Justin Casteel

SDEV 460 – Homework 2

July 13, 2018

**Phase 1 - Before Development Begins**

Before development even begins it is necessary that a full SDLC be defined where security may potentially be a threat. This SDLC structure is a framework for the entire team that consists of a detailed plan on how we will develop, maintain and replace our software. Three things that are a must in this stage of the development process is to ensure there are policies, standards and documentation. Without these, your team will have no general direction and no guidelines and policies to follow when working on the project. For example, if the application is being developed in Java, it is essential that there is a Java secure coding standard. Although these policies and standards may not cover every possible situation the team may face, it is important to document the common problems that the team may encounter so that fewer decisions will need to be made during development.

**Phase 2 - During Definition and Design**

The first phase of this phase is to review security requirements, which defines how our application with function from a security perspective. This is one of the most important steps in this design framework as it tests to see if there are gaps in the requirements definitions by testing the security requirements. Three security mechanisms we may run into on this stage may be User Managements, Session Management and Data Confidentiality. We create documented design and architecture which includes models and textual documents. We will need to test these documents as well to ensure that they conform to the proper design and architecture to enforce the proper level of security defined in our requirements. This is the time to review and identify and security flaws and one of the best place to make changes for the rest of the SDLC. We will also create and review UML models which describe how our application will function. These UML models are used to confirm with system designers and to give an exact understanding of how the application will work. Phase 2.4 will allow the team to create threat modeling exercises where they will develop realistic threat scenarios. After analyzing the design and architecture and mitigating any threats we will need to have our project accepted by a potential business to begin development. If a threat is unable to be mitigated, we will need to revisit the design and architecture and modify the design to mitigate these threats.

**Phase 3 – During Development**

During development we take our design from Phase 1 and 2 and begin implementing this design to make it a reality. Sometimes during the development process design decisions may be made during code development. These types of changes are typically necessary simply because these ideas were to details to describe in the design phase or there were no issues where the policy and guidance was offered. This phase will allow the security team to perform a code walk through with the developers and system architects. This is a high-level walkthrough that allows developers to explain the logic and flow of the implemented code and allows for answers on why specific ideas were developed the way that they were. Phase 3.2 is used after having an excellent understand of how the code is structured so that the tester can examine it for security flaws. There are many checklists available that will allow the tester to check all necessary sections of the code to ensure these flaws are being mitigated by the program. Static code reviews are also used to minimize the amount of resources used by the team, such as time and rely less on the skill level of the reviewer of the code.

**Phase 4 – During Deployment**

The first 3 phases of the SDLC allowed us to test the requirements, analyze the design and performed all code review. In this phase we can do penetration testing after it has been deployed which provides a last check to ensure that the team has not missed anything. Three things that we will check in this stage are checking of how the infrastructure was deployed and secured. We also check the configuration and insure that there is not an exploitation that may arise upon install.

**Phase 5 – Maintenance and Operations**

In the maintenance and operations phase, we have details in place that shows how both the application and the infrastructure is managed. Application health checks should be in place to ensure that no new security risks have been introduced to the application or infrastructure. In this phase we also ensure change verification which means that every change that’s approved and tested in the QA environment is checked to ensure the level of security is still intact. Regression tests and operational management reviews are key to this phase and are of utmost importance when keeping our program secure and keeping security flaws at a minimum.

