**Blockchain security:** Blockchain vulnerabilities, Smart contract vulnerabilities, Blockchain on CIA security triad, Blockchain based DNS security platform, deploying blockchain based DDOS protection.

Blockchain on the CIA Security Triad

What is the CIA security triad?

Confidentiality

Integrity

Availability

Understanding blockchain on confidentiality

Confidentiality in the existing model

Businesses, blockchain, and confidentiality

Achieving confidentiality with Hyperledger Fabric

Blockchain on integrity

Integrity in the current blockchain network

Block arrangement and immutability

Achieving integrity with Hyperledger

Verifying chain integrity

Understanding blockchain on availability

Availability in the current blockchain network

No single point of failure

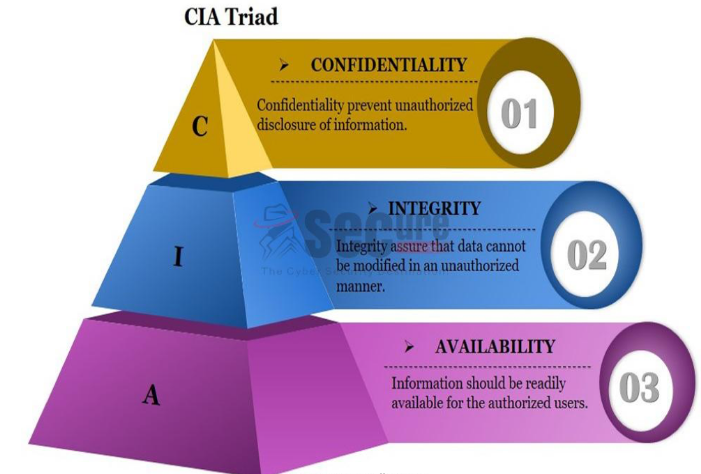
Business and availability

**Blockchain on the CIA Security Triad**

**Confidentiality, Integrity, and Availability** (**CIA**) security triad model is one of the oldest and most popular security frameworks connected with the blockchain structure. The CIA triad model is a model that helps organizations structures their security posture.

**What is the CIA security triad?**

* CIA is a framework/model that's used to arrange a list of security controls and systems used by the **information security** (**infosec**) team.
* The purpose of the triad is to deliver a standard framework to evaluate and deploy information security policies, independent of the underlying technology, network, or system.



**Confidentiality**

* Confidentiality is a way to keep information hidden from unauthorized people. In this current time of digital connectivity, everyone is aggressive enough to know that information that has been kept a secret.
* Security agencies are a prime **example** of a company breaking confidentiality so that they can perform forensics and use surveillance footage. Financially motivated cyber criminals do their best to break into security systems and gather confidential documents that will benefit their business adversaries.
* Organizations are spending millions of dollars every year to achieve fullstack confidentiality with cryptography and access control systems. Several methods are tested every day to protect data at rest and data in motion.

**Integrity**

* Integrity is a way to protect the unauthorized tampering of information. It is a mandatory compliance for every infosec body. It is also a method to maintain the consistency, accuracy, and trustworthiness of the respective data over its entire life cycle.
* There has to be complete security of the data, and any unauthorized access to it should be prohibited. Certain measures that aid these aforementioned things include file permission and user access controls.

**Availability**

* Availability refers to on-time and reliable access to data. The path of going from data to information and information to value means that the value will be illegitimate if the information is not available at the right time.
* **Distributed denial-of-service** (**DDoS**) and ransomware attacks are some of the most powerful weapons in the hands of malicious actors, and they use these attacks to keep information away from people who have authorized and legitimate access.
* Organizations make several attempts to combat these attacks, including web application firewalls, DDoS protection, **content delivery network** (**CDN**), and even disaster recovery.

**Understanding blockchain on Confidentiality**

Every digitally connected technology comes with the cost of security challenges, and these challenges can be about privacy exposure, confidentiality breaches, identity theft, and many more. Blockchain technology is a computing technology that runs over a digital ecosystem and hence it becomes important to pay attention to its fundamental security challenges.

**Confidentiality in the existing model**

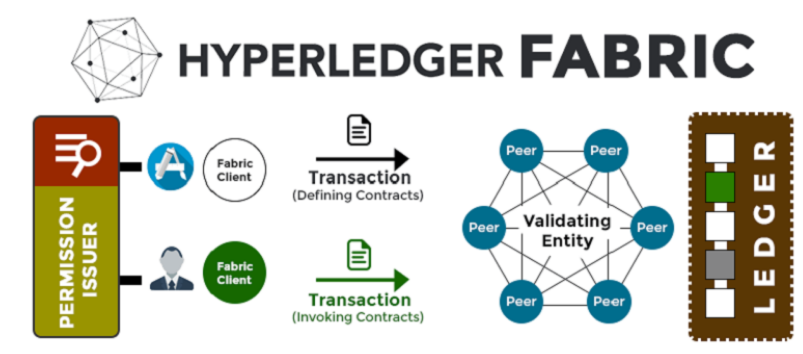
* As we already know, blockchain technology was never made to be restrictive in nature, as anyone with client software can participate in the block generation process, or mining, in the case of Bitcoin.
* Confidentiality with respect to the blockchain is simply about hiding transaction information from unwanted participants in the network. However, because of the open and Permissionless nature of the public blockchain such as Bitcoin, achieving a better confidentiality rank can be extremely difficult.

**Businesses, blockchain, and confidentiality**

* When it is about **business**, confidentiality becomes a critical pillar in the cyber security space to achieve better trust among customers and other stakeholders.
* The permissioned blockchain has gained a great appreciation as it allows only pre-selected participants to access the data in the distributed ledger network.
* When a business interacts with another business, it is not just about how much information to share, it is also about who should have access to which information under what conditions. While considering Hyperledger Fabric, IBM suggests that certain points should be kept in mind:
* With each transaction, it is important to know whether a participant can see complete information, a part of it, or no information at all. It has to be mentioned under a smart contract.
* If the regulator has been assigned, then they must confirm the extent of data accessed by the regulator.
* It is important to understand the nature of your network—static or flexible—as confidentiality parameters may change in the future, based on new participant roles and needs.

**Achieving confidentiality with Hyperledger Fabric**

**Hyperledger Fabric** is a modular **blockchain** framework that acts as a foundation for developing **blockchain**-based products, solutions, and applications using plug-and-play components that are aimed for use within private enterprises.



Hyperledger Fabric provides features to achieve confidentiality with the ease of calling a set of library files:

* **Attribute-based access control** (**ABAC**): The decision of users accessing a transaction is dependent on its identity. ABAC can support both chaincode and an entire fabric. The attributes used during transaction deployment has to be passed during Tcert creation by the user. It is an important step to determine whether a user can execute any specific chaincode.
* **Attribute Certificate Authority** (**ACA**) plays an important role in validating attributes and returning an **attribute certificate** (**ACert**). ACA maintains the database so that companies can store attributes for users and their affiliations.
* **Hyperledger Fabric encryption literary**: The smart contract can be configured to encrypt information or a subset of information in the transaction. This information will remain encrypted in the ledger with the key only being available to the peer who is supposed to see and access it. If the endorsement policy need peers from different organizations, then the information has to be encrypted before including it in the transaction proposal.

**Blockchain on integrity**

Encryption can only provide solid confidentiality against internal attacks can't protect data from corruption caused by configuration errors, software bugs, or espionage attempts. Although blockchain technology has its own solid approach in achieving immutability with the hashing algorithm and the Merkle tree model for integrity, we have to try and understand how it would practically work with real-world applications and Hyperledger Fabric

**Integrity in the current blockchain network**

Blockchain uses cryptographic hashing to ensure that the ledger remains tamper-proof. One of the key characteristics of this **hashing** function is that it is always oneway, which means it is logically impossible to get the data back from the hash result or from the message digest.

It is also difficult to analyze the pattern of message digest and predict the original data as even a slight change in the actual message can result in a big difference. Be it any flavour of blockchain, all of them use hashing extensively as follows:

* An Ethereum account identifier is created by hashing a public key with the Keccak-256 hashing algorithm
* A Bitcoin address is computed by hashing a public key with the SHA-256 algorithm.

