<CodeGeek/>

As an atom is to a chemical element, so is the code to a software application. The section brings in a short overview of advanced coding vulnerabilities present in software applications. It also gives detailed explanations on various approaches that a penetration tester needs to understand.

----------*Unlocking the Secrets of Secure Coding*

Implementing a TCP SYN scanner in less than 10 lines of code

Mahesh U.Patil
Joint Director
Centre for Development of Advanced Computing (C-DAC), Hyderabad

Recently, I came across an interesting packet manipulation tool called scapy . It can forge or decode packets, send them on the wire, capture them, and match requests and replies. Moreover, its developed in Python, what a perfect combination a pentester would desire. In this short article, I demonstrate the strength of this tool supplemented with Python's list comprehension feature.

Background on TCP SYN scan

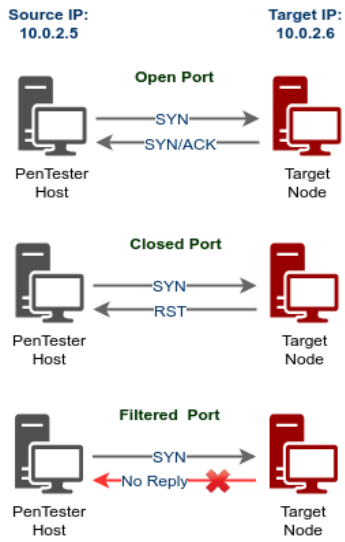
TCP SYN scan is one of the most popular port scanning techniques. It is popular because it is fast and is less likely to be blocked from firewalls. Another most important reason for its popularity is that it provides clear indications about the port state being open, closed or filtered.

It is built upon the concept of TCP three way handshake. As shown in the figure below, a TCP SYN packet is sent to the target IP with a destination port of interest. If the server replies with an SYN/ACK , the destination port is open and if it replies with a RST the destination port is closed. In case there is no reply received, the port is deemed to be filtered by the firewall.

Lets code

Goal:

To scan port 1 to 1023 on a target IP 10.0.2.6 from source IP 10.0.2.5



Code snippet

```
1.  from scapy.all import *
2.
3.  # Create TCP SYN packets
4.  p=IP(src="10.0.2.5", dst="10.0.2.6")/TCP(sport=5555,
    dport=(1,1023), flags='S')
5.
6.  # Send and Recieve the packets
7.  ans, unans = sr(p)
8.
9.  # Decode Answered packets
10. recvd_pkts = [recvd_pkt.getlayer('TCP') for sent_pkt,
    recvd_pkt in ans]
11. open_ports = [tcp_pkt.sport for tcp_pkt in recvd_pkts if
    tcp_pkt.flags == 'SA']
12. closed_ports = [tcp_pkt.sport for tcp_pkt in recvd_pkts if
    tcp_pkt.flags == 'RA']
13.
14. # Decode Un-answered packets
15. noans_pkts = [sent_pkt.getlayer('TCP') for sent_pkt in un-
    ans]
16. filtered_ports = [tcp_pkt.dport for tcp_pkt in noans_pkts]
17.
18. print("-------------------------------------------------")
19. print("[+] Open Ports")
20. print(open_ports)
21.
22. print("-------------------------------------------------")
23. print("[-] Filtered Ports")
24. print(filtered_ports)
```

Snapshot of output from our code

```
-------------------------------------------------
[+] Open Ports
[21, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514]
-------------------------------------------------
[-] Filtered Ports
[22]
```

Snapshot of output from nmap tool

```
root@kali:~# nmap -sS 10.0.2.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 18:23 EST
Nmap scan report for 10.0.2.6
Host is up (0.00031s latency).
Not shown: 977 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   filtered ssh
23/tcp   open     telnet
25/tcp   open     smtp
53/tcp   open     domain
80/tcp   open     http
111/tcp  open     rpcbind
139/tcp  open     netbios-ssn
445/tcp  open     microsoft-ds
512/tcp  open     exec
513/tcp  open     login
514/tcp  open     shell
```

Demystifying the code

Line Number	Explanation
4	Create an IP packet with source and destination IP filled in. Create a TCP packet with source and targeted destination ports. The flag field for TCP is set to S specifying that we are interested to send a TCP SYN. dport=(1,1023) means create a bunch of TCP packets with destination ports ranging from 1 to 1023 . p = IP()/TCP() # this syntax specifies to create a raw packet such that TCP packet is embedded in IP payload
7	This code sends the packets crafted in line 4. The received packets are in two categories i.e answered or unanswered. Answered packet is the one for which we have received reply for corresponding sent packet. Unanswered packet are the one for which we did not receive any reply from the destination.
10	In this line of code we harness the Python's list comprehension feature to collect all the received packets at layer TCP in a list called recvd_pkts
11	We filter out TCP packet where we received response with TCP flags as SYN/ACK. These are our packet with open ports.
12	We filter out TCP packet where we received response with TCP flags as RST/ACK. These are our packets with closed ports.
15	Here we handle the packets for which we have not received responses. These are precisely the packets that were filtered by firewall.
16	We filter out ports from unanswered TCP packets and these formulate our list of filtered ports

Coding Horrors

Shince Thomas
Chief Technology Officer (CTO), Digital Trust Technologies Private Limited

Importance of secure coding for improving application security thereby preventing cyber exploits is a well discussed topic today. Buffer overflow, session management, cookies and script related vulnerabilities are well known and today's secure coding practices put maximum focus to mitigate them. However, these strategies lack depth, largely putting thrust on server side security and often overlook vulnerabilities that nullify the security controls placed inside the code. This is a short overview of advanced coding vulnerabilities that compromise the entire gamut of security controls.