Password: umuc$d8v

Ip: 10.0.2.15

**Part 1 – Fingerprint Web Server (OTG-INFO-002)**

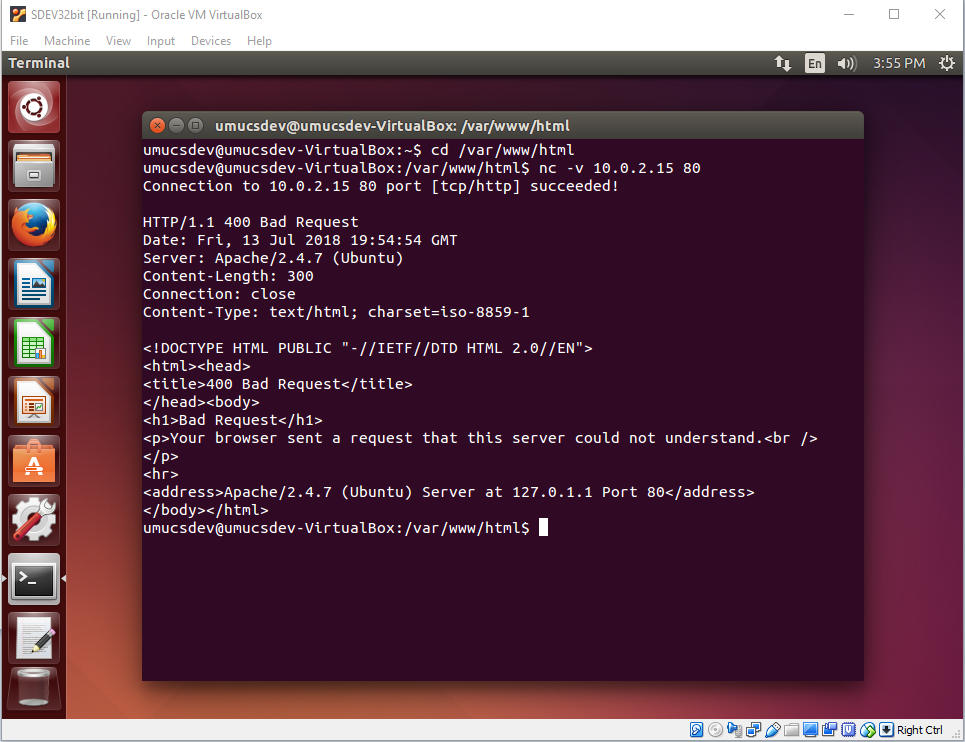
****

Figure 1 - Fingerprint

**2.4.7 Vulnerability List-** <https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-161847/year-2014/Apache-Http-Server-2.4.7.html>

The top-rated vulnerability which received a CVSS Score of 6.8 says the following:

“Race condition in the mod\_status module in the Apache HTTP Server before 2.4.10 allows remote attackers to cause a denial of service (heap-based buffer overflow), or possibly obtain sensitive credential information or execute arbitrary code, via a crafted request that triggers improper scoreboard handling within the status\_handler function in modules/generators/mod\_status.c and the lua\_ap\_scoreboard\_worker function in modules/lua/lua\_request.c.”

In layman’s terms, this vulnerability could allow an unauthorized remote attacker to execute code and gain access to information or cause a denial of service on the targeted system. The fix was not directly shown to the public but there are a few safeguards geared towards administrators to avoid this type of vulnerability in the future which are:

1. Ensure software is up-to-date
2. Only allow trusted users to have network access.
3. Monitor affected systems.

This vulnerability was fixed in Apache version 2.4.10.

**Part 2 – Review webpage comments and metadata for information leakage (OTG-INFO-005)**

Below are the apache directory files that I used for this part.

