ORNL/TM-2021/2331

A technique to make an enterprise network a Darknet on the Internet, while providing required services to authorized users

Joseph Olatt

February 2022

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-2021/2331

DarkNet, Electrification and Energy Infrastructures Division

A technique to make an enterprise network a Darknet on the Internet, while providing required services to authorized users

Joseph Olatt
olattjv@ornl.gov

February, 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

1. PROBLEM DESCRIPTION

In a well-designed and secure enterprise network, the hosts inside the network are not directly accessible from the Internet. The enterprise firewall blocks direct access to hosts inside the enterprise network and also blocks any attempts to probe or discover information about those hosts from the Internet. However, access to the enterprise network from the Internet is a must in today's day and age. Hence, specialized mechanisms to allow secure access are implemented.

There are numerous tools available to securely access an enterprise network that is connected to the Internet. These tools include freely available VPN[*] offerings like OpenVPN[†], Wireguard[‡], OpenSSH[§] from the OpenBSD[¶] project, in addition to the many other free and commercial offerings.

The problem with all these tools is that the listening ports of these tools are visible in port scans and they are subject to brute force attacks[‖]. This is not by any means problems with the tools themselves. This is how computers networks are designed to provide service. A simple, easy and relatively quick scan of the listening ports on a networked computer, using a free tool like nmap[**], will provide a list of open ports on that computer. With that in hand, a perpetrator can initiate brute force attacks to attempt to guess the passwords securing the security system. Often, other computers on other parts of the Internet, not belonging to them but controlled by the perpetrators via malware, phishing, etc., are employed to launch distributed brute force attacks.

If the perpetrators succeed in guessing a password or somehow compromising the security system, it results in a breach. Even if the perpetrators don't succeed, their actions still flood the network with unnecessary data traffic and impose unnecessary load on the computer systems to the point where they impair legitimate use. In addition, a zero-day vulnerability in any of the tools used could potentially result in an exploit and breach even before a patch to fix it is available.

Figure 1 shows the number of daily brute force attacks on port 22 (SSH) of a jump server[††] to a small /28 Class C network that does not even have a domain name associated with it.

Since the listening ports of these enterprise network access mechanisms are easily discoverable, from the Internet, one can guess the existence of the enterprise network, and hence it is not a true "Darknet"[‡‡]. It would be if the listening ports can be hidden as well!

2. Standard Mitigation Techniques and their Shortcomings

Some of the standard mitigation techniques and their shortcomings are as follows:

- Fail2ban—blocking IP addresses by monitoring log files; fails to protect against a distributed brute-force attack and since intrusion prevention kicks in after some attempts, there is no protection

[*] https://en.wikipedia.org/wiki/Virtual_private_network

[†] https://openvpn.net/

[‡] https://www.wireguard.com/

[§] https://www.openssh.com/

[¶] https://www.openbsd.org/

[‖] https://en.wikipedia.org/wiki/Brute-force_attack

[**] https://nmap.org

[††] https://en.wikipedia.org/wiki/Jump_server

[‡‡] https://en.wikipedia.org/wiki/Darknet

https://en.wikipedia.org/wiki/Fail2ban

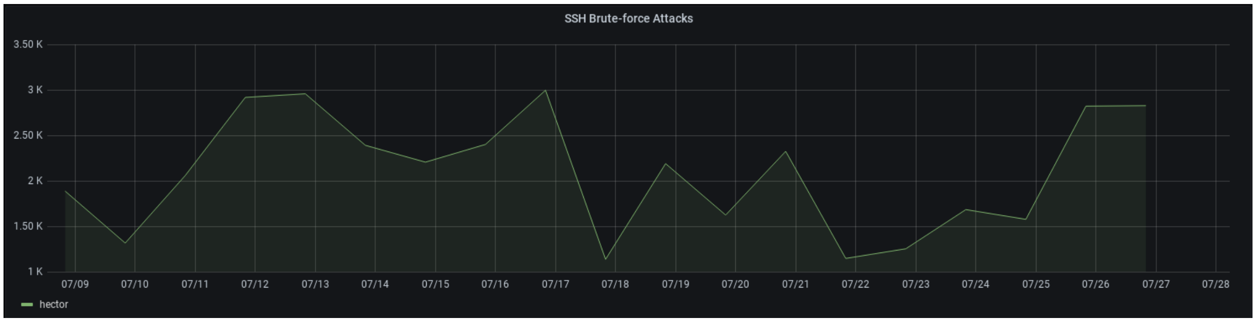

Figure 1. Daily SSH brute force attack counts

against zero-day vulnerabilities or carefully timed brute-force attacks

- DenyHosts—blocking IP addresses by monitoring log files. Has the same shortcomings as Fail2ban
- Port knocking—once the sequence of knocks is known, the secret is out
- Two-Factor Authentication—listening ports still visible in port scans and can be subjected to brute-force attacks and Denial of Service (DoS) attacks
- Non-standard port numbers—not too difficult to find out; once known, secret is out
- OpenSSH keys instead of passwords—a good solution; but listening ports still visible in port scans and can be subjected to brute-force attacks and Denial of Service (DoS) attacks

There are probably other good mitigation solutions. In which case, the technique described in this paper is yet another one.

