Wicked Quick Git

Set up a docker-based git server in minutes. Uses Gitea for a Git UI.

Install Deps

- Docker 17.05+
- Docker-Compose
- git
- Need a web browser on this or a different machine

Download and Configure Gitea

- Create a new folder called “gitea” and inside of it a file called “docker-compose.yml”
- Give docker-compose.yml the following content. Change the USER_UID and USER_GID to your own user id and group id (`id`). Also change SSH_DOMAIN to be the FQDN of the server.

```
version: "2"
services:
  server:
    image: gitea/gitea:latest
    environment:
      - USER_UID=1001
      - USER_GID=1001
      - SSH_PORT=222
      - SSH_DOMAIN=localhost
    restart: always
    volumes:
      - ./data:/data
    ports:
      - "3000:3000"
      - "222:22"
```

- Create a folder called “data” inside of “gitea”. Here the data for your repositories and Gitea database files will live.
- Build and run the service
    - `> docker-compose up -d`
- The server is now live on port 3000 of the docker host. Visit it in your web browser and click “register”.
- Check over the information. It should be all correct though. Once you are done, click “install gitea”. Don’t worry about setting up an admin user yet.

    - If the reload hangs, wait a minute or two and the manually navigate back to the main page.
- Once the install is complete, go back to the home page, and click “register” again. This first user will be the site admin.

Create a Shared Blue Team Repository

This will create one shared repository and one shared Gitea account. Another option is to create a user for each blue team member. The second option is probably better practice, but may take more time. You can skip the SSH steps if you want to use HTTP instead.

- Create an SSH key that will be used to access the repository. You should set a password for this.
    - > ssh-keygen -f blue_git
- In Gitea, navigate to the SSH keys page
    - Right Menu > Settings > SSH / GPG Keys
- Click “Add Key” under Manage SSH Keys
- Paste the contents of the PUBLIC key (blue_git.pub) in contents and save the key.
- Click the Plus button in the top bar
- Create a new private repository. Click the “Initialize with README”.
- Share the private key with your fellow blue team members. Here are some suggestions
    - Create a new directory
    - Put the SSH config file (see below) and the private key into the folder
    - > tar -cvf git_config.tar <directory>
    - > python3 -m http.server
    - > python -m SimpleHTTPServer
    - > nc -l 4000 < blue_git
    - > npm i -g serve && serve .

I want to use the repo

Get started

HTTP

- > git clone http://<whatever>/<user>/<repo>

SSH

- First, obtain the private ssh key used with the blue team central repository.
- There are two ways to use git with a custom SSH key. Pick one (1st is faster).
    - You can set the environment variable “GIT_SSH_COMMAND”. Run the following: `export GIT_SSH_COMMAND=”ssh -i <key file>” `, and maybe add this to your .bashrc file to make it permanent. You need git 2.10+ for this.

    - You can setup an ssh config file. Use the name "gitserv" instead of the normal server domain when cloning. Create ~/.ssh/config with the following

```
Host gitserv
    Hostname <gitea server location>
    Port 222
    User git
    IdentityFile ~/.ssh/blue_git
    IdentitiesOnly yes
```

- Clone the repository
    - > git clone git@<gitea host>:<222>/<gitea user name>/repo_name>
    - > cd <repo_name>

I want to make a change

- Stage your changes to that they can be committed
    - > git add *
- Commit the changes
    - > git commit -m "<your commit message>"
- Merge with any other changes
    - > git pull
- Push to the remote repo
    - > git push origin master
- When in doubt, check your status
    - > git status

Backup the data

You can run the following command to create a compressed archive of repository data:

```
> tar cvzf gitea.tar.gz gitea
```

Once you have this file, be sure to store it on a seperate server.

Oh No! The red team destroyed our git server

Through the magic of git you will be saved.
If you have the backup:

1. Restore the backup archive with the command `> tar -xvf <backup_file.tar.gz>`
2. Make sure docker-compose is installed
3. Run `docker-compose up`
4. Wait for the docker images to build and the repository to finish installing

If you don't have the backup, follow the Gitea installation instructions at the top of the page. Once you get to the part about creating a repo, instead opt to import someone else's copy of the existing repo.

Each person who has cloned the repo must create a new “remote”. Remotes are where git tracks remote copies of the repository that you push and pull changes from.

- Add: `> git remote add <remote name> <remote user>`
- List Remotes: `> git remote -v`
- Fetching: `> git fetch <remote name>`
- Pushing: `> git push <remote name> master`

Other Notes

- If you need CI, use Drone (https://docs.drone.io/installation/gitea/single-machine/)
- Gitea has great documentation. See https://docs.gitea.io/en-us/install-with-docker/
- If you need TLS, you can set this up with Gitea natively (https://docs.gitea.io/en-us/https-setup/). You probably want to add a reverse proxy (nginx) to the docker-compose file instead.
- Gitea running in docker exports certain logs to the /data directory. You may wish to monitor these with SIEM. They’re at /data/gitea.log.
- Gitea supports authentication via AD/LDAP or FreeIPA

Inject Response Template

You can use the following if you need to write an inject response to a “create version control” inject that asks for a background on Git.

Summary
The security team has successfully installed a version control service for use in our software environment. A version control service can allow users to collaborate and improve a software project, or any other work based around text files on a computer while lending the ability to return to an old revision if need be. Version control essentially offers a timeline of the work done on a service. This is often very useful for system administrators, because a service may not respond well to a change due to unforeseen circumstances, and so the ability to return to an earlier version of the system is often quite helpful during maintenance periods or routine work. It also allows for multiple team members to easily collaborate on the same project at once. Git is the standard version control software for large teams working in modern software environments, as it offers a good balance of performance and usability.

From a security perspective, version control is important as it allows a previously functional state to be restored in the case of malicious configuration changes. By comparing current and previous versions, potentially malicious changes can be discovered that may not have been noticed otherwise. Furthermore, non-malicious changes can result in accidental security risks. Access to previous versions enables a fast response to these configuration changes, thereby minimizing any impact to network security and end-user productivity.

Technical Details
Our version control system utilizes Gitea, an open source web interface for Git. It is running on <SERVER>, one of our <DISTRO> Linux systems. Gitea is a self-hosted alternative to GitHub. See Figure 1 in the appendix for screenshots of the web interface to Gitea. Gitea was chosen because of the simplicity of its configuration and speed and responsiveness of its interface.

The Gitea installation is running in Docker, which increases the security of our server by isolating critical application processes from one another, and also allows for more rapid deployment. We are using docker-compose to create a reproducible configuration for the service. The configuration of our Gitea server is stored in Gitea itself, as well as in clones of the repository on other machines.

Usage for Backups
The team has a number of important configuration files for our SIEM/centralized logging services. <DESCRIBE YOUR CONFIG FILES>

<HERE IS HOW YOU CAN USE GITEA>

Appendix
<INCLUDE SCREENSHOTS AND INSTRUCTIONS>