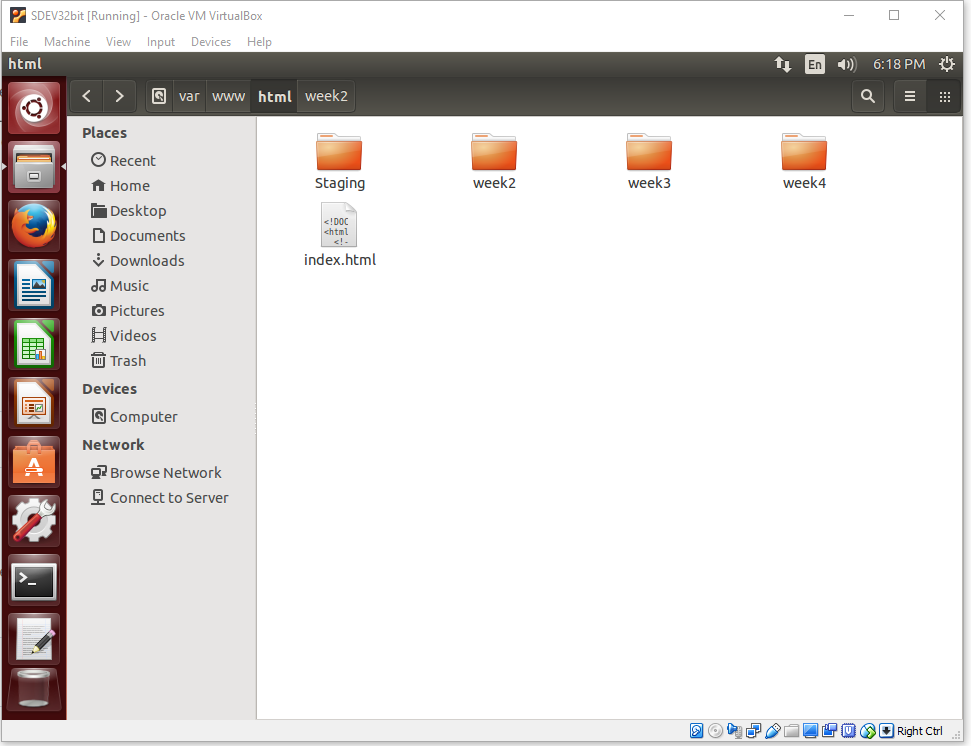


Figure 2 - web directory

File name: CNShome.php

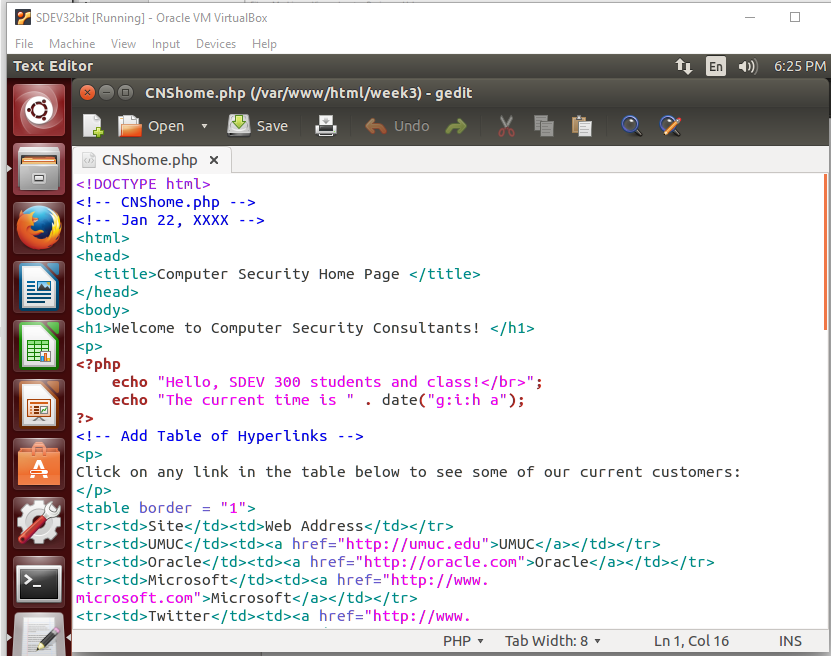


Figure 3 - CNShome.php

The file above violates HTML version information and does not display valid version numbers and Data Type Definition (DTD) URLs. To mitigate this potential threat, we could change the <!DOCTYPE html> line to something along the lines of:

<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML5//EN” <http://www.localhost.com/html5/strict.dtd>”>

The URI in each document type declaration allows user agents to download the appropriate DTD and any entity sets necessary. URIs refer to DTDs and entity sets in HTML4 which include:

Strict.dtd – default strict DTD  
loose.dtd – loose DTD  
frameset.dtd – DTD for frameset documents  
HTMLLlat1.ent – Latin-1 entities  
HTMLsymbol.ent – Symbol entities  
HTMLspecial.ent – Special entities

Another potential issue could be a meta tag such as:

<META name=”Author” content=”Andrew Muller”>

This Can potentially give an attacker the opportunity to profile a specific application to a user. This next example isn’t in the META tag but is in the comment at the top of the document, so I thought that this may still pose a potential risk because an attacker to profile this specific program to this person, Dr. Robertson.

