Abstract—Secure coding is a software development practice of developing software in a way that prevents accidental introduction of security vulnerabilities. In the current environment, there exists some tools like DVWA, WebGoat etc which in one way or another provides test case for certain number of well known vulnerabilities to the user to teach them about the vulnerabilities. The intended audiences of most of those tools are security researchers and not application programmers. This is the primary reason behind the development of our secure coding framework. This paper mainly focuses the deficiencies that the current conventional approach entails, how our secure coding framework rectifies those deficiencies, and finally justifies the major design decisions behind the construction of the framework.

Keywords—vulnerabilities; secure coding;

I. INTRODUCTION

In today’s environment, the amount of vulnerabilities being exposed and exploited due to inherent bugs within the code is rising by unprecedented amounts. By looking at the roots of this massive problem, we can understand that the main cause of this problem is the inability of the developers to avoid, identify and fix vulnerabilities, even the most basic ones. This can be rectified by teaching developers secure coding, by which we can enable the developers to not only avoid commonly occurring vulnerabilities in the codebase that they are writing, but also enable application developers to identify the bugs that are already in the codebase written by other developers, and then finally provide suitable fix to the vulnerability at hand [1]. This would in turn enable the software being developed by the group of people who are properly trained in secure coding to be more resilient to vulnerabilities than the software being developed by any other group. This in turn would enable the whole software industry to be more secure and non-vulnerable to malicious code in general [2]. The problem with learning/teaching secure coding is that the tools that are currently being used is not compatible with the overall idea of secure coding and is therefore severely inefficient. This problem arises because of the fact that most of the tools being currently used were not actually developed with the intention of being used as a secure coding teaching tool in mind and the target audience of the tools are also different from the audience of casual developers without much application security related knowledge [2]. These tools were merely adapted to the task at hand not because they were the most efficient or effective tools, but because they were the most compatible of all the tools available. The main problem that is tackled in this work is the overall lack of tools to teach and learn secure coding efficiently in the current industry along with the lack of general understanding of secure coding in the current software environment. To tackle such a huge problem, first of all an in depth analysis of all the tools and framework available in the current industry is needed, to find out whether the tools available and currently being used for teaching and learning secure coding is efficient and optimal for the task at hand. Although there have been several attempts at teaching secure coding at an industry wide level, most of such efforts have withered away after a short amount of time without any substantial effort. This confirms the inefficiency of the tools used for vulnerability analysis.

The aim of this paper is to propose a framework for teaching/learning secure coding, which is build from the scratch by focusing on the ideas of user friendliness and efficiency. A prototype of such a framework, using design goals and requirements created from the deficiencies identified by analyzing the current tools being widely used in the industry, was created and deployed. The aim of this paper is to identify the challenges and roadblocks that prevent the creation of such a framework and to identify the optimal solutions for the same without losing sight of the original goals/requirements.
II. RELATED WORKS

Secure development has been identified as one of the central problems to solve in the area of application security due to its inception itself and so this work is not about threading the same grounds. Since early times, a large amount of research has been done in the area of application security and therefore, the problem is not the availability of information regarding vulnerabilities and what causes them. The problem is the expansiveness of the knowledge that is currently available. Due to the massive amount of data available, the software developers are often overwhelmed when try to find an initial grasp of the information necessary for them to code in a secure manner. This paper aims to present a relevant starting point for any developers who intend to learn and understand the concept of secure coding.

A. Secure Coding Documentations

Some of the most relevant work in this area includes:

1) OWASP, the Open Web Application Security Project is a worldwide nonprofit organization focused on improving the security of software [3]. It is an online community which creates articles which are freely available, documentation, methodologies, tools, and technologies in the field of web application security. Each year, they organize several application security conferences in different locations throughout the world.

2) eLearnSecurity builders of Hack.me is a free Web Application Security virtual lab where everyone can build, share and run vulnerable web applications [4][5].

3) Mozilla secure coding guidelines, is a concise and consistent approach to secure application development of Mozilla web applications and web services [6]. It is also applicable to general secure software development.

4) Mozilla secure coding checklist [7]. is a checklist to cross check the developed applications against a standard checklist of recommended techniques for secure development.

5) MS SDLC, Microsoft Security Development Lifecycle is based on Microsoft’s focus on secure development practices which they have incorporated in their Software Development Life Cycle (SDLC) and provide resources for how others can do the same [8].