Client-server architecture was prominent in the 90s and was overtaken by 3-tier web based architecture in the 2000s. With the advent of smart phones, thick clients like mobile apps have become the choice of today for better User Experience (UX) and are preferred over web browsers. Today's application users more often live beyond the secure boundaries of an organization. This opens up unprecedented exposure of applications to malicious users. Unrestricted access to an application provides hackers ability to analyze significant amount of application code even in the form of binaries.

Hackers perform binary analysis using tools like IDA Pro that provides wealth of information about an application such as structure of the application code, function calls, string literals etc. Even behavioral runtime analysis of an application can be performed using these sophisticated tools. The analysis can also expose the location of security or encryption related code, location of hard coded encryption keys, private key PIN, passwords and other sensitive information. Without binary obfuscation and other complimenting measures, security controls placed inside an application code including cryptographic techniques are ineffective.

An attacker can easily bypass security controls including cryptographic techniques by manipulating an application binary. Binary patching and runtime manipulation are the main techniques used by hackers to bypass specific blocks of security code inside an application. It is important to note that hacker's ability to debug an application is the key to runtime manipulation. Tools like Cheat Engine help hackers to perform runtime manipulations of an application.

To prevent an application from binary patching, it is essential to have binary integrity checks. However, these checks also need to be obfuscated and protected against runtime manipulations. Hence, developers need to place enough anti-debugging controls in their application code to prevent itself from being debugged.

Integrity checks and binary protection are not just limited to the main application binary but even to the modules and third party shared libraries used by the application.

It is possible for an attacker to manipulate input and output from shared libraries through a proxy library or API hooking. An application need to check integrity of all the libraries that it uses to prevent proxy libraries. As much as possible security critical applications should avoid shared libraries and choose static linking instead. This will significantly reduce the attack surface.
Today's applications, especially thick clients such as mobile apps are being used for all purposes including fun, social network, business and even banking. These application binaries contain significant amount of security code and cryptographic techniques. Unless enough binary protection is applied on these applications, they become the target of security exploits.

The techniques discussed in this short article are the master security controls that are required to ensure that other security controls placed in the application are effective. In the absence of these master security controls, all the other security code written to protect the application has no effect in the hands of an able hacker.

Unsafe HTTP Methods and its Emerging Risks

Chetan Soni
Project Engineer
A-34 Industrial Area,
Phase VIII,

In the penetration testing of a web application, testing the HTTP methods always plays an important role. These types of vulnerabilities are easy to find, but it is not easy to use when it comes to performing penetration test against the web application. This paper will describe in brief why these HTTP methods are dangerous and how to use such a method for the penetration test.

Introduction

If you are using HTTP Protocol for surfing web, you usually use GET and POST Methods. However, HTTP Protocol has several other methods like TRACE, CONNECT, DELETE, PUT, OPTIONS etc. which have some server risk if these methods are open in your web server.

GET requests are the most commonly and widely used methods which is used to retrieve the data from a server at the specified resource whereas POST requests are used to send data to the server to create or update the resource. The data sent to the server is stored in the request body of the HTTP request.

HTTP methods can be used to help developers in the deployment and testing of web applications. On the other hand, when they are configured improperly, these methods can be used for malicious activity.

According to RFC2616, there are other 6 HTTP methods are available for HTTP/1.1 Protocol which are OPTIONS, PUT, DELETE, CONNECT, TRACE and HEAD. There are also extended HTTP methods such as PROPFIND, MOVE, COPY, LOCK, UNLOCK and MKCOL etc.

HEAD, GET, POST and CONNECT are completely safe methods but PUT, DELETE and TRACE are the most dangerous methods as it was originally intended as file management operations. Obviously if these methods are enabled on a web server, it opens you to some dangerous attacks like File Upload Vulnerability, Mass Defacement and Server Rooting.

This paper will explain such techniques further by providing a more explanation and a demonstration of their usage.

Compromising the target

Extracting relevant information can play a game changing role in many situations which is pretty simple and somewhat easy. It can further divide into 3 categories:

- Reconnaissance
- Vulnerability Discovery
- Exploitation

Reconnaissance

Suppose, you are doing an external/internal penetration test of a big organization with DMZ, Data centers, Telecom network etc. Moreover, the only information that we know at this moment is the company name and/or it's domain name such as example.com.