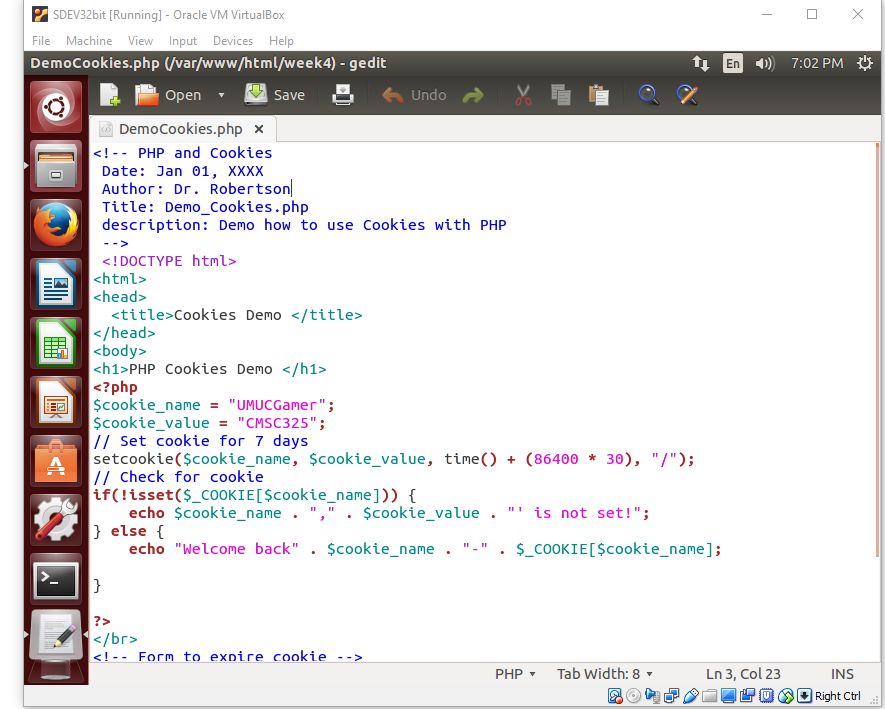


Figure 4 - DemoCookies.php

**Based upon online research, what are three or more categories of information that would be considered information leakage that is not acceptable?**

One category of information leakage that is not acceptable is not disabling or limiting detailed error handling. This means not displaying debut information to end users, stack traces, or path information. This falls under the protection category when dealing with information leakage. Another category is vulnerability and may not be acceptable when inducing an error displays too much information such as failed SQL statement or debugging information. The third category is verifying security and is typically acceptable but relying on automated approaches may be something to avoid in some cases since the program may find issues but will not be able to explain them all the time.

**Part 3 - Test HTTP Methods (OTG-CONFIG-006)**

I had a little bit of trouble using netcat for this section, so I had to use another command called “telnet” to see if certain methods were enabled. I believe that the OPTIONS command for netcat was disabled server side and therefore I was unable to use it to see information. I would always receive a 400 error which is below.

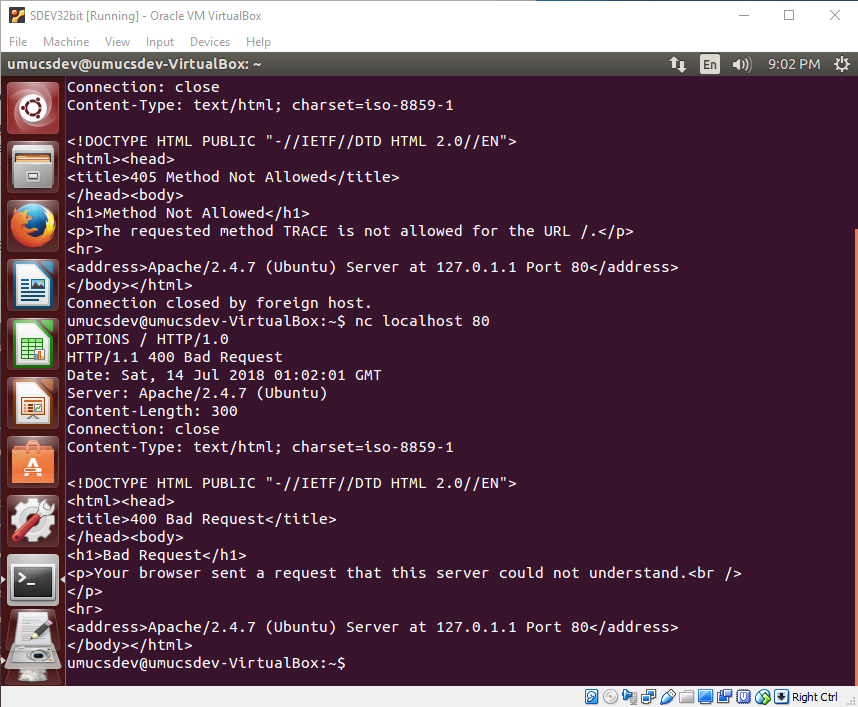


Figure 5 - netcat 400 error

Below shows the testing of the TRACE method to see if it’s enabled, and I find out it is not by clearly stating “Method Not Allowed”.