6) SANS institute and their resources [9], SANS have training programs and online resources to spread awareness and knowledge about computer security. The drawback for this is that the SANS courses are extremely costly ($ 5000 per head)

B. Intentionally Vulnerable Applications

There are also some set of applications which helps the user to learn security offensively, by exploiting an intentionally vulnerable application. Some of the more widely used ones are discussed below:

1) DVWA, Damn Vulnerable Web Application, an application which contains intentional vulnerabilities whose main goal are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications[10].

2) WebGoat, a deliberately insecure web application maintained by OWASP which is designed to have multiple lessons, each of which teaches a concept or a web vulnerability [11].

There are more tools like DVWA and WebGoat which does an excellent job teaching security professionals on how to attack a target with more precision. The problem is that these kinds of tools are not suitable for teaching its users why such vulnerabilities exist and what can be done to fix them entirely. Applications like DVWA, WebGoat just provides the web pages without any explanation of what the vulnerability is and how it works. The web pages, which are vulnerable does not provide any method to work out how to fix the vulnerability. All that can be done with the web pages is learning how to attack the webpage and subsequently to do successful attack. The framework itself doesn’t lend itself to much extension or addition of new challenges, and so the most recent and relevant attacks are not present. One other major defect of the application is that it only deals with web based vulnerabilities.

III. PROPOSED SYSTEM

Looking at all the defects of the applications listed in related works, we can derive a set of requirements for a good secure coding learning environment.

- There must be accurate information about the vulnerabilities themselves.
- There must be a way for the user to experiment and try out the code to arrive at a fix for the vulnerability.
- There must be some kind of informative feedback about the execution of code and the vulnerabilities present.
- The framework must be open and extensible, so that modern vulnerabilities can be added, so that the framework remains relevant in the future also.
- The framework must not be tied down to a certain kind of attacks.
- There must be a way to verify whether the user implemented fix works correctly or not.
- The possibility of false positives should be reduced as much as possible

Looking at all the requirements and researching whether there exists any framework which satisfy all these requirements, we arrived at the conclusion that rather than taking any other framework/application and adapting it to our need by extending it, we need to create a new framework from scratch. The reason for building a new framework from scratch is as follows...
1) It would make it easier to provide accurate exception handling and logging across the whole framework.
2) It would make for a much easier to install framework with much less dependencies.
3) It would make it more easier to make the whole framework more extensible and open.

A. Main Goal

The best way to learn anything is by practicing. Similarly, the only way we can teach developers about coding securely is by giving them different pieces of vulnerable code in which the developers should understand:
- What is the vulnerability in the given code.
- How the vulnerability in the code can be exploited.
- What makes the vulnerability so critical that it should be fixed.
- And finally how to fix this vulnerability so that attackers will not be able to exploit it.

B. Our system framework

We have created a web application which works in the exact same way as mentioned above. The framework can be deployed on any web server and it has a set of questions which can be solved by a developer. The challenges will be listed one by one which also has varying difficulty. Some of the challenges are really in which finding the vulnerability is easier while for some it requires some good amount of time.

The challenges are divided on the basis of criteria:
- a) All challenges
- b) SQL injection Challenges [13]
- c) Cross Site Scripting (XSS) Challenges [14]

and also based on the difficult of the challenges:
- a) Easy
- b) Medium
- c) hard

Developers can sort the challenges based on these so that they can attempt challenges on the section they wanted (not all developers wanted to learn SQL) along with the difficulty level they wanted. Once the user click on “Solve Challenge”, they are taken into a new page where the vulnerable code is shown to the user and he should read and understand the code.

Once the user understood what the code is doing, user is suppose to learn how the code can be exploited and how to prevent it. The existing vulnerable code should be modified so that the vulnerability is fixed and the final code should be submitted back to the framework. Figure 1 depicts the overall working of the framework.

C. Example challenge (in PHP)

A sample challenge structure and how it is expected to be solved.

```php
function sanitize($user_input) {
    $x = htmlspecialchars(trim($user_input), ENT_QUOTES);
    return "<span title="'.$x.'">User input added as span title $x</span>";
}
```

The above code is a sanitizing filter used by an application to HTML encode the user input so that it can prevent Cross Site Scripting attacks (XSS). In short, what the function does is that:
1) Takes an input from a user.
2) The user input is then passed through a PHP function `htmlspecialchars()` which HTML encodes special characters like `textless`, `textgreater`, and `"`.
3) Once the user input is HTML encoded, it is added to a `textless` span `textgreater` tag in HTML and is returned.

4) This resembles a real world XSS filters which blindly relies on standard functions without knowing how to use it well on different contexts.

