Hidden VNC for Beginners

By Marcus Hutchins

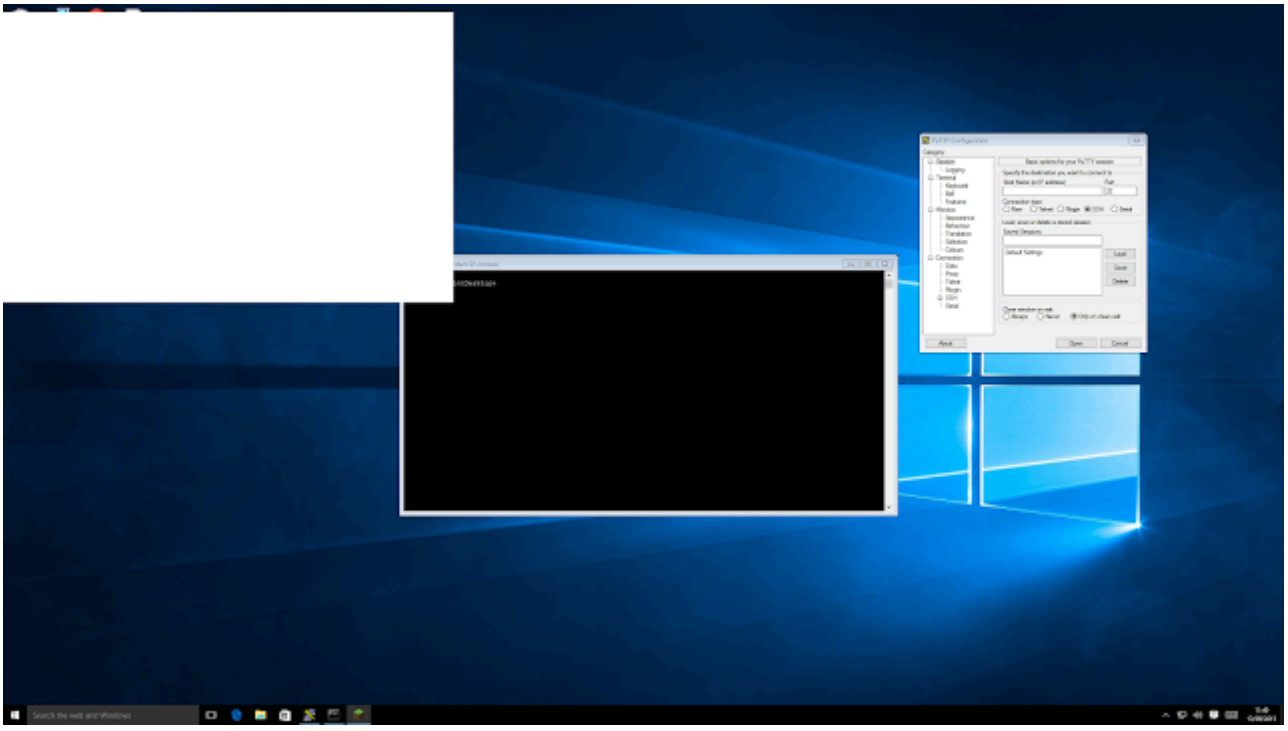
Published: 2015-09-13 · Archived: 2026-04-06 00:19:50 UTC

Hidden VNC is a creative solution to a solution to a problem which stemmed from banking fraud. Back years ago when fraud was uncommon, most banks only had basic IP or Geo-location checks to flag or block accounts if someone logged in from another computer. To combat this, banking trojans would run a SOCKS proxy server on the victims computer, allowing the fraudster to access the victims bank account with the same IP. As fraud became more prominent, banks started coming up with proprietary fraud detection systems which fingerprint the user's systems using a variety of check (Browser, OS/Plugin versions, locale, timezone, etc). The blackbox nature of these systems would require a fraudster to pretty much replicate the victim's system configuration in order to be sure the account wouldn't get blocked, so a more convenient method of fraud had to be found, that method was of course VNC. Fraudsters could VNC into a victims computer and use it to log into their bank account, but obviously this wasn't ideal. If the victim was using the computer, they'd see what the fraudster was doing, and if they weren't, the computer would probably be turned off. What was needed was some kind of VNC software that allowed fraudsters to access the system discretely, at the same time as the victim was using it.

Hidden Desktop Malware can make use of some little known Windows features such as CreateDesktop and cross-process window subclassing to implement an invisible environment for VNC to run. As most linux users will probably be familiar with, a lot of distros have the ability to run multiple simultaneous desktops with independent taskbars. Windows has had this ability to crate multiple desktops since 2000, but it's not a well known feature and there is no default application to make use of it. By calling CreateDesktop, software can create a hidden desktop and execute applications in the desktop's context. All applications running on the hidden desktop will be invisible to the other desktops (i.e. the one the victim is using), they will not even show in the taskbar outside of the hidden desktop. Sounds simple enough, right?

Screenshots Most VNC software works by taking periodic screenshots and sending them back to the client; however, Windows does not render any GUI elements to desktops which are not active (currently displayed on the monitor). One can't simply just capture screenshots of the hidden desktop, instead the VNC server would have to call EnumDesktopWindows to get a list of windows running on the hidden desktop, then call PrintWindow on each individual window, writing them to a bitmap in reverse Z-Order (starting with the bottom-most window and working its way to the top-most). Essentially the server is just emulating the screenshot feature by rendering each individual window to a bitmap in reverse of the order they appear on the screen.

Unfortunately, some applications don't properly handle WM_PRINT or WM_PRINTCLIENT messages (sent by PrintWindow), and as a result all or parts of the application will display as a white rectangle, as shown below.



To resolve this, the VNC server would need to implement WM_PRINT and WM_PRINTCLIENT message on behalf of the application, making sure it paints all visible elements to the buffer. This can be done by either injecting code into all processes and hooking various functions in user32.dll, or by using cross-process subclassing to give the VNC server the ability to process window messages destined for the target application from within the VNC process.

User Input When it comes to user input, the server has to emulate a virtual keyboard / mouse as input would be sent to the active desktop, not the hidden one. Normally when the mouse is moved or clicked the VNC client would sent the position along with a button click event to the VNC server, which would move the mouse to the given position and simulate a click, but because the real keyboard and mouse can't be use, things are far more complicated. The VNC server would have to keep track of every window on the hidden desktop, it's location, and its Z-Index; When a click even is sent, the server would need to find which window is at the cursor's current position by enumerating each window and checking its coordinates and visibility, then use PostMessage to send it a click event. For keyboard the same is true, the VNC server needs to keep track of which window is currently focused and use PostMessage to direct the input towards it.

Conclusion

Hidden VNC is probably one of the most complicated malware features to code and essentially requires coders to implement their own window manager, which is why there are very few unique implementations in the wild (most malware uses a single implementation unimaginatively named HVNC).

Sadly due to the fact it's a very sought after feature for banking trojans, I'm not going to post proof of concept code; however, I've uploaded some example code to demonstrate creating and switching between multiple desktops, you can find it here: https://github.com/MalwareTech/CreateDesktop/

Source: https://www.malwaretech.com/2015/09/hidden-vnc-for-beginners.html