**Block arrangement and immutability**

As we already know, each node stores the ledger in the form of connected blocks, and the creation of a new block depends on the hash of the previous block. This stops the possibility of malicious attempts to disturb, alter, or delete any blocks in the ledger. This helps organizations achieve a new level of cyber security integrity and provides a platform on which you can develop a tamper-proof business application.

**Achieving integrity with Hyperledger**

Committing a peer always validates the new block before adding it to the ledger. A situation where a peer is hacked means that the block may get compromised from the ledger. To avoid such a situation, there are certain methods to correct the way a block gets added in the ledger.

**Verifying chain integrity**

In this method, each peer periodically validates its blockchain and asks the Peer to recheck whether a broken block is detected. A function named CheckChainIntegrity() has to be called to keep the integrity check running:



**Understanding blockchain on availability**

Blockchain is a software application running on the cloud and keeps its value until it is not broken or disturbed. For users, the face of blockchain is simply a **decentralized application** (**dApp**), and in order keep it available all of the time, both the front end and the backend of the system should run seamlessly.

**Availability in the current blockchain network**

On-time and reliable access to information resembles availability. Cyberattacks such as DDoS cause huge disruption to internet services and result in websites becoming inaccessible, which costs businesses a lot of money.

*The decentralization nature of blockchain makes it harder to disrupt these applications.*

**No single point of failure**

Even if one node in the blockchain goes down, the information can be accessed and used by the rest of the nodes in the network. As all of the nodes keep the exact copy of the ledger, it will always be up-to-date. All of the nodes in the network are logically decentralized with their ledger, and there is a zero probability of system failure.

**Business and availability**

When it comes to the blockchain, its availability is determined by valid and successful transactions. For every business, keeping record of all transactions is a core function, and these transactions could be the entries of business activities, asset entries, supply chain management records, and many more.

Blockchain-Based DNS Security Platform

DNS

Understanding DNS components

Namespace

Name servers

Resolver

DNS structure and hierarchy&#xA0;

Root name server

Current TLD structure

Registries, registrars, and registrants

DNS records

DNS topology for large enterprise

Architecture

Challenges with current DNS&#xA0;

DNS spoofing&#xA0;

Blockchain-based DNS solution

X.509 PKI replacement

MITM-proof DNS infrastructure

Lab on Ethereum-based secure DNS infrastructure

Lab preparation

Namecoin blockchain installation

Installing PowerDNS

Installing DNSChain

**Blockchain-Based DNS Security Platform**

The **Domain Name System** (**DNS**) is mainly designed to resolve a host name query to an IP address. Internet users need to have domain names, such as www.packtpub.com, but the internet needs an IP address to route the request to the desired destination. This way, the DNS becomes the phonebook of the internet and allows everyone to use it globally at the same time it also leaves a high possibility of it getting misused.

**DNS**

For IT and security professionals, it is important to understand the basic structure, function, and operations of DNS. The DNS is a hierarchical database with delegated authority. There are two ways organizations can manage their DNS infrastructures—by allowing their **Internet Service Provider** (**ISP**) to manage it or by managing it internally. Any configuration mistakes or failure in the ISP network can turn down the organization's internet infrastructure.

With an efficient DNS deployment, organizations can even achieve better email spam-filtering systems and optimized network topologies. Here are just a few ways in which the DNS plays a vital role in organizations:

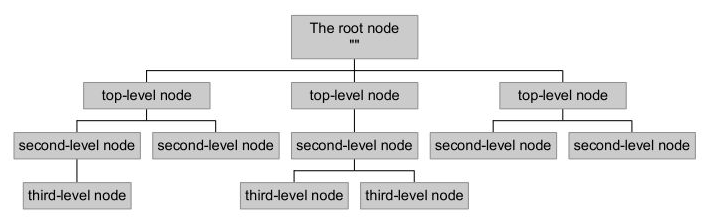
* **Anti-spam**: Some DNS mechanisms, including **Sender Policy Framework** (**SPF**) and **DomainKeys Identified Mail** (**DKIM**), ensure only a predefined list of domains should be allowed to send emails on behalf of a specific organization. These mechanisms are effective if the DNS in the organization is working properly.
* **Load sharing**: DNS services can optimize the server infrastructure by load sharing the traffic of highly utilized servers with other Underutilized servers.
* **Privacy**: DNS services ensure the privacy of an organization's namespace information by masking addresses with different names, depending on whether they are accessed from inside or outside of the network, helping to achieve stronger network security.

**Understanding DNS components**

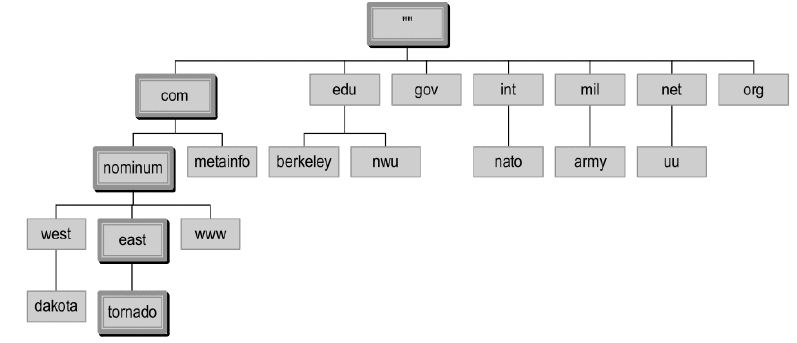
There are three core components of the DNS—the **namespace**, **server**, and **resolver**.

1. **Namespace**

A namespace is a structure of the DNS database. It is represented in the form of an inverted tree with its root node at the top. Each node in the tree has a label and the root node has a null label.



A domain name is the sequence of labels starting from a node to the root, separated by dots. The namespace can have a maximum depth of 127 levels and domain names can be of a size not more than 255 characters in length:



1. **Name servers**

Name servers are responsible for storing information about the namespace in the form of zones. There can be multiple name servers and ones that load a complete zone are said to be *authoritative* for the zone. Generally, there is more than one name server used as authoritative for a single zone, ensuring better redundancy and sharing the load:

There are two main types of name servers—**authoritative servers** and **Caching servers**:

* **Authoritative name server**: It provides responses to DNS queries. It is responsible for delivering original and definitive answers to each DNS query. There can be two types of authoritative name servers:
* **Master server (primary name server)**: It stores the original copies of all zone records. An administrator can only make changes to the master server zone database.
* **Slave server (secondary name server)**: A slave server keeps a copy of master server files. It is used to share DNS server load and to improve DNS zone availability.
* **Caching name server**: It brings the name service closer to the user and improves overall name lookup performance. It also provides a comprehensive mechanism for providing private namespaces to local users, by allowing users to obtain all names mapping from local caching.

1. **Resolver**

The name resolver is required to find out the name and IP address of the name servers for the root zone. The root name servers store information about top-level zones and direct servers in whom to contact for all **top-level domains** (**TLDs**). The resolver basically breaks the name into its labels from right to left. The first component, the TLD, is queried using a root server to obtain the designated authoritative server.

**DNS structure and hierarchy**

To deploy an internal DNS infrastructure, organizations can select any domain hierarchy; however, once connected to the internet, they have to follow the common DNS framework. Let's understand the name server hierarchy.

**Root name server**

With consistent namespaces across the internet, the root name server directly responds to requests for records in the root zone and answers other requests by returning a list of the authoritative name servers for the appropriate TLD.

In order to modify the root zone, a zone file has first to be published over the internet. The root zone file is published on 13 servers from *A* to *M* across the internet. The root zone contains the following information:

* Generic top-level domains such as .com, .net, and .org
* Globally recognized TLDs
* Country code TLDs, two-letter codes for each country such as .in for India or .no for Norway
* Globally recognized TLDs, generally similar to country code TLD Names

The root zone contains the numeric addresses of name servers that serve the TLD contents and the root server answers with these addresses when asked by a TLD.

When organizations get a new domain name, the registrar probably configures DNS records on their behalf and provides them with a **name server** (**NS**). Organizations need to have a name server to tell the internet's DNS directory the IP addresses of their web servers and corresponding services.