The above code looks secure at the first glance but this can be easily bypassed because by default htmlspecialchars() will not encode single quotes and if the developer uses single quotes to enclose strings (like the above case), this filter can be bypassed (input: `onload="alert(1)`).

So the proper fixed code should be:

```php
function sanitize($user_input) {
    $x = htmlspecialchars(trim($user_input), ENT_QUOTES);
    return "<span title=""$x"">User input added as span title $x</span>";
}
```

Now the code is complete because htmlspecialchars() along with ENT_QUOTES as argument will encode everything and an XSS in the HTML context is not possible. So once the user submits the fixed code, the unit tests will contain several XSS payloads which it uses against the returned code to test how rigid the filter is.

D. Sandboxiing

The final submitted code has to be tested to see if the fix is proper or if it still contains any vulnerability.

1) By using Regular Expressions, check back the returned code to see if the final code is proper.
2) By actually executing the returned code to see if it works properly without any vulnerability.

First method is easy but not full proof because if using regex, there are several ways in which it can be bypassed and also we cannot guarantee that the submitted back solution is actually proper. Users can even modify the existing code in a way it breaks the functionality and then submit back the code. Such methods should be blocked and hence only way is to execute the user submitted code and analyze its result. Directly running a user supplied code is extremely dangerous as it can contains system level code execution which in turn can be misused. Hence we need a sandbox environment for the

![Fig.1. System flow diagram](image-url)
proper execution of the code so that host machine remains unaffected. To enable a sandboxing environment in the most secure and reliable way so as to not affect the host machine, virtualization should be done. This way each challenge can be sandboxed and the user can be given full control over the modification and execution of the same. Since the nature of the challenge itself can be wide ranging like binary based, web based etc. Running the challenges within some type of virtualization is better than setting up a detailed secure environment within the host machine itself. This can be attributed to the following reasons:

1) Running the challenges by means of virtualization, inherently provides several level of security by means of isolation. Achieving the same level of security by means of a custom secure environment setup within the host machine requires a huge amount of time and skill. Also the secure environment also has a higher chance to be vulnerable since it's not extensively researched or written by security professionals.

2) Multiple types of challenges from web based to binary based will require the secure environment to adapt to the type of application, or there should be different secure environment for handling each type of application. Since the types of challenges are not limited in scope, virtualization is the best way to go, since virtualization offers the same amount of security for all applications regardless of their type.

Hypervisor is application/software that creates/run virtual machines within the host computer. Hypervisors are divided into mainly 2 types [15].

1) Native or bare metal hypervisors: These are hypervisors that directly run on the hardware bypassing the host machine. The operating systems runs on the virtual machines that are created by the hypervisors and they appear as a process within the host OS. Examples are Vmware ESX, Microsoft Hyper-V etc. Since they mostly bypass the host OS and run directly on the hardware, the scripting, programming and inclusion within a framework.

2) Hosted hypervisors: These run on top of OS just as conventional programs and virtual machines run on top of this hypervisor process. The hypervisor itself runs as a single process within the OS and so the OS has more control over the hypervisor. This makes it easier for scripting using the hypervisor. The disadvantage is that, since the hypervisor runs on top of another OS, it doesn't have direct access to the hardware and since the hypervisor itself provides an hardware abstraction for the virtual machine, if the guest OS has to access the hardware, it has to go through two layers of abstraction, which severely hampers the performance. Also deploying each virtual machine for each challenge and user will take up large amounts of resources, since the virtual machines themselves provide a fully functional OS for each challenge. Examples include Vmware Workstation [16], VirtualBox, QEMU etc.

Our challenges themselves doesn’t require a full OS to work, and therein lies the overhead. All the virtualization options discussed until now gives us a full OS virtualization and this is not required for our challenges. We need a technology that provides the security of a full OS virtualization while also only providing virtualization of necessary components. While researching for such a technology, we arrived at Docker [18]. Docker allows applications to be run within ‘containers’ which are isolated from the host machine. While containers provide OS level virtualization, it is not full hardware virtualization. Docker uses linux kernel features such as cgroups and kernel namespaces to provide resource isolation for the containers, this means that instead of going through multiple levels of abstraction, the containers can directly access the hardware, just like normal processes. This in turn translates into greater process execution speed for processes within the containers. The containers themselves also provide network isolation, which in turn is helpful while hosting web based challenges [19].

Also, the docker containers themselves are very lightweight. Only the bare minimum requirement for running an application is present within the containers [20]. The whole ideology behind docker containers themselves is that they should be used for running a single process and a single process only. The performance of docker containers when compared to virtual machines have been extensively studied and in most cases, where we are concerned, docker containers outperforms virtual machines. All the above said factors lead to our choice of using docker containers and sandboxes for our challenges.

