

Shell Enhancers

They created magic, enabling graphical Internet access over shell accounts. Though the shell days have gone by, here's how they worked...

When I showed my sister a Website for the first time back in 1996, she was disappointed. A black-and-white screen with oodles of text was not what she had in mind after having seen me gush about it all day long. We, like most people, had purchased a shell account from VSNL to begin with. Now, with heavy cuts in Internet access charges for TCP/IP accounts, I can safely say that shell accounts are in the last days of their existence. If we look back, we find that amidst the black-and-white VT100 terminals, there were points marked by some "magic" utilities that supposedly converted shell accounts into PPP accounts.

Shell account users everywhere were brought to the edge when these utilities made an appearance, for they were getting much more than they paid for.



VSNL on the other hand, played sheriff and tried to block these programs as quickly as possible. Some of these programs fizzled out and some continued to use new tricks to accomplish their motive.

Shell account: A nerd's eye view

Most people are satisfied with the definition that a shell account is a non-graphical Internet account. Actually, shell access to the Internet today is reminiscent of IBM SNA (Systems Network Architecture) in the 1970s. This was a centralized computing environment in which unintelligent terminals were connected to a central mainframe. These terminals had no processing power and simply displayed what the server (mainframe) asked them to, while all processing was done on the server.

Modern-day terminal protocols (such as VT100, VT102, etc) used with programs like HyperTerminal and Telix simply emulate these dumb terminals on your PC. With PPP access, your computer becomes a host on the Internet with its own IP address, but with shell access your computer simply acts as a dumb terminal and displays what the terminal server (giasdla, giasbma, giasbga, etc) tells it to.

For example, if you asked for a document, say, default.htm, from a server "pan.com" with PPP access, your Web browser would connect to the HTTP port of pan.com and send an HTTP request such as:

```
GET www.pan.com/default.htm
HTTP/1.1
Accept: image/gif, image/x-bitmap,
image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible;
MSIE 5.0; Windows 98; DigExt)
```

Pan.com would then send an HTTP reply along with the HTML file, which would then be interpreted and rendered on the screen by the browser.

On the other hand, if you were on a shell account and used "Lynx" to fetch the document, the server to which you

are logged on (such as giasdla, giasbma, giasbga) would send a similar HTTP request, receive the document, interpret it and simply send the result to your terminal. Your terminal will then display this result. So while with PPP access, your computer gets the actual source of the HTML, with a shell account it gets text with simple instructions on where and how to place it on the screen (through VT100).

This was the paradox. Your browser was capable of rendering HTML files graphically, if it had their source and the images embedded in them. The source could be obtained in more than one way from the shell server, and images could be downloaded like you download any other file. So effectively, all somebody had to do was to make a communications program that would sit between your browser and the shell server, take everything from the server in raw format (such as the source of the HTML files) and pass it on to the browser. Some software developers did take the initiative, and the result was programs like Blue Laser, Shellsock, Firecomm, and WAC.

How these programs worked

Telnet to port 80

Amongst all the techniques used, this was the easiest and the most efficient. Using this method, the program established a Telnet pipe between the main Webserver and the user. When the user typed a URL, say, www.microsoft.com/redhat.htm, the program trapped the HTTP request that the browser thought it was sending to microsoft.com. It then did "telnet microsoft.com 80" on the shell account and simply forwarded the request to it. So, eventually the request went through to microsoft.com just as it would've gone through a simple TCP connection. Whatever the server sent back (a 404 in this case) was passed on to the browser without any changes. This produced near-TCP/IP results.

This technique could also be extended to other protocols. A similar telnet pipe

could be established between port 110 (POP), 25 (SMTP), and 6667 (IRC). Thus, clients like mIRC, Pirc, Outlook Express, and Netscape Messenger could now be used on a shell account.

In fact, not only could you enhance the features that VSNL already offered, you could also use apps that VSNL never allowed. Two such apps were WHOIS and Finger. Although the binaries for these apps were not available on shell accounts, shell account enhancers let you use them by using a telnet pipe to the respective port.

With the immense support that Shellsock and other programs got, all this seemed too good to be true, and it turned out to be so. VSNL soon banned all telnets to port 80 of any host. This was done by replacing the telnet program with a simple script that disallowed port 80 to be used. However, since no other ports were blocked, one could still use programs like mIRC, Outlook, and Messenger.

With shell access your computer simply acts as a dumb terminal, and displays what the terminal server tells it to

Lynx-source

When port 80 was blocked, the authors of Shellsock found another way to get the source of an HTML file. Just type "Lynx -source <URL>" and you can get its source. However, using this was slightly tricky because HTTP requests had to be parsed and interpreted intermediately. The HTML returned was to be attached to an HTTP reply before being passed on to the browser. Similarly, doing "Lynx <image URL>" would retrieve images.