**Current TLD structure**

The TLD is one of the domains at the highest level of the DNS hierarchy.TLDs are installed in the root zone of the namespace. The domains in the last part of the system have to be recognized with fully qualified domain names. The **Internet Corporation for Assigned Names and Numbers** (**ICANN**) ensures that TLDs are managed by delegated organizations. The **Internet Assigned Numbers Authority** (**IANA**) is operated by ICANN and is responsible for managing the DNS root zone.

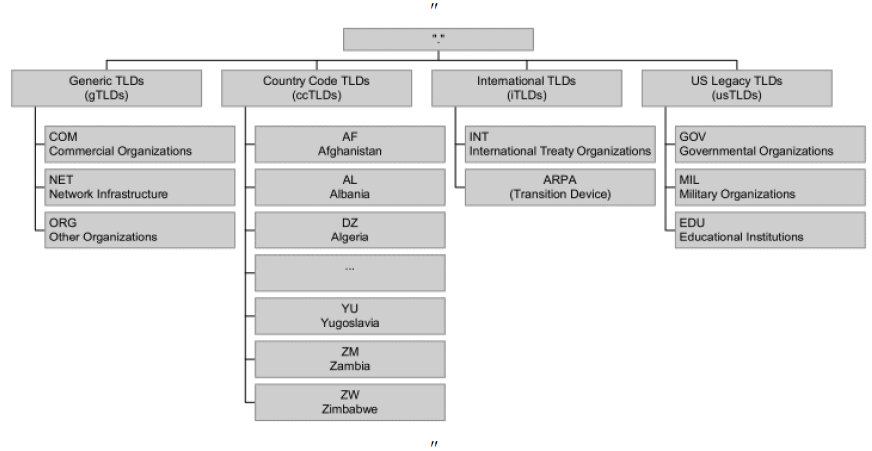
IANA is responsible for managing the following TLDs:

**ccTLD** — country-code TLDs

**gTLD** — generic TLDs

**.arpa** — infrastructure TLDs

This hierarchical diagram explains the existing TLD structure:

**

**Registries, registrars, and registrants**

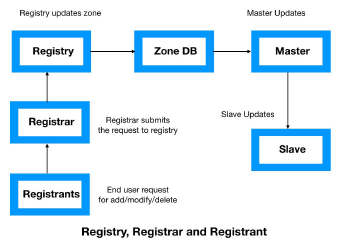
The DNS stores a massive database of domain names. In order to perform registration, there are three entities working together—**registry**, **registrar**, and **registrant**:

* **Registry**: An organization maintaining the database of namespaces that has edit rights to that database. The registry runs the authoritative NS for the namespace and manages the TLD names. Their role is in creating domain name extensions, setting up rules for the domain names, and working with registrars to provide domain names to the public. For example, **Verisign** manages the registration of .com domain names and their DNS.
* **Registrar**: An organization that reserves domain names and is accredited to sell domain names to the public. This registrar must be accredited by a **generic top-level-domain** (**gTLD**) registry or a **country code top-level domain** (**ccTLD**) registry. A registrar works under the guidelines provided by domain name registries.

Only a designated registrar can modify or delete information about domain names in the central registry database. End users buy domains directly from the registrar and the end user has complete rights to switch registrar, invoking a domain transfer process between registrars. Some of the most popular registrars are GoDaddy, HostGator, BigRock, and many more.

* **Registrant:** This is simply the end user who holds the rights to a domain name. As a domain name registrant, every person has certain rights and responsibilities, including access to information from the user's registrar regarding processes for registering, managing, transferring, renewing, and restoring the domain name registration.

Here is a diagram that shows the working of all three entities together:

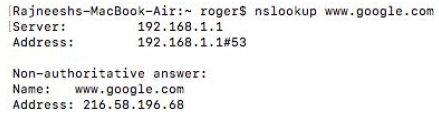


**DNS records**

* DNS records are mapping files that associate with DNS server whichever IP addresses each domain is associated with, and they handle requests sent to each domain.
* Various strings of letters are used as components that resemble the actions of the DNS server and these strings of commands are called DNS syntaxes. These syntaxes are A, AAAA, **Canonical Name** (**CNAME**), **mail exchanger** (**MX**), **pointer** (**PTR**), **name server** (**NS**), **Start of Authority** (**SOA**), **service** (**SRV**) **record,** **text** (**TXT**) and **Name Authority Pointer** (**NAPTR**).
* **SOA**: An SOA record notes the beginning of a zone file. It consists of the name of the zone, a technical point of contact, its NS, a serial number, and a timeout value:



* **NS**: The NS records identify the authorized name servers for the zone. The NS also delegates subdomains to other organizations on zone files. In the previous example, we can clearly see the list of NSes for ww.google.com.
* **Records**: Address records establish the forward binding from names to addresses. In this example, we have an IP address mapped with the domain [www.google.com](http://www.google.com):



* **MX records**: These records identify the servers that can exchange emails. A priority is always associated with each of the records, so the user can choose the primary and backup mail servers.
* **TXT records**: These records deliver a method to expand the information provided through DNS. This text record stores information about the SPF that can identify the authorized server to send email on behalf of your organization.
* **CNAME**: CNAMEs are essentially domain and subdomain text aliases to bind traffic. They indicate that the **Secure File Transfer Protocol** (**SFTP**) server is on the same system as the mail server. CNAME plays an important role, particularly when the server is not under organizational control such as a hosted or managed web server.
* **PTR records**: These records provide the reverse binding from addresses to names. PTR records should exactly match the forward maps.

**DNS topology for large enterprise**

For IT professionals, understanding DNS queries and the types of name server takes us most of the way to organizational DNS best practices:

**Network topology**:

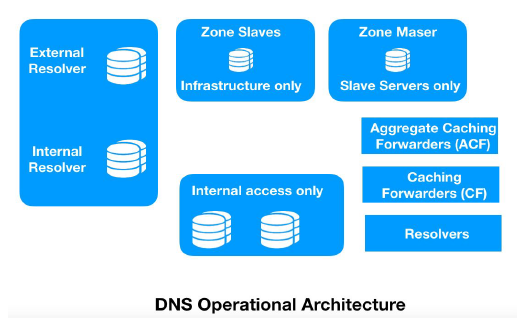
* Redundancy plays a critical role in domain infrastructure. Even if one server fails, another takes control to keep the service up and running.
* **BIND** (widely used DNS software) supports high redundancy through a master-slave relationship. The master NS updates the change in mapping to one or more slave servers through the zone transfer mechanism.

**Configuration files**:

* BIND's configuration is stored in a file called named.conf. This named.conf file helps the server to recognize the authoritative and/or caching server and whether it is the master or slave for any specific zone.
* The file points to zone files that contain the real mapping database. It contains lines or records that define name-to-address and address-to-name mapping for a specific domain.

**Architecture**

* With the changing technology and network transformation, DNS has had to be upgraded over time. There are bodies such as **DNS Operations, Analysis, and Research Center** (**DNS-OARC**) and **Internet Systems Consortium** (**ISC**).
* In the following diagram, we can see a standard DNS architecture built to optimize the DNS infrastructure:

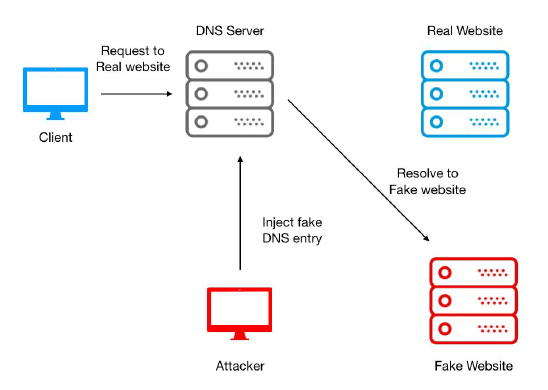


The preceding standard DNS architecture can be described as follows:

* **Master DNS zone:** The master zone contains a read/write copy of zone data. Only one master zone is allowed in a network. All the DNS records have to be written in the master zone manually or automatically. This data is then stored in a standard text file.
* **Slave DNS zone**: The slave zone is a read-only copy of the zone data. Usually, it is a copied version of master zones. If an attempt is made to change the DNS record on the secondary zone, it can redirect to another zone with read/write access. The slave DNS zone serves the purpose of backing up the DNS zone file.
* **Aggregate Caching Forwarder (ACF)**: It basically forwards the requests instead of processing them. When the server sends a response, it passes it back to its own client. In some situations, the resolver can also be a forwarder or caching forwarder. It may or may not cache the data; however, it is useful for systems such as **small office home office** (**SOHO**) gateways that want to provide DNS data to DHCP clients that don't have a predefined address for the DNS server.