After deciding the inclusion of docker as our challenge sandbox environment, we have to look at how the docker service actually works so that we can look at how we can integrate it into our framework and make the whole process user friendly [21].

The docker service use docker images as base when creating the docker containers. Docker images and docker containers can be said to be analogous to image iso's and virtual machines. Docker images themselves are created by using docker files. A docker file contains a base image from which the new image is to be generated along with all the additional alterations that are to be made within the base image to create the new image. Upon giving the appropriate command, the docker service downloads the base image from the central docker repository, executes all the instructions within the docker file and then creates the new image. The new image and base image resides within the host machine's local docker repository.

The central docker repository is a public repository that is available for all and contains all the popular docker images, while the user’s local docker repository is private to the user and contains only those images that the user downloads or creates locally. And when the appropriate command is given, the docker service generates a container based on the specified image in the user’s local repository. While creating the docker image we can also use simple port forwarding to access the services of the docker container from the host machine since docker containers are isolated on the basis of network.

By making use of the dockerfiles and the port forwarding, we can easily script the docker service and include the sandbox within our framework. The docker service also provides
bindings for all major programming languages including python. This means that we can directly use the official python library for interfacing with the docker service.

The challenges were implemented firstly by spawning the docker container and then using a web interface provide the user with a means to edit the challenge, execute the user modified code within the sandbox and the finally provide some kind of feedback to the user about the code that was executed. The way we implemented the feedback system was by means of unit test output. The way we tested whether the vulnerability present in the code was fixed or not was by means of testing it using predefined unit tests, which should be written by the challenge author while the creation of the challenge. By providing the results of these unit tests back to the user, the user can identify why the code still has the vulnerability present and can more quickly arrive at a accurate fix for the vulnerability present.

The first thing to do is to enable the framework to create a local service that will abstract away all the complexity of handling docker images and containers directly. The challenge itself should only compose of a dockerfile along with the files necessary for the challenge. The framework should handle all the rest from creating the docker image to creating and killing containers as necessary.

One of the main things that were of concern was the abuse of the system by spawning a large number of containers and depleting the resources of the host machine. The possible solution that we arrived at was the restriction of the amount of resources used by each individual docker container as well as making the container time bound, i.e. if no activity is found within a container for a specified amount of time, the service will kill the container owing to inactivity. By using this solution, we can prevent the individual docker container from existing for an infinite amount of time along with hogging too many resources.

Owing to the user accessibility side, one of the requirements for the framework was that it should be self contained, and all the dependencies needed by the framework should be installed by the framework itself. Another requirement for the framework was that it should have full exception and logging coverage. Finally, it was after looking at all the requirements along with looking at all the previous research that we designed our own framework.

In the current state, as soon as the main file of our framework is executed it checks whether all the dependencies required for the correct working of the system exists in the host system or not, including the docker service. If the dependencies are not found, the framework starts the installation of the missing dependencies and asks the users permission to proceed. After all the missing dependencies are installed, the framework then proceeds to creates and starts a local service that handles all the docker related matters and then also proceeds to start the django based frontend for the user.

When the user actually clicks the link provided for spawning a challenge in the webpage, the framework looks at the challenge-id and then sends the command to spawn a container using the base image for the respective challenge id.

The local service looks at the challenge id, identifies the image to be used for the challenge, creates a container based on the challenge. Then by means of port forwarding the port number at which the challenge resides in the container is forwarded to a random unused port within the host system. The the absolute URL of the challenge in the host system is returned to the framework, which takes the URL and open a new tab for the user with the URL. This new tab will contain an interface by which user can edit his source code, execute it and also see the outcome of the unit tests. When the user finally closes the tab, or the time limit for inactivity exceeds, the framework kills the container.

E. Adding challenges to framework

Right now, the docker image is built over standard ubuntu base image and has basic software installed while running installation script. For normal or ordinary challenges which doesn’t require much configuration changes can work well with the current format but when it comes to more advanced challenges, the current version is not completely enough. The major drawbacks are that the as of now new challenges:

1) Cannot install new softwares or packages
2) Cannot change any configurations already present (like mysql password)
3) Cannot install libraries of any kind or choose custom operating systems of their choice (everything has to be depend on Ubuntu machine)
4) To implement Binary or system level challenges like buffer overflows need much more flexibility than the current extend.