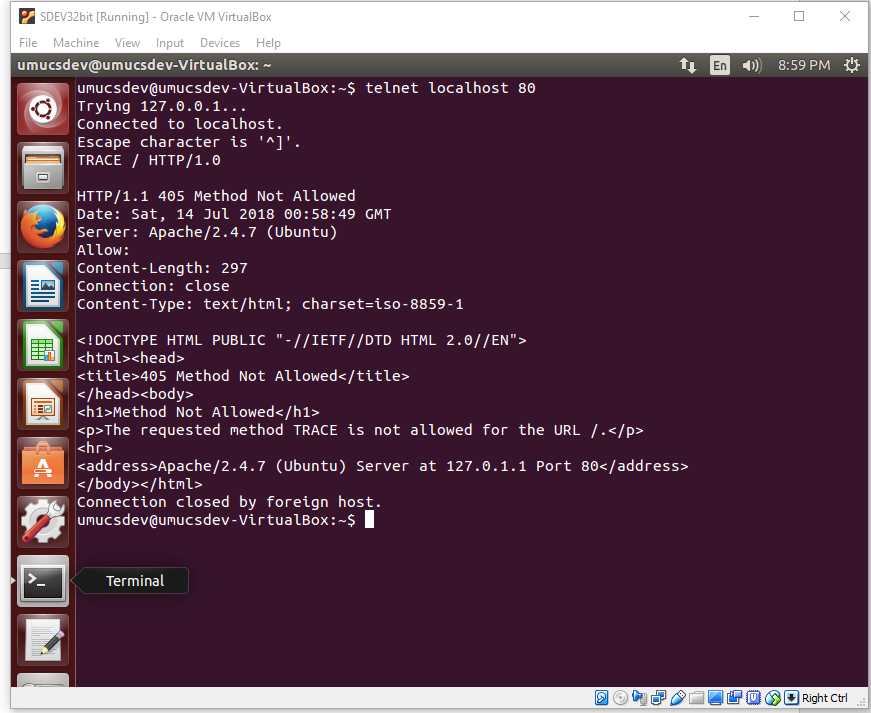


Figure 6 - TRACE Method

The following screenshot shows the testing of the HEAD method and shows that it is enabled.

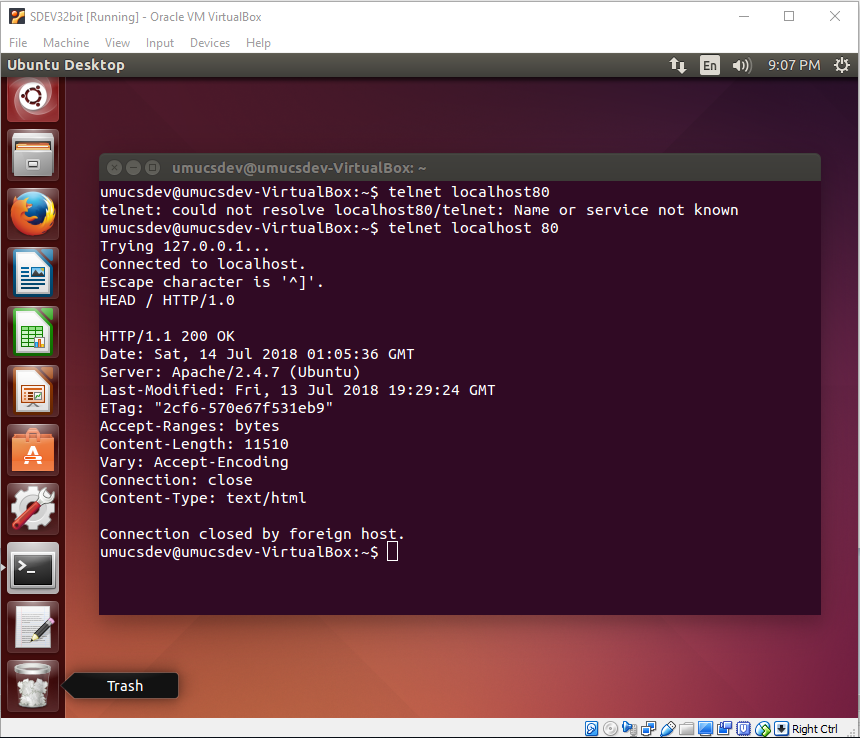


Figure 7 - HEAD Method

I then use the OPTIONS command that I later found out worked with telnet as well, which displays all available methods that are enabled, shown below. The server allows the GET, HEAD, POST, and OPTIONS commands as shown below.