**Challenges with current DNS**

Today, DNS has become the backbone of the internet and organization's networks. The DNS is mission-critical infrastructure that no organization can function without. However, despite growing investment in network and information security, attackers still manage to invade the network, and the DNS remains a vulnerable component in the network infrastructure that is often used as an attack vector. Firewalls leave port 53 open and never look inside each query. Let's look at one of the most widely used DNS-based attacks:



**DNS spoofing**

When a DNS server's records are altered to redirect the traffic to the attacker's server, the DNS gets hijacked. This redirection of traffic allows the attacker to spread malware across the network. DNS spoofing can be carried out in one of following three ways:

* **DNS cache poisoning**: An attacker can take advantage of cached DNS records and can then perform spoofing by injecting a forged DNS entry into the DNS server. As a result, all users will now be using that forged DNS entry until the time the DNS cache expires.
* **Compromising a DNS server**: A DNS server is the heart of the entire DNS infrastructure. An attacker can use several attack vectors to compromise a DNS server and can provide the IP address of a malicious web server against each legitimate DNS query.
* **Man-in-the-middle (MITM) attack**: In this type of attack, a threat actor keeps listening to conversations between clients and a DNS server. After gathering information and sequence parameters, it starts spoofing the client by pretending to be the actual DNS server and provides the IP addresses of malicious websites.

**Blockchain-based DNS solution**

* **DNSChain** is one of the most active projects to transform the DNS framework and protect it from spoofing challenges.
* DNSChain is a blockchain-based DNS software suite that replaces X.509 **public key infrastructure** (**PKI**) and delivers MITM proofs of authentication. It allows internet users to set a public DNSChain server for DNS queries and access that server with domains ending in .bit.

**X.509 PKI replacement**

X.509 is a standard framework that defines the format of PKI to identify users and entities over the internet. It helps internet users to know whether the connection to a specific website is secure or not. DNSChain has the capability to provide a scalable and decentralized replacement that doesn't depend on third parties.

**MITM-proof DNS infrastructure**

This uses a public key pinning technique to get rid of the MITM attack problem. Public key pinning specifies two **pin-sha256** values; that is, it pins two public keys (one is the pin of any public key in the current certificate chain and the other is the pin of any public key not in the current certificate chain):

* It works in parallel with existing DNS servers
* Websites and individuals store their public key in the blockchain
* The keys are shared over the DNSChain software framework

**Lab on Ethereum-based secure DNS infrastructure**

DNS infrastructure has been the most targeted asset of organizations. Traditional DNS is vulnerable to several sophisticated threats. The current DNS system is hierarchical and the system root server becomes the high value attack vector. Since the entire infrastructure is centralized, even a slight failure can lead to whole system failure. A group of engineers—Greg Siepak and Andrea Devers have developed an Ethereum-based DNS platform to connect client and name server without any involvement of third-party in between. The project is named **DNSChain** and hosted over GitHub at the link <https://github.com/okTurtles/dnschain>.

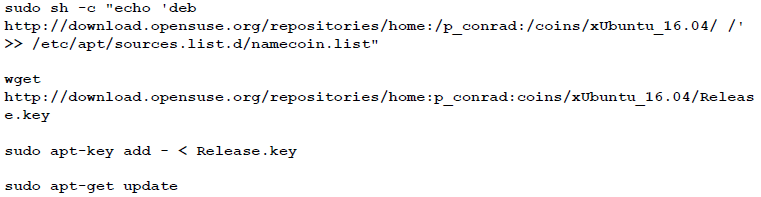
**Lab preparation**

Configure the DNSChain server in Ubuntu. . It will run the PowerDNS Recursor, issuing DNS queries for .com and .net domains as you would expect, but consulting the local Namecoin blockchain to resolve .bit domains.

We will start with a fresh copy of Ubuntu LTS. In our lab, we will deploy this Ubuntu system over Amazon's AWS Cloud.

*Some of the examples here might use expired domains, so it's best to test domain resolution on a domain that you personally registered on Namecoin's blockchain. Start with a fresh copy of ubuntu 16.04 LTS. I'm using Ubuntu 16.04 LTS on Amazon Cloud.*

We will use the following commands to prepare the infrastructure:



**Namecoin blockchain installation**

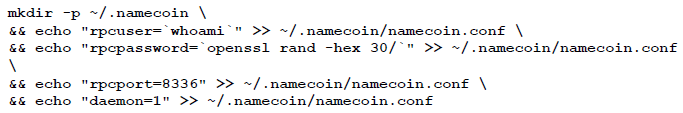
In this section, we will begin with the installation of the Namecoin blockchain.

**Installation:** We need to install the namecoin blockchain in the local system:

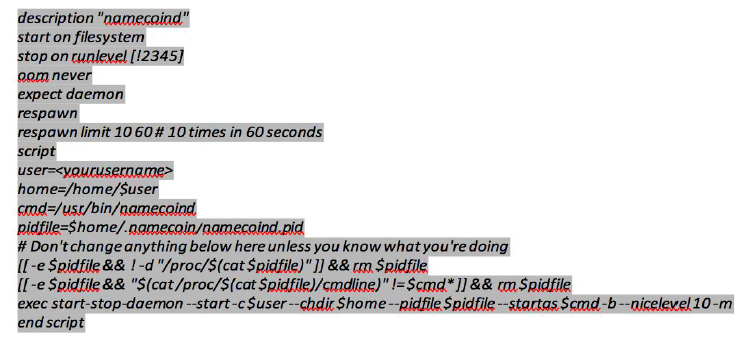
**sudo apt-get install namecoin**

To configure namecoin, follow the quick start guide. Rather than creating multiple users, this tutorial will use the current user.

**Configuration**: Once the installation is completed for Namecoin, we need to configure the blockchain with the following commands:



We will go ahead and run namecoind to get things started. Let’s check progress in downloading the blockchain using namecoind getinfo. For Ubuntu, instead of systemd, we use upstart. We need to write this file into /etc/init/namecoind.conf, remembering to substitute your username; in my case it's ubuntu:

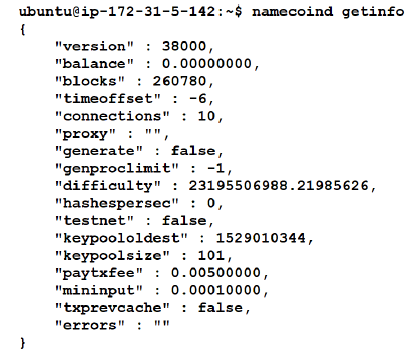


We now have to use namecoind stop to stop the process. After this, we need to issue the sudo initctl reload-configuration command, then restart using sudo shutdown -r now. Finally, namecoin gets restarted automatically.