If you're outside the network, then you need to figure out the attack surface area first with:

- Domain/Sub-domains present
- IPAddresses/Network ranges etc.
- Finding open ports
- Email addresses or people working for the organization

When we are inside the organization then there are two common situations:

- External Consultant (with no internal access)
- Potentially, posing as an employee (having already access)

At this stage, the penetration tester runs an NMAP scan against the target/network and find the following information:

- 192.168.73.130 – A Linux machine with TCP Port 80 open

```
root@kali:~# nc 192.168.73.130 80
OPTIONS / HTTP/1.1
Host: 192.168.73.130

HTTP/1.1 200 OK
Date: Mon, 19 Nov 2018 09:01:12 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
DAV: 1,2
MS-Author-Via: DAV
Allow:    OPTIONS,GET,HEAD,POST,PUT,DELETE,TRACE,PROPFIND-
,PROPPATCH,COPY,MOVE,LOCK,UNLOCK
Content-Length: 0
Content-Type: httpd/unix-directory
```

Since Port 80 is open, the pentester does a further check and finds out that HTTP methods are enabled or not. There are so many ways to check the HTTP methods against any target; the easiest way is to check with the help of Netcat.

Vulnerability Discovery

As this point, we know that the port 80 is listening on host <192.168.73.130>, which runs a Linux machine with WebDaV as a webserver, as well as which HTTP methods are enabled on host as shown in above screenshot.

In the world of web protocols and APIs it predates both SOAP/XML and RESTful architectures. Despite its longevity, WebDAV implementations can be quirky. Many servers and clients implement subsets and extends the set of standard HTTP methods and headers to provide the ability to create a file or folder, edit a file in place, copy or move or delete a file, etc.

As an extension to HTTP, WebDAV normally uses port 80 for unencrypted access and port 443 (HTTPS) for secure access.

A range of applications have the ability to work with files accessed via WebDAV. The application's file selection dialog supports entering not just a local filename, but a WebDAV URL, with the username and password needed for the WebDAV server. These applications include Microsoft Office (Word, Excel, etc); Apple iWork (Pages, Numbers, Keynote); Adobe Photoshop and Dreamweaver; and others.

In our case, you can access the WebDaV server at http://192.168.73.130/dav/.

Exploitation –

The penetration tester needs to take several steps in order to exploit the above target with PUT method.

The simplest and most basic form of identifying HTTP Servers is to look at the Server field in HTTP response header with the help of either Netcat or Burp Suite tool.

Curl is another useful utility through which you can test any HTTP method whether its open or closed.

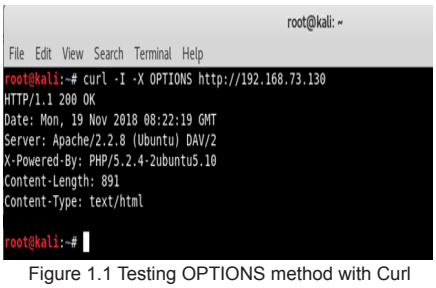


Figure 1.1 Testing OPTIONS method with Curl

For Example, if you want to test the OPTIONS method against the target example.com then the command is:

With the help of Curl, you can even send a HTTP request with custom method to a custom server as shown below:

root@kali:~# curl -X ABC -I -H "New Header: CHETANSONI" http://example.com/

In the same way, you can upload any malicious file or shell with the help of PUT method as shown below:

root@kali:~# curl -i -X PUT -H "Content-Type: text/plain; charset=utf-8" -d "Your Text" http://example.com/malware.php

As you can see that, a new file has been created of name malware.php remotely without any authentication of response code 201 Created.

Following services are also using PUT and DELETE HTTP methods:

- https://developers.google.com/drive/v2/reference/files/delete

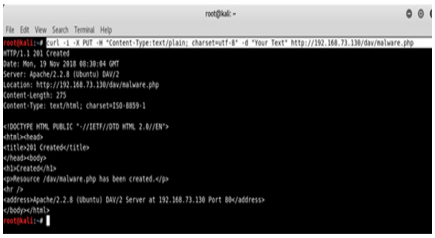


Figure 1.2 Uploading File with PUT method

- http://developers.facebook.com/docs/reference/api/Comment/
- http://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectDELETE.html
- http://www.salesforce.com/us/developer/docs/api_rest/index.htm
- http://developer.tradeshift.com/rest-api/#tsapi.conventions
- https://developer.paypal.com/webapps/developer/docs/api/#delete-a-stored-credit-card

Conclusion –

The main problem related to these HTTP methods was that the server operators weren't aware of their existence introducing the possibility of HTTP Verb Tampering.
For a professional penetration tester, testing web technology has become one of the most basic and important skills need to have. Testing HTTP methods for a web application or server is just one part of such testing; the results can be considered minor findings during a test, but this simple technique can open the door to the next level. Furthermore, an attack using such techniques can be devastating to critical websites. Although it seems very simple, so it is wise for penetration testers to practice the approach in order to become more knowledgeable and proficient in its use.

References:

- *OWASP: https://www.owasp.org/index.php/Test_HTTP_Methods_(OTG-CONFIG-006)*
- *Yeahhub: https://www.yeahhub.com/http-put-method-exploitation-live-penetration-testing/*
- *SANS Reading Room: https://www.sans.org/reading-room/whitepapers/testing/penetration-testing-web-application-dangerous-http-methods-33945*
- *Security Exchange: https://security.stackexchange.com/questions/38635/how-is-http-put-and-delete-methods-insecure-if-they-really-are*
- *RFC2616: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html*

Secure Your Android Mobile Apps

Are you the Android Developer ? Then this is for you!