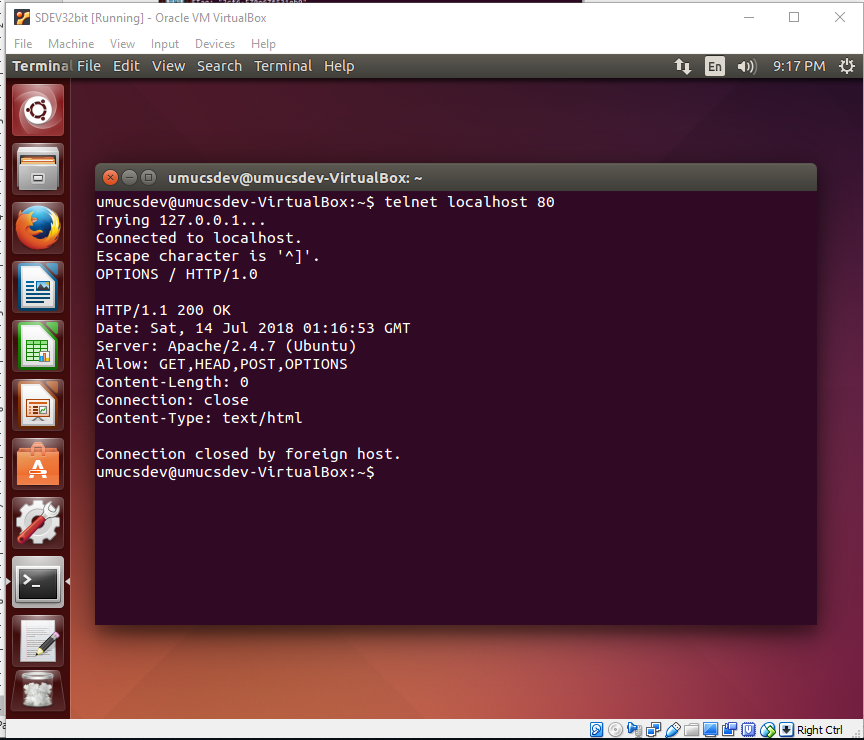


Figure 8 – OPTIONS

**Which methods (and why) have potentially pose a security risk for a web application. Describe how these pose a risk.**  
  
HEAD, GET, POST and CONNECT methods are typically safe when it comes to the HTTP Method. The request may contain malicious parameters but is separate from the method. These are the types of methods that are typically enabled on a web server.

PUT and DELETE methods were originally used for file management operations. This allows you to change or delete files from the servers file system and is a target for vulnerability and attacks. When dealing with many files on a server we need to ensure that these aren’t easily accessible by outsides and have a strictly limited permission set in place to avoid exposure of files where needed.

The OPTIONS method is a method used mainly for debugging and will also show us info like in the screenshot above. It lets us know which methods are enabled, the server currently running with the version as well. Although this does give useful information for the user it may grant an attacker a little bit of information to discover other holes in the system. This method typically does not need to be enabled and if it is enabled just becomes another potential threat on your system.

The TRACE method is a diagnostic method that returns an entire HTTP Request which includes the request body and headers. These include the cookies, authorization headers and more. This method typically doesn’t need to be enabled and only risks for exploitation, for example being able to retrieve HttpOnly cookies, authorization headers and more.

**References**

*The global structure of an HTML document*. (2018). *W3.org*. Retrieved 13 July 2018, from <https://www.w3.org/TR/html401/struct/global.html#h-7.2>  
  
Methods, H. (2018). *How to exploit HTTP Methods*. *Information Security Stack Exchange*. Retrieved 14 July 2018, from https://security.stackexchange.com/questions/21413/how-to-exploit-http-methods