**Verification:** As mentioned, namecoind is going to begin downloading the blockchain. We won't be able to look up domain names from the blockchain until it has made some progress. Later, when we revisit Namecoin, we can try the following command:

**namecoind getinfo**

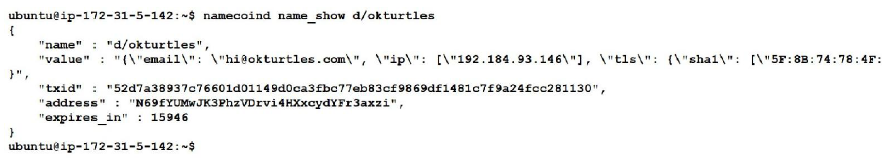
In the output, we can clearly see the details about Namecoin and details on the difficulty level, connections, timeoffset, blocks created, balance, and even any errors:



Furthermore, we will use the following command to get details over Namecoin transaction ID and address:



The following screenshot shows the output of running the preceding command:



Additionally, we can also check the RPC interface (use the rpcuser and rpcpassword from namecoin.conf):



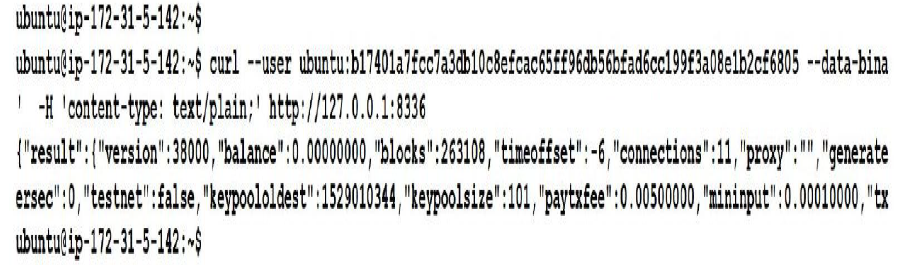
Now, we will use the curl command to get web information about content hosted over <http://127.0.0.1:8336>:

**curl --user**

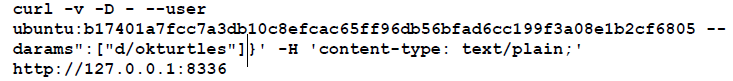
**ubuntu:b17401a7fcc7a3db10c8efcac65ff96db56bfad6cc199f3a08e1b2cf6805 –data**

**bina' -H 'content-type: text/plain;'** [**http://127.0.0.1:8336**](http://127.0.0.1:8336)

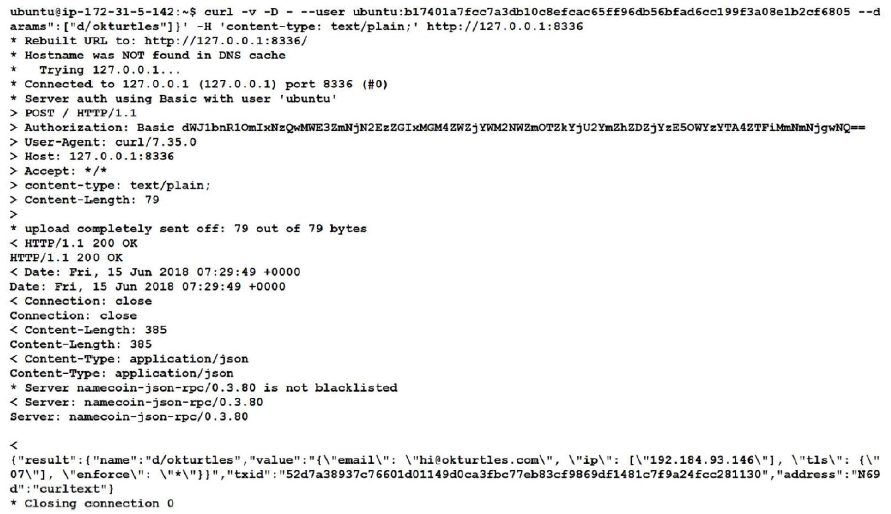
We can see the following output of the curl command against content hosted over the local system:



Additionally, we can use the curl command extract headers that an HTTP site sends to us. The following command stores the headers that an HTTP site send to us. This command writes the received protocol headers to the specified file:



The output of running the preceding command can be shown as follows:



**Installing PowerDNS**

**PowerDNS** is a premier supplier of open source DNS software, services, and support. PowerDNS is a DNS server, written in C++ and licensed under the GNU **General Public License** (**GPL**). It runs on most Unix derivatives.

It features a large number of different backends ranging from simple BIND to relational databases. We will use the following command to install PowerDNS on the system:

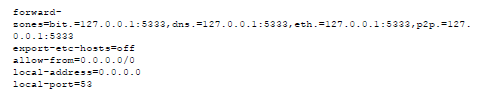
**sudo apt-get install pdns-recursor**

The output of running the preceding command can be shown as follows:



**Configuration**: We need to configure PowerDNS in our local environment. We will order PowerDNS to send requests for .bit, .eth, and .p2p domain names to port 5333.

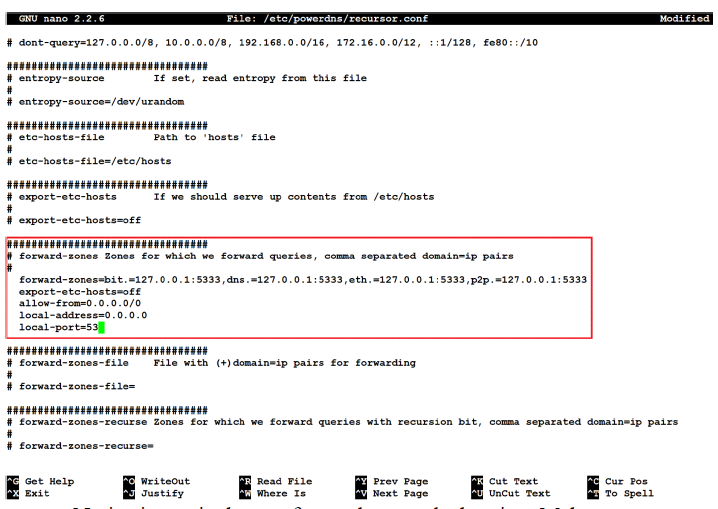
This configuration is specified in the /etc/powerdns/recursor.conf file:



In the following screenshot, we can see the extraction of queried forward zone files:



We can find the forward zone information hosted on 127.0.0.1:5333 in the following screenshot:

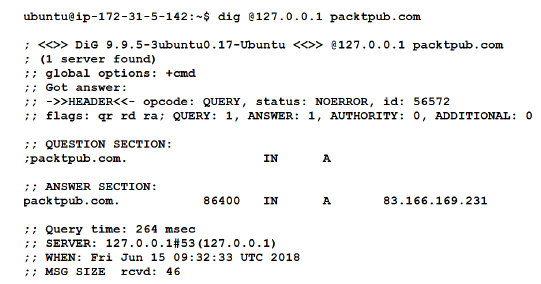


Notice in particular our forward-zones declaration. Make sure you restart PowerDNS at this point using sudo service pdns-recursor restart.Then, confirm that PowerDNS can correctly resolve conventional domain names before we move on:

**Verification:** To verify the forward zone PowerDNS installation, we need to run the following command:

**dig @127.0.0.1 packtpub.com**

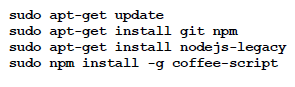
We will find the following output with an IP address found for packtpub.com:



**Installing DNSChain**

First, we will update apt-get and install some prerequisites. Note that while we install npm (that requires Node.js installation), nodejs-legacy is needed to be installed.

* **Installation of dependencies**: In this step, we will install all the dependable scripts and commands:

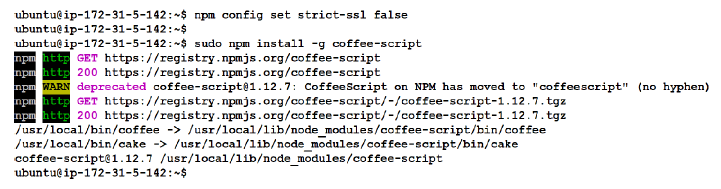


* **Installation of DNSChain:** The installation of DNSChain can now be executed with the following command:

**sudo npm install -g dnschain**

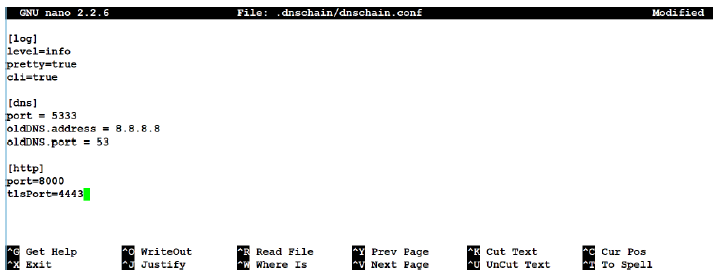
In the following output, we can find the result includes the extraction of registry file

coffee-script-1.12.7.tgz:

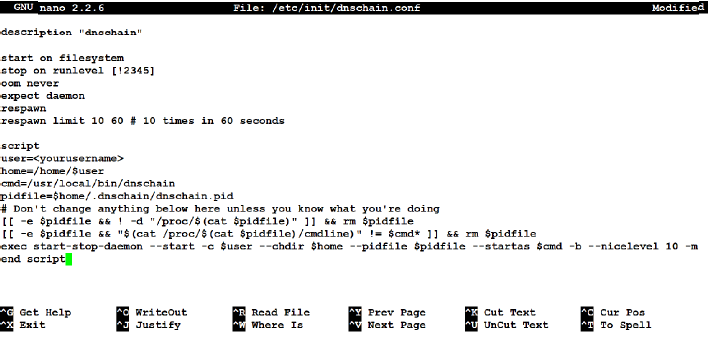


**Configuration**: We need to configure DNSChain to bind it to port 5333, but you can use any high port number as long as it matches the port number that PowerDNS is handing off requests to. This was specified earlier in /etc/powerdns/recursor.conf. Another great feature of DNSChain is that we can expose the lookup results through HTTP. We'll specify port 8000 for this, but you can use any high number port that's open.

