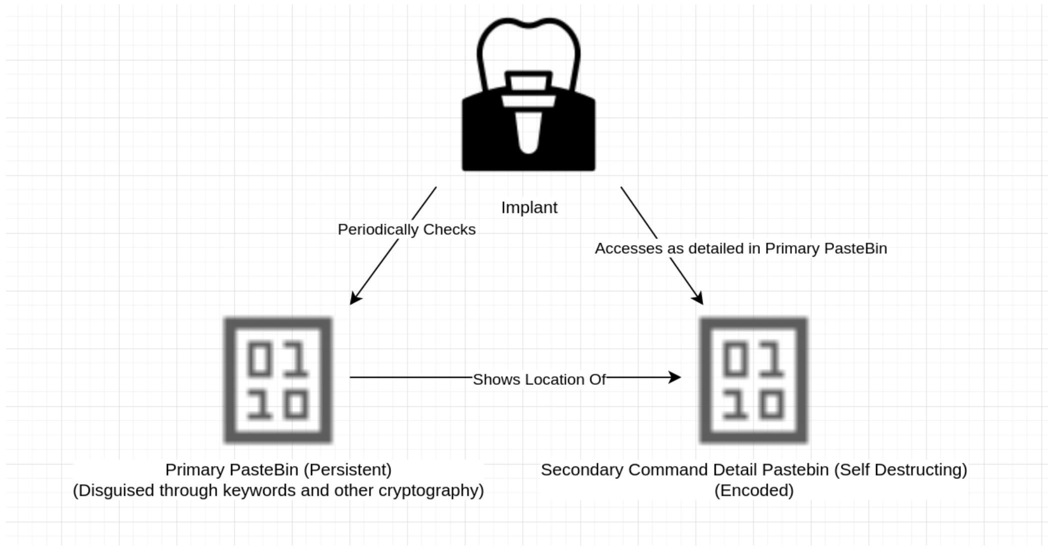
Implant
Periodically Checks
Accesses as detailed in Primary PasteBin
Shows Location Of
Primary PasteBin (Persistent)
(Disguised through keywords and other cryptography)
Secondary Command Detail Pastebin (Self Destructing)
(Encoded)

Diagram clarifying the structure of how the two pastebin setup will work

Implant Design:

Distribution Method

Distribution can be achieved through a number of methods depending on the target device. For example, if the target device is an android phone, social engineering could be used to convince a target to install an APK from a third party app host; this APK would contain the implant within it. (The details of the cover program for the APK are not important and could be targeted based on the target). The implant could also be installed if an exploit allows for files to be uploaded and executed as seen with the midterm, in this case the file could be uploaded as a binary and executed.

Evasion Techniques

Under ideal scenarios the target will install the implant themselves in some application that they were convinced to install. However, if that is unable to happen, there are a few things that may be done:

- If possible, the implant will be installed in places in which are unlikely to be searched.
- In addition it may be hidden. (See Additional Functionality: Hiding Itself) for more detail.

Base Functionality

File Exfiltration:

Files will be uploaded to file sharing sites like sendgb or swisstransfer. Files will be sent to a burner email (by the file sharing site) as instructed by the commands in the pastebin. Likewise, files can be retrieved based on a URL from sendgb or swisstransfer as directed in the primary pastebin.

Directory Traversal:

The implant will interface with the system for directory traversal. Since it will likely be implemented in C++, the system() function would be used for this to execute sh/bash/batch depending on the OS.

Command Execution:

Commands will be encoded in the primary paste bin cover story, the implant will then parse and respond according to the commands coming from the pastebin.

Self Destruction:

Upon receiving an indicator from the master server, the implant will delete itself. In addition, the implant will check for any active tools that may compromise it (such as Ghidra or Frida) and attempt to kill the process. If killing the process fails it will self-destruct and kill its own process.

Target Platforms

Core implant operations will be coded outside of an application environment in a language that has wide support cross-platform such as C++. This means it will be easily ported to specific environments such as into an android APK or packaged as a binary for various architectures and systems.

Persistence Mechanism

The implant will implement the necessary checks and calls based on the OS. For instance, on linux devices, the implant will create a cronjob (if non-existant) associated with executing the implant. On a packaged APK version, the implant will listen for the "BOOT_COMPLETED" broadcast.
On demand, may allow for the implant to create a hidden copy of itself. See Additional Functionality: Hiding Itself for more detail.

Additional Functionality

What additional actions do we want our implant to perform on device?

Pastebin Pivoting:

In the case that there is suspect of the primary pastebin being compromised, a command can be issued to pivot to a new primary pastebin. This would also essentially allow for new pastebins to be issued as desired, allowing for the primary pastebin to be changed between each command. It would be; however, essential to keep track of what the implant perceives as the primary paste bin.

Hiding Itself:

Implant may make a copy of itself and hide itself when issued a command from C2 via the primary pastebin. This will be done if it is suspected that the target wants to uninstall the implant (if they installed it themselves) or if it may be detected soon.

Permanent Death:

Implant will delete itself from the disk and kill its process.

Defense/Obfuscation Mechanisms

Messages to the implant will be embedded in a pastebin "story". The story will be edited at specific points to send commands to the implant. The implant will parse the story and search for specific signals within the story. In addition, lengthy commands, such as shell commands, will have a secondary pastebin that contains the full command. This secondary pastebin will be encoded, the implant will decode this as the pastebin "story" command suggests.

Upon receiving an indicator from the master server, the implant will delete itself. In addition, the implant will check for any active tools that may compromise it (such as Ghidra or Frida) and attempt to kill the process. If killing the process fails it will self-destruct and kill its own process.

On demand, may allow for the implant to create a hidden copy of itself. See Additional Functionality: Hiding Itself for more detail.