The possible solutions for these challenges are:

1) Allow challenge authors to submit their own custom docker images. But this is not an optimal solution even though at first, this looks like a great option as challenge authors have complete control over the docker, and can create custom configurations etc. But this is not a viable option because of the following:

   a) Docker images will be big in size, say approximately 250 MB per standard ubuntu base image, storing such big size images need more space.
   b) While installing, the client needs to download a lot of files and images (some of them will be redundant as both images would be derived from same base image)
   c) Let’s say there are 100 custom challenges with 100 different images. In order to install this locally, the client has to download an approx 250 * 100 = 25,000 MB (~25 GB) which is totally a bad idea.

2) Allow challenge authors to modify the existing Dockerfile to add their own rules. This looks like a viable option at first but in the long run, it’s not because:

   a) Changing the existing configuration of the docker image by modifying the Dockerfile can break several other challenges.
   b) For a proper integration without breaking other challenges, the author must spend huge amount of
time learning other challenges and integrate in such a way that everything works properly. 

3) Allow challenge authors to submit custom Dockerfile: This is the best possible idea due to several reasons:
   a) Challenge authors have full rights to configure their docker images. All the configuration changes and build instructions are written in a custom Dockerfile.
   b) Users don’t have to download huge files, a standard docker image can be used to build new custom images based on the Dockerfile.
   c) Since custom challenges use custom Dockerfile to build a new image, the existing functionality is preserved and chances of failure are low.

Other advanced challenges like Buffer overflows could be easily integrated since authors have full freedom to customize the installation.

The working of the whole process is as follows:
1) Challenge authors submit new Dockerfile and place it inside the “Custom challenges” directory.
2) When students want to try out the challenge, he/she clicks on “Try challenge” and a new docker image build is run.
3) Using the newly created image, the challenge will be hosted and URL to access the challenge is given to the user.

IV. TESTING AND ANALYSIS

During the testing phase of the project, we have done several experiments to study the feasibility of the project. The primary tests was to see how well the docker performed when compared to other virtualization solutions before using it as a permanent sandbox solution.

A. Memory Analysis

Since we are dealing with a framework where so many users can try out the challenge simultaneously, the sandbox solution should be robust, easily controllable and should not take much resource even if the number of instances is high. So we took a normal workstation and see how many instances of a VM and Docker can be run simultaneously with bare minimum operating system running inside it. It also includes the very essential applications to run a web application which includes Apache2, PHP and MySQL [24].

Results:
The VM’s couldn’t launch more than 4 instances since they use too much memory, the system disk and memory becomes a bottleneck. The number 4 is quite low and it cannot be used for the framework whatsoever. The Docker test result actually surprised us as the number of parallelly running Docker instances on a normal workstation went up to 200 instances without having much bottleneck for either system memory or disk which clearly shows that Docker is the best sandbox solution which can be used for the framework.

B. Disk Space Analysis

When looking into the performance of the docker based solution, we had to look into another problem, that affects the general scalability of the framework, the hard disk space used by the individual images. If each individual image exists independently, then each image would consume a considerable amount of disk space and as more and more images are added in accordance to the addition of more challenges, the disk space consumption of the total framework would increase to enormous sizes. The easiest solution to this problem was to make the disk images based off a single consolidated base image. One of the unique features of docker virtualization solution is its AUFS file system. Any image is stored within the host machines computer as several independent layers which can be shared across several images. Therefore, if all the challenge images are based upon the same base image, the same base image layer can shared across several images. Then the individual challenges are only made up of the layers that are built upon the base image.

Result:
We created several challenge images based upon the same base image and checked the total image size of the images and then checked the total size occupied by the individual layers. This trade off will represent the gain from making the images off the same base image. As seen in the above case, by using a single base image of about 300MB size as the common size, we have saved 300MB when using two varied images based on the same image. This savings will scale along with the number of challenges deployed and thereby saving valuable disk space [25].

V. CONCLUSION

In today’s online cloud based workflow, there is a severe lack of secure coding practices. This is evident in the ever increasing number of vulnerabilities that are being found out
in several modern popular softwares, most of which are caused due to simple bugs in the code, which could have been avoided very easily. The main problem is that the tools that are used or that can be used for teaching/learning secure coding are not intended for the same and are also not intended for the audience of application/software developers. After extensively researching the deficiencies of current tools as well as the requirements that a secure coding framework should have, we came up with a set of basic design goals for our framework and after this we finally created our secure coding framework from scratch. After the addition of interesting and ‘efficient’ challenges to our framework, our tool can be used for learning and or teaching secure coding at both educational and enterprise level. The remaining work that comes outside the scope of this paper includes the addition of efficient and interactive challenges to the secure coding prototype.

REFERENCES