Unlike the telnet pipe that lasted for a long time, this option was blocked almost immediately after it was released.

"Slashing" your way through

In Lynx, there's yet another way to get images and the source of an HTML file.

All you have to do is press "\ " and you get the source of the current page. A list of images can be obtained by sending a "*" and an image can be selected using the arrow keys. Using this feature was even trickier than the previous one. Here, the program had to include a VT100 emulator which would filter out the Lynx menu, pauses like "press space to go to the next page" and other VT100 escape sequences before adding an HTTP reply header and passing it to the browser.

Either this method is too clumsy and inefficient, or because shell accounts are obsolete now, VSNL has never blocked this one. Yes, it still works. Programs like WAC (on this month's PCQ CD) and Firecomm still use it, although WAC uses another method (discussed later) which is more efficient.

Return of the telnet pipe

It was a simple answer to a simple question. If VSNL blocked telnets to port 80, then why not use a different port for HTTP and use a telnet pipe again? The problem of course would be to find free gateways that offered HTTP on ports other than 80, but one didn't have to look too far. There were free anonymizers on the Net which were actually meant to serve a different purpose, but couldn't have been custom-made better for us. These were HTTP proxy servers that let people surf anonymously. Since they ran on ports like 8080, 8888 and 23, they could be used exactly like the original port 80 telnet pipe. This method too hasn't been blocked and is currently being used by WAC.

D for download

There are even more ways of doing it. Take a look at the Lynx man pages in Linux, and you'll probably get your own ideas. For example, the "d" option in Lynx can be used to download a link. Bingo! You can use it to download an HTML, get its source, and send it to Netscape to be rendered. You can also download images with this method.

So you see, once the basic idea of what had to be done was clear,

The source code of some of the best shell enhancers is on the CD



implementing it was never a problem. Let's take a look at some of the programs that used these techniques.

The programs

Shellsock

Created by Jayakrishnan and Kurian from Xtendtech, Chennai, Shellsock came up with more versions and had a larger following than similar programs. It was also unique in its implementation. As opposed to other programs which shipped as local proxy servers, it shipped as a new implementation of "Winsock.dll" specially made for shell accounts. Shellsock not only allowed you to browse, but also allowed you to send e-mail and IRC using graphical clients. Its first version used the old telnet to port 80. Once port 80 was blocked by VSNL, the Lynx -source version was released. But Shellsock also had its chinks. First, it was a 16-bit app, so it had umpteen problems with Win 95. Also, if somebody had to use a TCP/IP account on his system, he would have to replace "Winsock.dll" with its original Windows version each time.

The Blue Laser

This was the first "shell account enhancer" ever released. It used the old telnet to port 80, but unfortunately disappeared soon after port 80 was blocked. This was set up as a proxy server, which means that if a user had a shell account as well as a TCP/IP account, switching between them was that much easier. The Blue Laser is written by P S Karthikeyan, again from Chennai (source code on the CD).

WAC

Released slightly late in the day, WAC

used "Slashing your way through", "D for download" and the "Return of the telnet pipe". It would automatically switch to another mode if one of these stopped working. WAC also set itself as a proxy server and allowed browsing, POP and SMTP mail, WHOIS, Finger and also graphical IRC. The binaries and source code of WAC can be found on this month's CD.

Firecomm

This one was written in Visual Basic and supported only browsing. It was based on "Slashing your way through". Firecomm had a good interface and was easy to use. It was an all-in-one package, which included the dialer and the browser in the same window. It had a special, meticulously-written VT100 emulator built in.

Finally, as shell accounts join vacuum tubes and acoustic couplers in the good old recycle bin, and before you say "thank god they're gone", I'd like to make one final remark. Despite the fact that shell accounts were slow and limited in their capabilities, I think they've served their purpose at a time when bandwidth was low and had to be used judiciously. Shell accounts provided the cheapest possible way of accessing the Internet. One can't blame VSNL for having blocked the programs that we've talked about, simply because the bandwidth that shell accounts were supposed to distribute was never meant to handle images and other binary data. Getting graphics was contrary to the very basic idea of having a shell account.

And as for our shell account enhancers, the fact is that none of them actually worked for more than two months. But they did bring about a realization amongst many developers in India that just by using resources in a slightly different manner and being innovative with them, you could get a lot more out of what you had. They also helped bring down Internet access charges to some extent.

Sapan J Bhatia

<sapan@altavista.net>