Secure Logging in to EC2

INSIGHTS GATHERED FROM VARIOUS BLOGS AND POSTS:

1. **SSH/RDP Port**
2. **Access Control Lists (ACLs)**: inbound and outbound rules to allow only specific IP addresses or ranges to access certain ports on your EC2 instances
3. SSH Key Pairs: use SSH key pairs instead of password authentication.
4. Multi-factor Authentication (MFA)
5. **Bastion Hosts**: intermediary for SSH or RDP access to your EC2 instances
6. VPN Access: access the EC2 instances as if they were on the same network.

SSH BASED WORKAROUNDS:

> Use of bastion host and SSH tunnel through this bastion host to reach targeted ec2 instance from local

> SSH 22 and RDP 3389 ports are used for remote login, you can change the default ports to non-standard ports to reduce the visibility of your instances to automated scanning tools

>Gravitational teleport-ssh system that integrate with other services such as github auth and allows for ssh recording and connection through a web-ui

WAYS FOR SSH TO USE Fail2Ban OR sshguard

1. Ensure that root login is disabled.
2. Prevent password login for users on your machine. (This is especially important in the case where your target username is known)
3. Install a means of banning brute-force login attempts (fail2ban is a good option)
4. Change your server port number.
5. Whitelists are better than Blacklists. If you can tell your SSH server which IP's are allowed to connect, you can dramatically decrease attacks. (This can be complicated and cause problems in the future, so read and be prepared if you take it this far)

From <https://askubuntu.com/questions/77329/what-is-the-best-way-to-secure-my-ec2-server-in-regards-to-the-ssh>

HIGHLY RECOMMENDED PRACTICE: SETTING UP OF SSM FOR LOGGING IN

Logging in to an EC2 instance use SSM
No usage of SSH at all
SSM better, easier and more granular permissions structure, much better auditing for visibility

EC2 INSTANCE CONNECT

provides a simple and secure way to connect to your Linux instances using Secure Shell (SSH). Usage of AWS IAM policies and principals to control SSH access to your instances, removing the need to share and manage SSH keys.

On connecting an instance using EC2 Instance Connect,

>the Instance Connect API pushes a one-time-use SSH public key to the instance metadata where it remains for 60 seconds

>IAM policy attached to your IAM user authorizes IAM user to push the public key to the instance metadata

>The SSH daemon uses AuthorizedKeysCommand and AuthorizedKeysCommandUser (configured when Instance Connect is installed, to look up the public key from the instance metadata for authentication, and connect to the instance)

EC2 Instance Connect Endpoint is an identity-aware TCP proxy that enables secure connectivity to EC2 instances and other VPC resources from the Internet without the need for a bastion host or public IP addresses. It combines identity-based and network-based access controls, providing enhanced security and control over the connection process. The EIC Endpoint works with the AWS Management Console, AWS Command Line Interface (CLI), and popular client tools like PuTTY and OpenSSH.

From <https://www.linkedin.com/pulse/securely-connecting-private-ec2-instances-instance-connect-benis/>

EC2 instance connect
not an option to use it as a solution for instances running in a private subnet. Furthermore, it offers limited OS support. Currently, only Amazon Linux 2 or Ubuntu (16.04 or later) are supported, and Windows support is completely left out.

From <https://medium.com/swlh/aws-ec2-secure-shell-access-like-a-boss-55eaaf5e4ed2>

OTHER METHODS:

> Place ec2 instance into private subnet and add NLB or ALB : ALB- use its security group as a source for EC2 instance security and terminated SSL certificate on it

> AWS SSO + cognito

> Plan a well-defined threat model: systematically identify and categorize the info apps interact with, how it flows in sys

>Ssm+Sso

>Use of iptables- Linux's built in firewall management tool: restricts SSH access to EC2 instance from specific IP addresses or ranges

>The way to accomplish what you want is to use the IAM condition key ec2:osuser which restricts which local OS user you are allowed to push the public ssh key for.

Couple this with dynamic session/principal tags restricted via SCP and you have a very locked down set of actions.

From <https://www.reddit.com/r/aws/comments/zq3o51/ec2_instance_connect_impersonating_users/>

LINKS TO VARIOUS ARTICLES:

https://www.linkedin.com/pulse/exploring-ec2-instance-connect-ssh-connectivity-public-kasaudhan/
https://dzone.com/articles/hardening-an-aws-ec2-instance-1
https://www.linkedin.com/pulse/secure-connectivity-from-public-private-introducing-ec2-ashraful-alam/
https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html