Mr Vanaparthy Seshi Shekar Prasad
Project Engineer
Centre for Development of Advanced Computing (C-DAC), Hyderabad

Since the Android is free and open source platform, it has changed the mobile world. Though Android has become one of the popular mobile platform there are number of security concerns that are increasing exponentially. Because of increase in number of available applications, a single click can lead to the obstruction in security. The open source of the platform always welcomes the changes in the application and countermeasures. These changes can even help or violate the security, therefore, it's more necessary to focus on protecting the device from hackers. The loopholes and the bugs of the application can act as favorable conditions for the hacker to exploit. For a good smart phone there are millions of apps that offers various functionalities that covers all aspects of our day to day lives. Most of these apps are free to download and they are user friendly. All smartphones, as computers, are preferred targets of attacks. These attacks exploit weaknesses in smartphones that can come from the communication mode like Short Message Service (SMS, or text messaging), Multimedia Messaging Service (MMS), Wi-Fi, Bluetooth and GSM, the default global standard for mobile communications. According to a finding by McAfee in 2008, 11.6 percent users had been affected by mobile malware, but only 2.1 percent had personal experience on such problem. However, this number is expected to grow.

As a good developer you must be following good coding practices and taking care of certain things like having appropriate permissions, proper validations,2 factor authentication where ever necessary etc., Apart from these Let us understand few of the secure coding practices which when followed, your mobile app will have an added security layer.

Root Device Detection:

Android is one of largest popular mobile operating systems being used all around the world today. Every Android phone is running on the Linux kernel and middleware similar to a Linux operating system which we will install in our computer. Rooting is the process of allowing users to attain privileged control (known as root access) over various Android subsystems. When the device is rooted the user can access the internal storage of the devices and he can view and modify the data.

This article is intended for developers who write code for mobile applications. In view of growing usage of Android mobiles and attacks over it, it's also the responsibility of the developer to ensure that his code would not open up gates for attacks which can impact the end users mobile.

So, the developers should write a code in such a way that the application should not run in the rooted device to prevent the attackers from accessing the sensitive data.

The following code snippet can be made part of your main activity program (defined in manifest file) which will prevent your application to run on a rooter device.

Snippet 1 :
Code example for Root Detection

```
public class DeviceUtils {
private static boolean checkRoot-
Method() {
Boolean b = true;
String[] arrayOfString = new
String[10];
arrayOfString[0] = "/system/app/Su-
peruser.apk";
arrayOfString[1] = "/sbin/su";
arrayOfString[2] = "/system/bin/
su";
arrayOfString[3] = "/system/xbin/
su";
arrayOfString[4] = "/data/local/
bin/su";
arrayOfString[5] = "/data/local/
xbin/su";
arrayOfString[6] = "/system/sd/
xbin/su";
arrayOfString[7] = "/system/bin/
failsafe/su";
arrayOfString[8] = "/data/local/
su";
arrayOfString[9] = "/su/bin/su";
int j = arrayOfString.length;
int i=0;
if(i<j)
{
if(!new File(arrayOfString[i]).
exists())
{}
}
for(;;)
{
return b;
i++;
break;
b=false;
} }

public static Boolean isDeviceRoot-
ed()
{
return checkRootMethod();
} }
```

There are some open source libraries as well for detection: You can give reference to them also
eg. https://github.com/scottyab/rootbeer.

App Signature Verification :

By default, the Android OS requires all applications to be signed in order to be installed. In very basic terms, this means that the application signature is used to identify the author of an application (i.e. verify its legitimacy), as well as establish trust relationships between applications with the same signature.

The app signature will be broken if the .apk is altered in any way , unsigned apps cannot typically be installed. We can imagine an attacker removing license-checking code to enable full app features without paying, for instance by altering the .apk to enable full app features or include malware in a legitimate app to harvest sensitive user data. For the altered .apk to be installed, the attacker must resign it. Now, when you run checkAppSignature() on your app , when signed with your release developer certificate , you should see that it returns 0, i.e. valid.

The following code snippet can be made part of your main activity program (defined in manifest file) which will prevent your application to run on a rooter device.

Snippet 2 :
APK Validation Process Code sample

```
private static final int VALID = 0;
  private static final int INVALID
= 1;
  public static int checkAppSigna-
ture(Context context) {
    try {
 PackageInfo packageInfo = con-
text.getPackageManager().getPack-
ageInfo(context.getPackageName(),
PackageManager.GET\_SIGNATURES);
      for (Signature signature :
packageInfo.signatures)
 {
        byte[] signatureBytes =
signature.toByteArray();
        MessageDigest md = Message-
Digest.getInstance("SHA");
        md.update(signature.toByte-
Array());
        final String currentSigna-
ture = Base64.encodeToString(md.
digest(), Base64.DEFAULT);
        Log.d( "Current SIGNATURE:"
+ currentSignature);
        //compare signatures
        if (SIGNATURE.equals(cur-
rentSignature)) {
          return VALID;
        };
      }
    } catch (Exception e) {
//assumes an issue in checking
signature., but we let the caller
decide on what to do.
    }
    return INVALID;
  }

//the original signature of the ap-
plication in different class

private static final String SIGNA-
TURE = "478yYkKAQF+KST8y4ATKvHkYi-
bo=";
```

Verify the installer :

Each app contains the identifier of the app which installed it. Generally the app can install in the mobile devices either through Play-Store or through adb commands. The application should install in the mobile devices only through Play-Stores not through any other media. To enforce this, you can add the following code snippet in your main activity program (defined in manifest file).