DNSChain can be set up to be accessed by the web server, through port 8000 for example. For this example, write into /home/ubuntu/.dnschain/dnschain.conf:



We need to make another upstart file for DNSChain, and write this file into /etc/init/dnschain.conf:

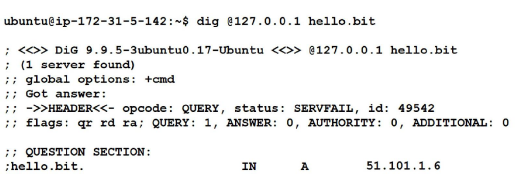


We will now run sudo initctl reload-configuration, then restart the machine. Finally, let's test it by trying to resolve a .bit domain name. *You may have to wait until a lot of the blockchain is loaded before it works.*

**Verification**: Finally, we can verify the DNSChain by performing lookup for the hello.bit address. In the output result, we can find the result with the desired IP address of server 51.101.1.6:

**dig @127.0.0.1 okturtles.bit**

The following screenshot shows the output of running the preceding command:

****

Deploying Blockchain-Based DDoS Protection

DDoS attacks

What is a DDoS attack?

How does it work?

Building up the botnet

Reconnaissance

Weaponization

Delivery

Exploitation&#xA0;

Installation&#xA0;

Command and control (C2)

Action on objectives

Types of DDoS attack

Attacks targeting network resources

User datagram protocol (UDP) flood

ICMP flood

Internet Group Management Protocol (IGMP) flood

Amplification attacks

Attacks targeting server resources

TCP SYN Flood

TCP RST attack

Secure sockets layer (SSL) based attack

Encrypted HTTP attacks

Attacks targeting application resources

DNS flooding&#xA0;

Regular expression DoS attacks&#xA0;

Hash collision DoS attacks

Challenges with current DDoS solutions

How blockchain can transform&#xA0; DDoS protection? lab

**Deploying Blockchain-Based DDoS Protection**

**DDoS attacks**

* A DDoS attack is a malicious attempt to disrupt legitimate traffic to a server by overwhelming the target with a flood of requests from geographically dispersed systems.
* During DoS attacks, the attackers bombard the target machine with a massive amount of requests that lead to the exhaustion of server resources and, as a result, it fails requests from legitimate users.
* In a DoS attack, a threat actor uses a single machine to exhaust the target server; however, a DDoS attack is much more powerful as millions of machines can be used to exhaust a target server.

**What is a DDoS attack?**

* More and more organizations are moving to the cloud with massive infrastructure to fulfil their immersive customer demands. Organizations either build their own heavy server infrastructure, or they move to cloud providers to host their servers.
* Today, attackers prefer the DDoS attack method to disrupt target services as they can generate GBs to TBs of random data to overwhelm the target, and also it becomes difficult for a target security team to identify and block each individual attacking machine, as they are millions in number.
* Furthermore, attackers never legitimately control their attacking machines, but rather they infect millions of computers worldwide with some tailored malware and then get complete access to launch a massive DDoS attack. This collection of millions of infected computers is named a **botnet** and the individual infected computers are named **bots**.
* The first instance of DDoS is a bit hard to recall exactly, but the first noticeable and significant attack occurred in 1999, and it targeted the University of Minnesota. It impacted more than 220 systems and brought down the entire infrastructure for several days.
* On Friday, October 21, 2016, the entire world witnessed one of the most complex and sophisticated DDoS attacks on Dyn (a managed DNS provider). Dyn confirmed the Mirai botnet as a primary source of malicious attack traffic. The attack opened up an important concern on internet security and threats.

**How does it work?**

To launch a DDoS attack, a threat actor can either build the entire botnet network or rent it from a dark web marketplace. Once the attacker is ready with their weapons, they need to discover vulnerable sites or hosts, or maybe an entire network.

A computer scientist at Lockheed-Martin Corporation coined a term called **cyber kill chain** that lays out the stages of a cyber attack, starting from reconnaissance to final goal of attack. These stages are:

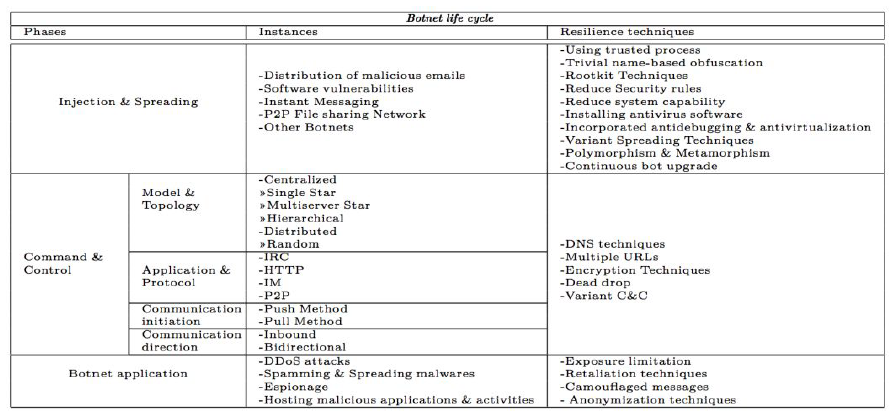
* **Reconnaissance**: The attacker identify its target device and starts searching for vulnerabilities in it.
* **Weaponization**: The attacker uses a remote tool kit and malware such as virus or worm to address the vulnerability.
* **Delivery**: The threat actor inject the cyber weapons to the victim network through several methods such as phishing email, drive-by download, USB drives, insiders and so on
* **Exploitation**: The malware code is used to trigger the attack, taking action on target network to exploit vulnerabilities
* **Installation**: Malware is now installed in the victim machine
* **Command and control**: This malware allows the remote threat actor to gain access to victim machine

In order to understand each of these stages from DDoS perspective, it is important to understand the botnet infrastructure and how it is built.

**Building up the botnet**

There are several tools, such as Jumper, Dirt, and Pandore, that eliminate the technical barrier in creating these botnets.

The following graphic outlines the botnet life cycle:



**Reconnaissance**

* The targeted system can be as large as a data center and as small as a computer. In both cases, the development of a botnet involves identifying hosts with vulnerabilities that can be exploited with some malware families.
* Attackers look for information directly or indirectly related to their target to gain unauthorized access to their protected assets. The threat actor tries all possible ways to bypass the existing security systems, such as firewalls, **intrusion prevention system** (**IPS**), web application firewalls, and endpoint protection.

**Weaponization**

The wide range of open source software has removed the technical barrier for creating malicious code. If a programmer has malicious intent and develops the code, a new breed of malware can be developed that would be difficult for security systems to detect.

The following is a list of some of the popular tools for developing DDoS:

**Low Orbit Ion Cannon (LOIC)**:

* This is one of the favourite tools, used by the popular hacktivist group *anonymous*. It is a simple flooding tool that can generate a massive volume of TCP, UDP, or HTTP traffic to overload the target server.
* It was originally developed to test the throughput of server performance; however, the anonymous group used this open source tool to launch sophisticated DDoS attacks.
* The tool was later enhanced with IRC features, which allow users to control the connected machines over IRC.

**High Orbit Ion Cannon (HOIC)**:

* A couple of years after effectively using LOIC, the anonymous group dropped it and used the HOIC tool to first target the US **Department of Justice** (**DOJ**) in response to its decision to take down website megaupload.com.
* HOIC is again a simple application built to support cross-platform basic scripts for sending HTTP POST and GET requests with an easy and simplified GUI. It was later powered with *booster* scripts, which are text files that contain additional basic code, called **booster script**.
* This booster script also allows the attacker to specify the list of target URLs to attack. HOIC is still in use by the anonymous group to launch DDoS attacks globally.