3. Overview of Proposed Technique

- A base install of OpenBSD on a capable dual-homed computer server. Absolutely no third-party packages are installed. Only software available in the base install are used.
- The built-in Packet Filter (PF) firewall configured to only allow access to listening ports of remote-access tools like VPN software, SSH, etc. from IP addresses listed in a specific PF table.
- The built-in web server (httpd) and FastCGI protocol server (slowcgi) configured to run the custom CGI developed as part of this technique that implements the S/KEY algorithm.
- Custom command line program run by admin user to generate and populate the initial one-time password (OTP) in the built-in NDBM database.
- User's OpenSSH account with SSH public key or VPN access is set up.

https://en.wikipedia.org/wiki/DenyHosts
https://en.wikipedia.org/wiki/Port_knocking
https://en.wikipedia.org/wiki/Dual-homed
https://en.wikipedia.org/wiki/Common_Gateway_Interface
https://en.wikipedia.org/wiki/S/KEY
https://man.openbsd.org/ndbm.3

- When User wants to access the network remotely, user goes to web site and enters login ID.
- Web form automatically display editable IP address that should be granted access and prompts user to enter Nth OTP.
- If OTP is correct, the user's specific IP address is allowed to access the service ports; After 3 incorrect OTPs, user's account is locked.
- After enterprise-defined N hours/minutes, the IP address is removed from allowed list. The network will once again become "dark" to that IP address. The default is 8 hours.

4. Detailed Description

The general premise of the technique is based on defense in layers. A web form that requires OTPs as authentication tokens will permit/deny communication with the service ports (for example: 22, 1193, 51820, etc.). Authorized users will, just before initiating a connection to the service port, interact with this web form. The web form will request the user's login ID. It then directs the user to enter a specific OTP from the list of 25 active OTPs that were generated and sent directly to the user, via the web interface, when the user first interfaced with this web form using his/her initial OTP, or when all the OTPs in a given list were used up.

If the user enters more than 3 incorrect OTPs, then that user's account is locked. Unlocking can only be done by an authorized admin.

If the web form receives more than 40 requests in a 10 second interval from a particular IP address, then that IP address is blocked from accessing that web form. This is the default behavior and can be adjusted.

The S/KEY algorithm is based the principle of a one-way cryptographic hash function where it is easy to create a fixed size value of any given piece of data, but practically infeasible to reverse the computation (i.e. derive original data from fixed size hash value). The default algorithm used is MD5 with the OTPs being 32 characters long, in hexadecimal format. The choice of the algorithm (despite published collision weakness) and format was chosen so as to fairly easily facilitate conveyance of the initial OTP, from admin to user, via telephone and also because this system is not the primary authentication system.

When the last OTP from the current set of 25 OTPs is entered by the user, the web CGI will automatically stream a new set of 25 OTPs to the user. The user will save that set and use the first key from the new set to have his/her IP address allowed. This is to ensure that the user does not save the new set somewhere and not remember where it was saved, in the next go-around.

Once the user has supplied a correct OTP, that OTP will be stored in the database for the next usage. The next time the user will supply an OTP whose MD5 output will match the one that is stored in the database. The IP address that the web CGI detected (or, edited by the user in the web form) will be allowed to access whatever service is configured for remote access. If OpenSSH is the software that is being used for remote access, usage of SSH keys instead of password authentication is recommended for an extra level of security. If VPN is being used, then the user can proceed to interact with the VPN software.

A general session timeout of 8 hours is built into the system. This value can be adjusted. Upon expiration of this timeout, the allowed IP will be removed from the PF table and the user will have to redo the authentication with a new OTP.

With this setup, the entry point to the enterprise network or subnetwork remains “dark”. Port scans will only reveal the the web server port, which is in turn rate-limited with automatic blocking via the PF firewall.

5. Use Cases

- Electric grid substation networks
- Enterprise DMZ networks
- Enterprise networks

6. Security Features of Components Used

6.1 OpenBSD

The reasons for choosing OpenBSD as the operating system are:

- OpenBSD is widely regarded as being one of the most secure operating systems currently available.
- Touted as “secure by default” with only two remote holes in the base system install.
- Designed with focus on security, writing quality code, code audits, default secure and sane configurations, privilege separation, privilege revocation and chrooting; and eschewing complexity for simplicity and security.
- Integrated with cryptographic API and the Packet Filter firewall, all included in the base system install.
- Small, simple chrooted HTTP server, httpd, included in the base system install.
- A chrooted FastCGI protocol server, slowcgi, included in the base system install.
- Easy to install, configure, maintain, update and upgrade.