Snippet 3 :
Runtime apk verification installer code sample

```
private static final String PLAY\_
STORE\_APP\_ID    =    "com.android.
vending";
  public static boolean verifyIn-
staller(final Context context) {
final String installer = context.
getPackageManager().getInstall-
erPackageName(context.getPacka-
geName());
    return installer != null
        && installer.
startsWith(PLAY\_STORE\_APP\_ID);
  }
```

This snippet shows getting the InstallerPackageName from the PackageManager and verifying against the known value of the Google Play Store, com.android.vending.

Do not work when proxy is enabled in the mobile network :

By default the apps works on both WI-FI and mobile network. But when the application is working on WI-FI network, there are chances to intercept the mobile traffic using the proxy tools. Thus, as a security practice check whether the proxy is enabled by using the following code snippet in your main activity program (defined in manifest file).

Snippet 4:
Runtime wi-fi proxy verification code sample

```
public boolean isProxyEnabled()
  {
    boolean wifi = isWifi();
    boolean bool = false;
    if (wifi)
    {
      String str = System.getProp-
erty("http.proxyHost");
      Object localObject = new
StringBuilder();
      ((StringBuilder)localObject).
append(str);
      ((StringBuilder)localObject).
append(":");
      ((StringBuilder)localObject).
append(System.getProperty("http.
proxyPort"));
      str = ((StringBuilder)local-
Object).toString();
      localObject = TAG;
      StringBuilder localString-
Builder = new StringBuilder();
      localStringBuilder.append("
proxyAddress : ");
      localStringBuilder.ap-
pend(str);
      localStringBuilder.append("
IsEmpty ?");
      localStringBuilder.ap-
pend(str.contains("null"));
      LogUtil.info((String)local-
Object, localStringBuilder.to-
String());
      if (!str.contains("null")) {
        securityFailureFlag = 1;
      }
      if ((!str.contains("null"))
|| (hasCertificateInstalled())) {
        bool = true;
      }
      return bool;
    }
    return false;
  }
```

When the app starts during the runtime, it will check for the network proxy is enabling or not. If it is enabled then the app will get terminated. If this feature is enabled, the attacker cannot intercept the packets using the network proxy tools.

By using these code snippets in your android apps, you can secure your mobile applications.

Mitigation against Mobile Application and Operating System Attacks

- *Update the mobile operating system regularly.*
- *Upgrade the operating system to its latest version.*
- *Always install applications from trusted sources.*
- *Consider installing security software from a reputable provider and update them regularly.*

How to make your smart contracts secure

Roshan Singh (Student)
Department of Computer Science and Engineering. Central Institute of Technology, Kokrajhar

These days "Blockchain" is everywhere. With its first application as a peer to peer cryptocurrency Bitcoin [1] introduced in 2009, the technology has continuously evolved. "Ethereum" is yet another blockchain platform. Unlike Bitcoin, the Ethereum Blockchain supports smart contracts. A smart contract[4] is a self-executable computer code having some predefined set of rules running on top of a blockchain. With the notion of " Code is Law" smart contracts are used to establish trust among various untrusted parties. However, one important aspect of these smart contract which people and developers often oversee is that unlike any other software these smart contracts are immutable they cannot be changed once deployed over the blockchain. So, there exists no scope for fixing any bugs identified after the deployment. Any mistakes in the code can literally claim millions of dollars like the --DAO attack [5], 2016. So ultimate care should be taken while writing these smart contracts especially when it has to deal with valuable assets like money. As a human one cannot guarantee to commit any mistakes.

So, how can I secure my smart contracts ?

Best practices performed while writing smart contracts can help eliminate most of the serious mistakes that the programmers often do. I am using the Solidity v0.4.24 programming language here but most of the logic remains the same for any smart contract programming language.

Always specify access modifiers to your functions:

Solidity[3] by default considers a function to be public if no access modifiers are explicitly mentioned for that. That means anyone internal or external to the contract can call the function. This is exactly what caused the Parity hack, 2017 which swept away about 150K ethers from user accounts.

Perform good accounting and avoid Reentrancy inside your functions:

A computer code is called reentrant if it can be interrupted in the middle of its execution and then safely be called again i.e the control flow of the program can be "re-entered". This reentrancy vulnerability was exploited by the attackers performing the DAO attack which drained about 3.6 million ethers from the contract. DAO stands for Decentralized Autonomous Organisation its an organizational setup which runs on computer programs that is transparent and is not influenced by any central authority. The DAO which I am referring here operated on the Ethereum Blockchain. The following code snippet shows how a re-entrancy can happen in a function.

```
contract TokenLocker{
    mapping (address=>uint) private myTokens;

    function withdraw_myToken() public {
      uint amount = myTokens[msg.sender];
      myTokens[msg.sender]=0;
      require(msg.sender.call.value(amount)());
}
}
```

Fig (a) Vulnerable code

The function *withdraw_myToken* lets an user to withdraw his fund from the contract. Ethereum Virtual Machine (EVM) the execution environment for smart contracts residing on Ethereum[2] blockchain executes the code sequentially. As one can notice that the balance of the account is set to 0 after the function call to withdraw the amount is made. It's a vulnerability, an attacker can call this function again and again before the account balance is set to 0 and the contract will literally allow this since it is written in this way only. Thus allowing the attacker to get extra token from the contract which he never owned. This major vulnerability can be fixed with a single line arrangement in the code.

```
contract TokenLocker{
    mapping (address=>uint) private myTokens;

    function withdraw_myToken() public {
      uint amount = myTokens[msg.sender];
      myTokens[msg.sender]=0;
      require(msg.sender.call.value(amount)());
}
}
```

Fig (b) Vulnerability fixed code

As you can see now that we are doing the accounting first, before releasing the tokens. A malicious user won't be able to get anything after the first call.

Consider data types carefully :

One should also be very careful while choosing the data type for its application. Like if an uint is made to be less than 0, it will cause an underflow and will get set to its maximum value. As one can see in the following code. The if condition will never get executed and b value will be as 9.

```
unit a = 8;
 unit b = 9;

  function x() public {
   a = a-b;
   if(a<0){
      b=b+100;
  }
}
```

The condition could have met if we considered the data type to be an int.

Some quick tips for you to make your smart contracts secure:

- "Keep your contracts simple" - Don't overload your contract with too much of complex computations. Avoid looping as much as you can. Smart contracts are meant for implementing logic based on rules not for resource intensive complex computations.
- Stay up to date -- The blockchain community is evolving daily one should always be in touch with the community to be updated.

And at last I will encourage you to get your hands dirty ! That's the only way one can learn

References:

- *Bitcoin: A Peer to Peer Electronic Cash System, Satoshi Nakamoto. https://bitcoin.org/bitcoin.pdf*
- *A Next Generation Smart Contract and Decentralized Application Platform, Vitalik Buterin, https://github.com/ethereum/wiki/wiki/White-Paper*
  https://solidity.readthedocs.io/en/v0.4.24/
- *Formalizing and Securing Relationships on Public Networks, Nick Szabo,1997. https://ojphi.org/ojs/index.php/fm/article/view/548/469*
- https://www.coindesk.com/understanding-dao-hack-journalists

FUZZ using ZZUF

Dr. Parag H. Rughani
Associate Professor
[Digital Forensics]
Institute of Forensic Science
Gujarat Forensic
Sciences University
Gandhinagar, Gujarat - India

Fuzz testing or fuzzing is a type of software testing in which deliberately malformed or unexpected inputs are delivered to target to see if failure occurs. Here software is anything that is compiled from source code into executable code that runs on some sort of processor, including systems, desktop applications, mobile applications, embedded system firmware, system on a chip and more.

When a software fails accidentally due to unexpected or malformed input, it is a robustness problem. When software fails due to a deliberate attack, it is a security problem. A software failure that causes harm or death to humans is a safety problem. Robustness, security and safety are 3 faces of the software bugs. A bug is a mistake made by a developer; under right conditions, the bug is triggered and software does something it was not supposed to do.

Fuzzing is the process of sending intentionally malformed inputs to a piece of software to see if it fails. Each malformed input is a test case. Failure includes a found bug, which can be then fixed to improve robustness and security of the target software.

A fuzzer is a piece of software that test a piece of target software. A proper fuzzer consists of three components:

- The test case generator, test engine or anomalizer is responsible for creating test cases.
- The injector, delivery mechanism or test driver sends the test cases to the target.
- The oracle determines if the target has failed.

This article focuses on one of the open source fuzzers is zzuf (reverse of fuzz). Following sections explain use of zzuf.

Zzuf is a transparent application input fuzzer. Its purpose is to find bugs in applications by corrupting their user-contributed data (which more than often comes from untrusted sources on the Internet). It works by intercepting file and network operations and changing random bits in the program's input. Zzuf's behaviour is deterministic, making it easier to reproduce bugs.

Its main areas of use are:

Quality Assurance: use zzuf to test existing software, or integrate it into your own software's test suite.

Security: very often, segmentation faults or memory corruption issues mean a potential security hole, zzuf helps exposing some of them.

Code Coverage analysis: use zzuf to maximise code coverage.

Zzuf's primary target is media players, image viewers and web browsers, because the data they process is inherently insecure, but it was also successfully used to find bugs in system utilities such as objdump.
Zzuf can be downloaded from: https://github.com/samhocevar/zzuf/releases

Basic Usage:
dd if=first.txt | zzuf -r 0.5 > fuzzed.txt

Figure 1

Above command reads content from first.txt and applies 0.5 fuzz ratio for fuzzing the data. The resultant data is stored in fuzzed.txt as shown in the figure.

The zzuf tool can be used to fuzz test any application as shown in following figure:

zzuf -s 0:1000000 -c -C 0 -q -T 3 objdump -x /home/parag/Downloads/sample.exe

Figure 2

In the above command we are testing sample.exe, the target file. The -s means we'll try one million seed values. The -c means zzuf should only fuzz the files given on the command line. This is useful because otherwise often the tools will already throw error messages from reading config files or other things, so they won't really get to our fuzzed input. -C 0 means zzuf should not stop after the first crash found. -q suppresses the output of our fuzzed command. -T 3 sets a timeout of three seconds so zzuf won't hang if we run into an endless loop.

As you can see in above figure you get signal 11 (SIGSEGV), this means zzuf spotted a segfault with parameters -s 1048 and -r 0.004.

Now we re-create this malformed file using command:

zzuf -r 0.04 -s 1048 < /home/parag/Download/sample.exe > crash.exe

Figure 3

As it can be seen from above figure, a crash.exe file is created. This crash file can be further analysed to understand the vulnerability.

Happy FUZZING!!!

References:

- *What is Fuzzing: The Poet, the Courier, and the Oracle*
- *http://caca.zoy.org/wiki/zzuf*
- *https://fuzzing-project.org/tutorial1.html*
- *https://declara.com/content/35DYjjLa*

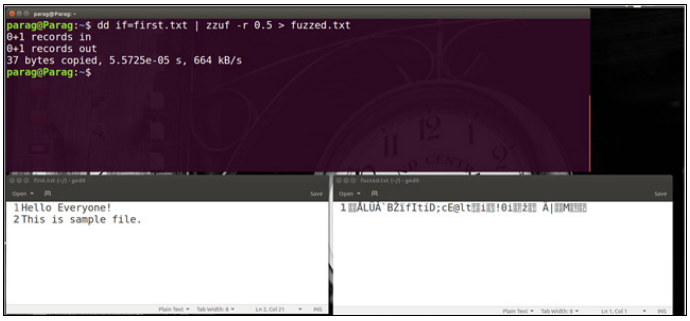


Figure 1

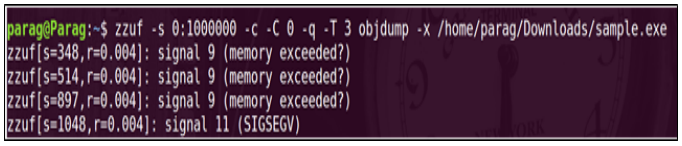


Figure 2

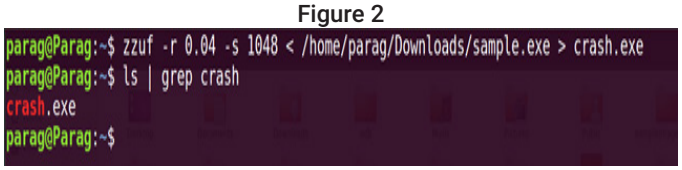


Figure 3

Information Security Services [ISS]
Your Security, Our Priority

C-DAC
Security Services Portfolio

IT Audit and Compliance Services

We help you meet the requirements of all the current and future State, Central Government and International standards and Information security regulations, any newly proposed legislation from Central Government which may apply to your business.

Our programs include:

- ISMS 27002 Services and support to ISO 27001 Services
- OWASP compliance Security audit of web applications and web services
- OWASP compliance security audit of mobile applications
- Infrastructure Security Audit
- Abuse based audit

Vulnerability Assessment & Penetration Testing

We help you by providing Assessment for several vulnerabilities in Internet and Intranet devices by validating open ports, services, domain names, IP network ranges operating system and applications, to identify systems on the network and exposed services.

Our program includes:

- Perimeter Security Testing
- System hardening
- Web Apps Security Assessment
- Vulnerability Assessment
- Penetration Testing
- Wireless Network Assessment
- Architecture Review

Security Solutions

We help you in implementing our in-house developed security products and provide end user support

- Application and Device Control (ADC) - a centralized application & device control solution
- AppSamvid - an application whitelisting solution for the desktops
- JSGuard: JSGuard -Browser Add-on) detects & defends from JavaScript & HTML tags-based attacks
- PHP Application Vulnerability Scanner - PAVS is a source code scanner for finding the code vulnerabilities in PHP based web applications
- USB Pratirodh: It is a USB control software targeted towards securing end systems from unauthorized usage of portable USB storage devices.
- m-Safe : Mobile SDK for secure communication and storage

Other Security Services

We help you in analysis and handling the following:

- Incident Handling
- Intrusion Analysis
- Malware Analysis
- Implementation of Security services
- Design & development of policies, procedures and guidelines
- Risk Management and Disaster Recovery Services
- Business Continuity Process Development
- Information Security Awareness Training
- High End Information Security training for organizational needs
- Risk Assessment
- Log Analysis
- Incident Analysis
- Database analysis

C-DAC Expertise

Product Development → Proof of Concept Labs → Cutting Edge Research → Security Consultancy → Security Training & Awareness

International Trained and Certified Professionals

Standards, Methodologies and Guidelines followed by C-DAC

- ✓ Cert Guidelines
- ✓ Open Web application Security Project (OWASP)
- ✓ ISO/IEC 27002
- ✓ ISO/IEC 27005
- ✓ ISO/IEC 27033
- ✓ PCI -DSS
- ✓ NIST – National Institute of Standards & Technology
- ✓ CIS Benchmarks – Centre for Internet Security
- ✓ OSSTMM – Open source security testing methodology manual
- ✓ Best practices from SANS, ISACA, COBIT etc.,

More than 80 Certified Professionals in CEH, ECSA, CISA, CISM, CISSP, SANS GIAC, ISMS

C-DAC is Cert-In Empaneled Organization

for more Information contact: Coordinator, Information Security Services,
Centre for Development of Advanced Computing, (C-DAC),
(A Scientific Society of Ministry of Communications and Information Technology, Government of India)
Plot No. 6 & 7, Hardware Park, Sy No. 1/1, Srisailam Highway,
Pahadi Shareef Via Keshavagiri (Post), Hyderabad - 501510
Phone: +91-7382303598; email: cswan@cdac.in

ONLINE SHOPPING THREATS for women

Online shopping — the glorious invention which allows people to buy things from the comfort of their homes. No more travelling to multiple stores to find the right product; no more having to deal with over-enthusiastic sales persons; no more standing in long lines at the checkout counter. The e-commerce boom has certainly changed the way we shop for the better. But, like everything else, the world of online shopping is not all roses. Despite all the efforts of e-commerce companies to alleviate them, there are a few problems that customers still have to face while shopping online

Let's look into few ways that cyber criminals may target women

Expensive branded products at low cost:
In social networking sites very often we get advertisements showing expensive branded products at unbelievable prices. This catches attention of customer's most likely women and they may end up paying money for those products which are not genuine. For example branded bags, clothes, costly phone and beauty products.

Natural remedies for weight loss:
Most often in our social networking and Instant messaging Applications we get messages giving tips on weight loss and for further inputs they request for payment to purchase their product. Women who are desperate to weight loss get trapped by these messages. They end up paying money for fake products.

Expensive Jewelry:
Cyber-criminals may spoof certain online jewelry websites and give exciting discounted offers for jewelry products targeting women customer. They purchase products online with certain value but end up receiving some other products of lesser value. And they feel cheated and when they raise complaint to the original website they just deny that purchase happened through their website. This may lead to loss of your money?

Risks in online shopping

A few questions you need to check before you start online shopping

Brand
is the e commerce site genuine?

Security
Is your credit card safe?

Privacy
Is your information being sold?

Shipping
Are you getting the correct product at the requested time?

Tips for safe online shopping

- **Keep computer OS updated:**
  Make sure your PC is secured with an antivirus, anti spyware, firewall, system updated with all patches and web browser security with the trusted sites and security level at high.

- **Shop only through trusted sites:**
  Research about the web site that you want to buy things from, since attackers try to trap with websites that appear to be legitimate, but they are not. So make a note of the telephone number's physical address of the vendor and confirm that the website is a trusted site. Search for different web sites and compare the prices. Check the reviews of consumers and media of that particular web site or merchants.

- **Check the security aspects of the website:**
  If you are ready to buy something online check whether the site is secure with https or padlock on the browser address bar or at the status bar and then proceed with financial transactions.

- **Keep track of your digital payments:**
  Immediately check the credit card statements as soon as you finish and get them to know about the charges you paid were same; and if you find any changes immediately report to concerned authorities.

- **Don't save the card details or bank details on websites:**
  Do not store the card no either debit or credit on the shopping websites.After finishing your online shopping clear all the web browser cookies and turn off your PC since spammers and phishers will be looking for the system connected to the internet and try to send spam emails and try to install the malicious software that may collect your personal information.

- **Never respond to email which asks about your purchases:**
  Beware of the emails like "please confirm of your payment, purchase and account detail for the product." Remember legitimate business people never send such emails. If you receive such emails immediately call the merchant and inform the same.

- **Change passwords frequently:**
  Don't use a single password for a long time, change your Email id, bank account, credit-debit card passwords frequently.

- **Different passwords for different websites:**
  If hackers crack your one password they may crack all others if you are using same or similar password for all. So use different t password for all websites. However it is more complex to remember all passwords but it will add the Safety layer too.

- **Use Secured Networks:**
  Always use secured internet connection. Public Wi-Fi spots are Vulnerable to cyber attacks.

- **Don't click on links offering discounts/ prizes:**
  Cyber criminals sent messages in featuring great discounts in popular e commerce websites. It is always better to check in the original website for offers rather than clicking on links received in WhatsApp groups or from unknown numbers.

For more details / queries on
Cyber Security visit or call us to our Toll free number

Ministry of Electronics & Information Technology, Government of India

www.infosecawareness.in 1800 425 6235

For Virus Alerts,Incident & Vulnerability Reporting
certin Handling Computer Security Incidents
http://cert-in.org.in/

www.cyberswachhtakendra.gov.in