Distributed DHT & Meta tag web architecture

Current web 1.0 architecture

- WEB 1.0 is client server architecture
  - Thin client just visualize the data
  - Web server serving files representing the data.
    - Web server may be smart, modifying the data, depending on input or other (for example extended with CGI, PHP, Java frameworks, etc)
    - Only one web server serves the client. Multiple clients can be connected to the server. It is pure thin client – server architecture

Current web 2.0 architecture

- One step above the Web 1.0 architecture
- The same client-server model, with more smarter clients that offload part of the work usually done by the server
- Client usually run Flash applets or JavaScript/HTML5
- Web 2.0 makes the experience more interactive, should (but it not necessary) decrease the load of the servers, improve the user experience
- Still one server serves multiple clients and we have classical client server architecture. The security model expects all client requests to go to the same domain. The server is expected to be smart and have advanced logic for advanced applications

Future web 3.0 architecture

- It is an enhancements over Web 2.0 where we will have more smarter clients in the browsers, receiving data from multiple servers and sources (this require modification to the current html/ javascript security model)
- For example a client may use google docs for form editing but to post the form on facebook in a smoothly integrated way
- The model is distributed – a single client may work with multiple servers. The servers may be smart, but not as smarter as in web 1.0/2.0

Scalability

- The scalability is related to security, as most of the current DoS attacks are designated to affect the capabilities of a server to serve its content
- More distributed approach (web 3.0 vs web 2.0 vs web 1.0) improves the performance, scalability and security, but may affect the stability

BalkanLeaks web client

- Web 2.0 application based on HTML5 standards
- Extremely dummy server – it only store and serve the files, has no keys and no logic. Can be implemented with CDN without any logic in the CDN
- All the search, decryption is performed in the client
- It can work in online and offline mode. In offline mode, all the data is preloaded in the browser and everything operates locally. In online mode the data is transferred on demand and stored in the client. It can work with thinner clients (smartphones and tablets), preserving the same level of security with the exception that it rely on a server to serve the new data and this is a risky point (DoS attacks to the server may prevent clients to access data. Clients are also detectable by network sniffing, for some that is monitoring activity to the server)

The new idea for distributed infrastructure for Wikileaks

- How to implement infrastructure that require no servers?
- How to implement infrastructure that can be highly distributed?
- How to implement infrastructure dependable mainly by its users?
- Can this infrastructure have also acceptable security?

Web 3.1

- All the current web models (1.0, 2.0, 3.0) depends on a single or limited amount of servers. The client connects to them and retrieve data. The servers may be distributed by CDN, but at the end the infrastructure depends on servers, dns protocol, etc
- Lets imagine that we use the network differently – we use the bittorrent DHT instead of servers but everything else is the same?

Magnet tag as url

- Imagine, that we replace the HTTP protocol in the browser uri, with Magnet tag
  - Meta tag allow us to use torrent DHT to serve file archives, without need of having torrent tracker
- You can enter url in the following format:
  - magnet:?xt=urn:btih: 956b0722aac4de545d1bae42d7e811b9f2fdeafd/filepath
  - Your browser may have integrated DHT torrent client and to download the file archive stored in the DHT network, identified by the hash id, and then extract from the inside the archive the specified file, and then to visualize it

Magnet as URL - advantages

- Simple
- Conceptual
- The DHT network serves the files – no need of any servers or infrastructure
- No need of DNS and domains
- Minimize the single point of failure

What the browser will do?

- The user enters magnet url
- The browser downloads the referenced archive from the DHT network
- The browser extract the referenced file from the archive
- The browser visualize the file content
- The downloaded torrent archive will be seeded as long it is in the browser cache

Follow the traditions

- If we compare it to the traditional http, we can assume that the DHT will represent TCP and web/file server, the DHT hash in the magnet will be equivalent to the server hostname, and the file path part will be the file path part
- The traditional html and javascript will work as is and with the help of HTML5 (see the balkanleaks client example)
- No need of server infrastructure. A single file will be alive as long as there is even single user that has it in the browser cache
- No need for DNS and domains, so no domain filters

How can we protect the integrity of the data?

- The DHT present transport mechanism
- The magnet present addressing scheme
- But they do not enforce any security. How can we be sure that the data is originated by the right source?
- HTTP uses SSL/TLS for it, but this can not work for DHT as DH algorithm can not be implemented without active server
- However, we can use PGP for this, having keys as certificates the same way as in SSL
- We can use the certificates to trust the sources of the data, and to ensure they are the right one

Advances

- Can be used for any web data not only for Wikileaks
- Can be the next step of the web 3.1 – fully distributed system, no servers, easy to scale, dependable mainly on the users, hard to filter (with torrent encryption enabled), having encryption both to the file archive, or to point to point links for download

What should be done?

- Browser should have support for torrent protocol with DHT and Magnet tag
- Browser should support gpg
- Browser should have the logic integrated
- Modification of the magnet url, to follow the format of the traditional uri (backslash)

Security issues

- DHT is not resilient to smart DoS attacks
- Fake DHT clients may “filter” files and hashes
- Some can publish different file with the same hash – no protection against it, only the integrity verification can prevent spoofing and fake files, the risks stands
- However the DHT advances fast and the community may fix all the issues