6.2 Web CGI, ancillary executables and shell scripts

- A single CGI displays the forms to collect the login ID and OTP. It is also responsible for streaming the set of 25 new OTPs, when required. The CGI is written in C/C++ and statically compiled so that it can run in a chrooted directory (/var/www/cgi-bin) without any other dependencies.
- Upon verification of OTP, the IP address passed through the web form is passed to a UNIX socket. Since the web server and CGI are running in a chrooted directory, it is unable to access the PF firewall controller program, pfctl, that resides in /sbin. With this approach an IP address is passed to

https://secureblitz.com/most-secure-operating-systems
https://allthatiswrong.wordpress.com/2010/01/20/the-insecurity-of-openbsd
https://man.openbsd.org/chroot
https://man.openbsd.org/pfctl

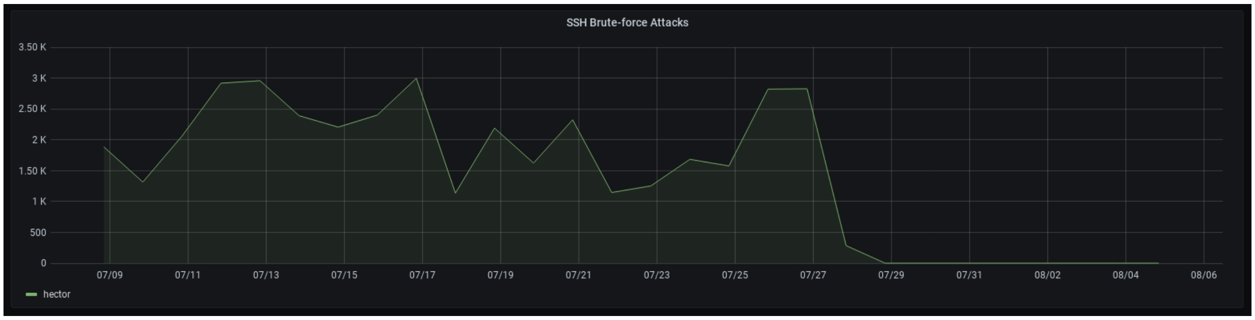

Figure 2. Drop in SSH brute force attacks after implementation of this technique

a separate user-level daemon, via the socket, which then calls a script that will only accept an valid IPv4 address as argument. The script is called with the privilege escalation command, doas. Special rules in /etc/doas.conf allows the non-privileged user, jb4mgr, to invoke that script. This way, jb4mgr, will not have carte blanche privilege to run the pfctl command.

- An additional compiled executable is invoked by the cron daemon on a periodic basis to remove expired IP address from the PF firewall table. It accepts an argument that represents the IP address expiration time in minutes.

6.3 Cryptographic hash function and seed values for OTP

- The MD5 cryptographic APIs that are part of the base system install are used to generate the OTPs.
- The arc4random is used to generate a seed for the MD5 hash functions.

7. Conclusion

Figure 2 shows a count of the SSH brute force attacks after this system was implemented on the afore-mentioned network.

Figure 3 shows a count of the SSH brute force attacks in around 4 months of implementation of this technique/solution.

This project is called Justbfore. It can be a mnemonic for: JUmp server Setup to Thwart Brute FORce Efforts. “jb4” is the short form. It also represents the action a user needs to take “just before” the user initiates remote access.

8. Acknowledgements

Thanks to Ketan Patel (patelkr@ornl.gov) for reviewing the general technical accuracy of the technique presented in this paper.

https://man.openbsd.org/doas
https://man.openbsd.org/arc4random

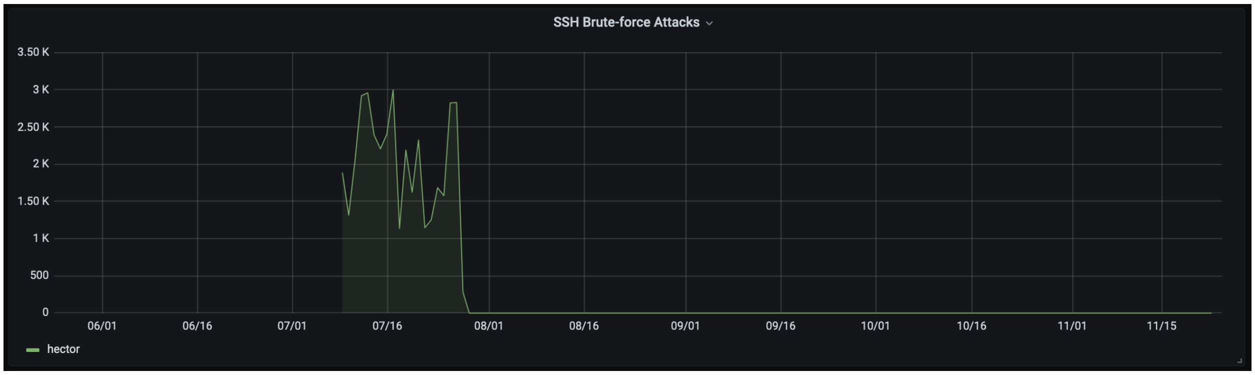

Figure 3. Daily SSH brute force attack counts over a 4 month period