**hping**:

A tool called *hping* was developed to overcome anonymity challenges with Ion Cannon tools. It is again used to generate a massive volume of TCP traffic at the target, and it can remain anonymous by spoofing the source IP address.

**Slowloris:**

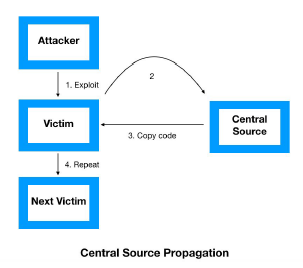
* Slowloris is one of the most advanced tools used to make attackers difficult to detect and track. This tool was developed by a gray hat hacker who is known as **RSnake** and is able to initiate DDoS for servers by creating very slow HTTP requests.
* It generates a bulk of tiny HTTP headers that target the server and make it wait for the rest of the headers to arrive.

**Delivery**

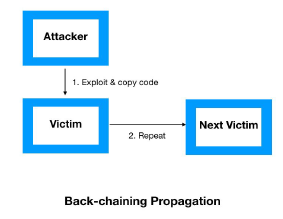
Once the malicious code is developed or software purchased from the dark web marketplace, this software can either be delivered through spear phishing emails or can also be sent through spam email campaigns. The selection of either depends on the target and also the sophistication of the operation.

We can classify the process into the following three groups of methods for propagating malicious code:

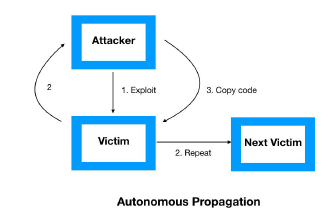
* **Central source propagation**:
* In this method, the vulnerable system that an attacker is planning to make into one more bot will be given to a central system so that the copy of the attacking system is transferred from centrally hosted infrastructure to the newly compromised system.
* After the entire toolkit is moved, a script automatically initiates a fresh attack cycle. This entire mechanism uses HTTP, FTP, and **remote procedure call** (**RPC**) protocols. In this method, threat actor exploit the victim machine, compromised system get connected to central repository of attacker and finally, central source pushes the code. Take a look at the following diagram:



* **Back-chaining propagation**:
* In this method, the attacker's toolkit is relocated to the newly compromised host by the attacker. The attacker's toolkit is specially designed to accept a file request from a compromised system. The back-channel file copy can be done by a port listener using **Trivial File Transfer Protocol** (**TFTP).**
* Unlike central source propagation method, attacker transmit both exploit and code together into the victim machine:



* **Autonomous propagation**:
* In this mechanism, the moment an attacker breaks into a system, their toolkit is transferred to the compromised host. This mechanism differs in terms of method of transfer, as attack toolkits are first planted into the compromised host by the attackers only.
* In this method, attacker transmit the exploit first and then the code from himself but not from any central repository. Take a look at the following diagram:



**Exploitation**

Once the malware is delivered to the network, it will initiate the process of exploiting unpatched software vulnerabilities, weak software coding practices, and lack of user attention. Usually, there are numerous vulnerabilities present in the network; however, the availability of exploits makes the vulnerability much more critical in nature.

**Installation**

* In the installation stage, the malware is installed in the targeted system and allows the remote attacker to gain access to it. During the installation process, the malware may be installed in the user space or kernel space of a system.
* Malware installed in the user space has a high possibility of detection; however, malware installed in the kernel space has a low chance of being detected by security systems, such as endpoint protection, endpoint detection, and response platforms.

**Command and control (C2)**

* After the weapon has been successfully installed, the target is now completely under the control of a remote central system, named the system.
* The network of compromised devices are called botnet, completely under control with threat actor; however, the botnet remains silent until it get activated from attacker. There are even several types of encrypted bot-to-bot communication present over public peer-to-peer networks.

**Action on objectives**

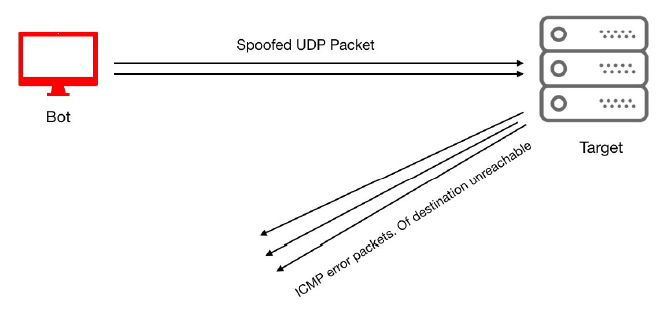
Once the C2 channel has been established, the attacker can launch the DDoS attack on the target. At this stage, the attacker runs the script to activate all the bots in the entire botnet. The attacker also configures the botnet regarding what type of traffic needs to be generated.

**Types of DDoS attack**

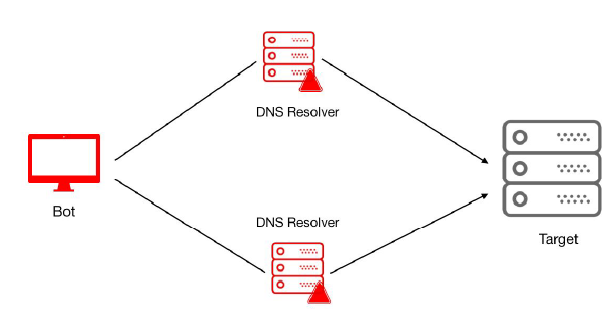
DDoS attacks are carried out in several ways. However, attackers select one of them based on different factors, such as target difficulty, financial capability, anonymity, priority, and other factors. It does not take much technical expertise to run the DDoS attack program and launch it. There are mainly three types of attack, categorized as follows:

1. Attacks targeting network resources
2. Attacks targeting server resources
3. Attacks targeting application resources
4. **Attacks targeting network resources**

* These are attack campaigns in which it is planned to consume the network resources of the target system. In this attack, network bandwidth gets completely consumed by flooding. The following are several types of flooding attacks.
* **User datagram protocol (UDP) flood :**
* UDP is a protocol embedded in the IP packet for socket-level communication between two devices.
* A UDP flood attack does not exploit any specific vulnerability of the target system, but rather it simply disrupts the normal traffic of the target system by overwhelming it with a high level of flooding.
* It points to random ports on the target server and consumes all the traffic bandwidth for the target system.
* This UDP flood doesn't even allow the system to send **Internet Control Message Protocol** (**ICMP**) destination unreachable packets. Usually, this kind of attack is considered in the class of a small-to-medium-level flood attack and measured in Mbps and PPS, as shown in the following diagram:



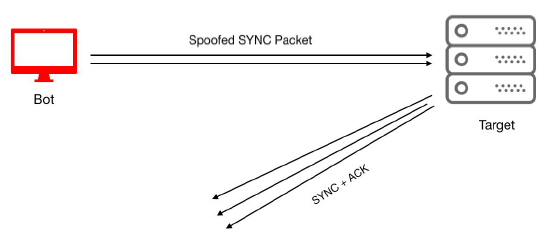
* **ICMP flood**
* ICMP is another connectionless protocol used for IP-level reachability and management operations. Again it doesn't rely on any vulnerabilities to work.
* An ICMP flood can be performed with any type of ICMP message, such as echo requests and echo replies. Being one of the oldest flooding techniques, organizations have practices to deploy control-plane policies over network devices to restrict the amount of ICMP packets that can processed by the control planes of devices.
* **Internet Group Management Protocol (IGMP) flood**
* IGMP is a multicast protocol, connectionless in nature. It is non vulnerability based, involving the sending of a large amount of IGMP message reports to networks or routers.
* **Amplification attacks**
* An amplification attack takes the opportunity of a disparity between a request and a reply in a communication channel. An attacker can compromise a router and force the router to send broadcast messages on multicast addresses by spoofing the source address.
* It can even be used with DNS amplification, in which the attacker can compromise a recursive DNS name server to cache large files. Take a look at the following diagram:



1. **Attacks targeting server resources**

Attacks that target server resources of the victim and exhaust the entire server processing and memory eventually cause disruption for legitimate traffic. In this category, attackers identify the vulnerabilities of the target server and weaponized the malware to exploit those vulnerabilities.

* **TCP SYN Flood**
* This attack makes use of the TCP three-way handshake mechanism and consumes most server resources with TCP sync messages.
* In the TCP three-way handshake, a client first sends the TCP packet with the sync flag set that requests a server to allocate a resource and establish a communication channel. In a TCP SYN attack, attacking systems send a series of TCP requests with TCP flags set to SYN.
* To manage each of these requests, the server has to open and allocate certain CPU resources, and also buffer to prepare further communication. Now, the server sends a TCP message with a flag set to SYN-ACK, and expects the client to acknowledge that with a TCP message with the ACK flag.
* The attacking systems receive that but never respond, and as a result, the server keeps the socket open and resources allocated for the same client machines. Server resources are limited, but the attackers can keep multiplying the request to the server to finally exhaust the server and make it unavailable for legitimate user traffic.
* TCP has a specific timeout for the request and response process, but the attacker gains the advantage of the same period to send massive TCP requests. Take a look at the following diagram:



* **TCP RST attack**
* In the TCP/IP stack, **Reset** (**RST**) flag in TCP is used to notify a server to reset its ongoing TCP connection. In a TCP RST attack, the attacker intercepts an active TCP connection between the client and the server by trying a random sequence of numbers.
* After successfully identifying the sequence of numbers, the attacker then spoofs the TCP RST message to the client's source IP address. For humans to perform such an activity, this would be very difficult. Hence, bots are used to intercept and identify the active sequence number.
* **Secure sockets layer (SSL) based attack**
* SSL is standard security protocol for establishing encrypted channels between a web server and a browser. This ensures that all transmitted data is encrypted between web server and browser, and hence provides a better privacy and integrity solution for internet users.
* SSL runs over TCP/IP and sends the SSL *hello* only once the TCP three-way handshake is completed. SSL-based DDoS attacks can be performed in a variety of ways, such as targeting the SSL handshake mechanism, sending random and garbage data to the SSL server, or exploiting certain function-related SSL encryption key mechanisms.
* **Encrypted HTTP attacks**
* With the growing use of SSL/TLS-encrypted web applications, attackers are also moving toward encrypted HTTP-based attacks. Most organizations don't have a security solution that can inspect SSL traffic and hence fail to protect it from malicious traffic.
* Attackers make use of this weakness and adopt more and more capabilities to compromise networks through encrypted HTTP.

1. **Attacks targeting application resources**

DDoS attacks are on the rise; threat actors are moving from traditional methods to more advanced and sophisticated application-based attacks. These are not just limited to HTTP-based attacks but are even adapting to HTTPS, DNS, FTP, SMTP, and VOIP.

Applications are built with several independent components and hence are vulnerable. Therefore, application based attacks become more attractive for threat actors. We will cover some of the most widely used attacks

* **DNS flooding**
* DNS is used everywhere, and every organization network has to have the DNS port open for name resolution. It is easy to launch DNS-based flooding and also difficult for the security system to detect it.
* DNS uses the UDP protocol for faster request and response times, without establishing a new connection (like in the TCP handshake). In this kind of attack, the DNS server can be overwhelmed with a massive amount of DNS requests, making the victim server unable to process legitimate requests.
* This technique was used in the recent Mirai attack on the Dyn network that left users unable to access YouTube, Twitter, Netflix, and several other applications.
* **Regular expression DoS attacks**
* These use the *low and slow* methodology to attack the victim server. The attacker leverages vulnerabilities in the library files deployed in the server.
* Whenever a client sends a request with regular expressions, a server has to spend a large amount of resources to process the regular expression. Attackers use this to exploit the server and send regular expressions periodically that security systems fail to detect.
* **Hash collision DoS attacks**
* With this kind of attack, makes attackers spend days to months identifying vulnerabilities in the web application frameworks. Hash tables are used to index POST sessions in most of the application servers.
* The server has to manage hash collisions when similar hash values are returned. Collision resolutions consume a lot of processing resources as the attacker keeps sending POST messages with a multitude of parameters.
* Attackers build the parameters in such a way that they cause hash collisions on the server side and as a result keep the server busy processing them.

**Challenges with current DDoS solutions**

On February 28, 2018, GitHub, the code hosting website, was hit with the largest-ever DDoS attack, recorded at 1.35 Tbps. As DDoS attack falls under cyber threat category that makes it unfeasible to deploy any security prevention mechanism as system vulnerabilities are under control of organizations but threats can’t be controlled. The frontend of web application remains centralized for all users; hence, it leaves a single point of failure situation for organizations.

**How blockchain can transform DDoS protection?**

* By definition, blockchain is a decentralized network that allows independent parties to communicate without any third party involvement.
* In order to protect networks from DDoS attacks, organizations can be made distributed between multiple server nodes that provide high resilience and remove the single point of failure.

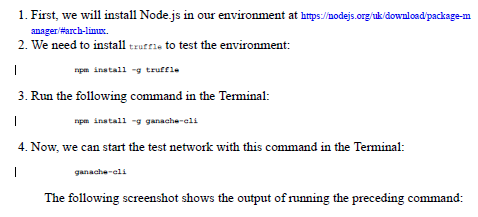
There are two main advantages to using blockchain, as follows:

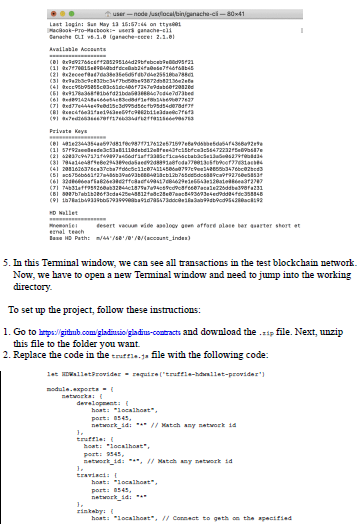
* Blockchain technology can be used to deploy a decentralized ledger to store blacklisted IPs
* Blockchain technology eliminates the risk of a single point of failure

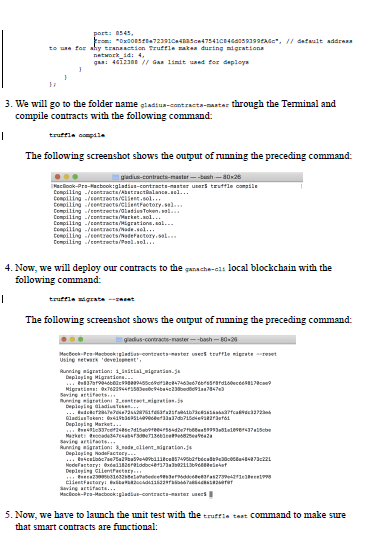
LAB

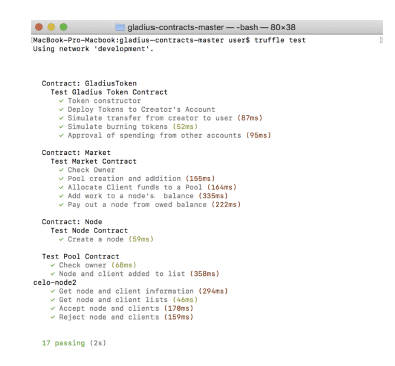
In order to deploy the blockchain based DDoS protection platform, we must prepare the test environment with Node.js and Truffle with ethereum blockchain. We will be using an existing blockchain project to defend network from DDoS attack. The project link can be found at link https://github.com/gladiusio/gladius-contracts.

Furthermore, we need to follow the steps to prepare the infrastructure for Gladius project:









6. Go to https://github.com/gladiusio/gladius-control-daemon, download the .zip, and unzip it to the same folder as gladius-contracts.

7. Next, we locate the gladius-control-daemon-master folder in the Terminal and link contract **Application Binary Interface** (**ABI**). ABI is the interface between two program modules, one of which is at the level of machine code:

**ln -s ../gladius-contracts-master/build build**

The following screenshot shows the output of running the preceding command:



8. Next, we will install dependencies with the npm install command:



9. Next, we will start the script with the node index.js command:



