ONIONS:
TOR-POWERED DISTRIBUTED DNS
FOR ANONYMOUS SERVERS

by

Jesse Victors

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Ming Li Major Professor	Dr. Nicholas Flann Committee Member
Dr. Daniel Watson Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2015

CONTENTS

LIST OF FIGURES

CHAPTER 1
PROBLEM STATEMENT

1.1 Assumptions and Threat Model

OnioNS' basic design and protocols rely on several assumptions and expected threat vectors.

1. Let Alice is a Tor user and let Bob be a recipient. If Alice constructs a three-router Tor circuit via NTor and sends a message m to Bob, we assume that the Tor circuit preserves Alice's privacy and provides her with anonymity from Bob's perspective. Namely, we assume that out of Alice's identity, location, and knowledge of Bob, an outside attacker Eve can obtain at most one, and that Bob can know no more about Alice than the contents of m. The exception to this is if Alice chooses to disclose her identity only to Bob, but then Bob does not know Alice's location. We assume that neither Bob nor Eve can force Alice into breaking her privacy involuntarily. The protection of a Tor circuit relies on Tor's assumption that Eve only has access to some Tor traffic; no defence is made by Tor nor by OnioNS to defend against a global adversary.

2. Adversaries cannot break standard cryptographic primitives.

   (a) We assume that they cannot efficiently break AES ciphers, SHA-384 hashes, RSA-1024, Curve25519 ECDHE, Ed25519 digital signatures, or have hardware-level attacks against the scrypt key derivation function.

   (b) We assume that they maintain no backdoors or other software breaks in the Botan or the OpenSSL implementations of the primitives.

   (c) We assume that they are not capable of breaking cryptographic protocols built on the primitives such as TAP and NTor.

3. We assume that not all Tor nodes are honest. We assume that at least some Tor nodes are run by malicious operators, curious researchers, experimenting developers, or government organizations. They may also be wiretapped, exploited, broken into, or become unavailable. This assumption is also made by Tor's developers and therefore we must conclude that any new functionality added to existing Tor nodes must also fall under their assumption. However, we assume that the majority of Tor nodes are honest and reliable. We consider this reasonable because it is a prerequisite for the security of Tor circuits.

4. We assume that the percentage of dishonest routers within the Tor network does not increase in response to the inclusion of OnionNS into Tor infrastructure. This assumption simplifies our threat model analysis but we consider it realistic because Tor traffic is purposely kept secret and is therefore much more valuable to Eve. OnioNS uses public data structures so we don't consider the inclusion of OnioNS a motivating factor to Eve. Moreover, we assume that Eve cannot predict in advance the identities of the Tor routers that act as authorities for OnioNS.

5. Let C be the set of Tor nodes that have the Fast and Stable flags and let Q be an M-sized set chosen randomly from C. Q may be under the influence of one or more adversaries who may cause subsets of Q to collude, but our final assumption is that the largest subset of Q is acting honestly according to specifications.

1.2 Design Objectives

Tor's high security environment introduces a distinct set of challenges that must be met by additional functionality. Privacy and anonymity are of paramount importance. Here we enumerate a list of requirements for any security-enhanced DNS applicable to Tor hidden services. In Chapter we analyse several existing prominent naming systems and show how these systems do not meet these requirements. In Chapters and we demonstrate how we overcome them with OnioNS.

1. **The system must support anonymous registrations.** It must be hard for any party to infer the identity or location of the registrant, unless the registrant chooses to disclose that information. Hidden services are privacy-enhanced servers and the registration of a domain name should not compromise that privacy.

2. **The system must support privacy-enhanced lookups.** The resolver should not be able to identify a client nor track their lookups. In other words, clients should ideally have anonymity and be indistinguishable from other clients from the perspective of a resolver.

3. **Clients must be able to authenticate registrations.** The users of the system must be able to verify that the information they received from the service has not been forged and is authenticate relative to the authenticity of the server they will connect to.

4. **Domain names must be provably or have a near-certain chance of being unique.** Any domain name of global scope must point to at most one server. In the case of naming systems that generate names via cryptographic hashes, the domain name key-space must be of sufficient length to be remain resistant to at least collision and second pre-image attacks.

5. **The system must be distributed.** Systems with root authorities have distinct disadvantages compared to distributed networks: specifically, central authorities have absolute control over the system and root security breaches could easily compromise the integrity of the entire system. Root authorities may also be able to easily compromise user privacy or may not allow anonymous registrations, although this is system dependent. For these reasons, naming systems with central authorities can safely be considered dangerous and ill-suited for hidden services.

6. **The system must be simple and relatively easy to use.** It should be assumed that users are not security experts or have technical backgrounds. Tor's developers go to great lengths to ensure that Tor tools are straightforward, and the success of Tor's low-latency onion routing model demonstrates the effectiveness of convenience within high-security settings. Any naming system used by Tor must be equivalently simple, resolve protocols with minimal input from the user, and hide non-essential details.

7. **The system must be backwards compatible with existing protocols.** Just as the Clearnet's DNS did not introduce any changes in the TCP/IP layer, naming systems for Tor must preserve the original Tor hidden service protocol. DNS is optional, not required.

Additionally, we specify several performance-enhancing objectives, although these are not requirements.

1. **The system should not require clients to download the entire database.** It can safely assumed that in most realistic environments clients have neither the network capacity nor the storage to hold the system's database. While they may choose to download the database, it should not be required for normal functionality or to meet any of the above objectives.

2. **The system should not introduce significant burdens to the clients.** Despite the rapid and exponential growth of consumer-grade hardware, in most environments not every client is on a high-end machine. Therefore the system should not burden user computers with significant computation or memory demands.

3. **The system should have low latency** and resolve lookup queries within a reasonable amount of time.

CHAPTER 2
ANALYSIS

2.1 Security

Now we examine and compare OnioNS's central protocols against our security assumptions and expected threat model.

2.1.1 Hidden Service Protocols

Record Generation

Tor hidden services are identified by their public keys, and assuming a hidden service Bob does not provide a PGP key in his Record, we leak no identifiable information

Although this approach does not entirely thwart Sybil attack, this attack vector is difficulty to impossible to counter in a privacy-enhanced environment, and trading anonymity for defence is highly undesirable.

Let Bob create the Record r.

The proof-of-work scheme is carefully designed to limit Eve to the same capabilities as legitimate users, thus significantly deterring this attack. The use of scrypt makes custom hardware and massively-parallel computation expensive, and the digital signature in every record forces the hidden service operator to resign the fields for every iteration in the proof-of-work. While the scheme would not entirely prevent the operator from outsourcing the computation to a cloud service or to a secondary offline resource, the other machine would need the hidden service private key to regenerate *recordSig*, which the operator can't reveal without compromising his security. However, the secondary resource could perform the scrypt computations in batch without generating *recordSig*, but it would always perform more than the necessary amount of computation because it would could not generate the

SHA-384 hash and thus know when to stop. Furthermore, offloading the computation would still incur a cost to the hidden service operator, who would have to pay another party for the consumed computational resources. Thus the scheme always requires some cost when claiming a domain name.

Record Broadcast

Tor circuit preserves privacy.

2.1.2 OnioNS Server Protocols

Page Selection

Synchronization

Quorum Qualification

The quorum nodes hold the greatest amount of responsibility and control over OnionNS out of all participating nodes in the Tor network, therefore ensuring their security and limiting their attack capabilities is of primary importance.

In section 1.1, we assumed that an attacker, Eve, already controls a fraction of Tor routers. For a dynamic network size, a fixed fraction of dishonest nodes, and a fixed quorum size, every time a quorum is selected there is a probability that more than half of the quorum is dishonest and is colluding together. If this occurs, records may be dropped rather than archived. *Here we explore the optimal quorum rotation rate, given the balance between high overhead and high security risk if the quorum is rotated quickly, and long-term implications and possible stability issues associated with very slow rotation.*

If Eve controls some Tor nodes (who may be assumed to be colluding with one another), the attacker may desire to include their nodes in the quorum for malicious manipulation, passive observation, or for other purposes. Alternatively, Eve may wish to exclude certain legitimate nodes from inclusion in the quorum. In order to carry out either of these attacks, Eve must have the list of qualified Tor nodes scrambled in such a way that the output is pleasing to Eve. Specifically, the scrambled list must contain at least some of Eve's malicious nodes for the first attack, or exclude the legitimate target nodes for the second attack. We initialize Mersenne Twister with a 384-bit seed, thus Eve can find k seeds that generates a desirable scrambled list in 2^{192} operations on average, or 2^{384} operations in the worst case. The chance of any of those seeds being selected, and thus Eve successfully carrying out the attack, is thus $\frac{2^{384}}{k}$.

Eve may attempt to manipulate the consensus document in such a way that the SHA-384 hash is one of these k seeds. Eve may instruct her Tor nodes to upload a custom status report to the authority nodes in an attempt to maliciously manipulate the contents of the consensus document, but SHA-384's strong preimage resistance and the unknown state and number of Tor nodes outside Eve's control makes this attack infeasible. The time to break preimage resistance of full SHA-384 is still 2^{384} operations. This also implies that Eve cannot determine in advance the next consensus document, so the new quorum cannot be predicted. If Eve has compromised at least some of the Tor authority nodes she has significantly more power in manipulating the consensus document for her own purposes, but this attack vector can also break the Tor network as a whole and is thus outside the scope of our analysis. Therefore, the computation required to maliciously generate the quorum puts this attack vector outside the reach of computationally-bound adversaries.

OnionNS and the Tor network as a whole are both susceptible to Sybil attacks, though these attacks are made significantly more challenging by the slow building of trust in the Tor network. Eve may attempt to introduce large numbers of nodes under her control in an attempt to increase her chances of at least one of the becoming members of the *quorum*. Sybil attacks are not unknown to Tor; in December 2014 the black hat hacking group LizardSquad launched 3000 nodes in the Google Cloud in an attempt to intercept the majority of Tor traffic. However, as Tor authority nodes grant consensus weight to new Tor nodes very slowly, despite controlling a third of all Tor nodes, these 3,000 nodes moved 0.2743 percent of Tor traffic before they were banned from the Tor network. The Stable and Fast flags are also granted after weeks of uptime and a history of reliability. As nodes must have these flags to be qualified as a *quorum candidate*, these large-scale Sybil attacks are financially demanding and time-consuming for Eve.

Flooding

2.1.3 Tor Clients

Quorum Derivation

Domain Query

Tor circuit preserves privacy.

OnionNS records are self-signed and include the hidden service's public key, so anyone — particularly the client — can confirm the authenticity (relative to the authenticity of the public key) and integrity of any record. This does not entirely prevent Sybil attacks, but this is a very hard problem to address in a distributed environment without the confirmation from a central authority. However, the proof-of-work component makes record spoofing a costly endeavour, but it is not impossible to a well-resourced attacker with sufficient access to high-end general-purpose hardware.

Hidden service .onion addresses will continue to have an extremely high chance of being securely unique as long the key-space is sufficiently large to avoid hash collisions.

As we have stated earlier, falsely claiming a negative on the existence of a record is a problem overlooked in other domain name systems. One of the primary challenges with this approach is that the space of possible names so vast that attempting to enumerate and digitally sign all names that are not taken is highly impractical. Without a solution, this weakness can degenerate into a denial-of-service attack if the DNS resolver is malicious towards the client. Our counter-measure is the highly compact hashtable bitset with a Merkle tree for collisions. We set the size of the hashtable such that the number of collisions is statistically very small, allowing an efficient lookup in $\mathcal{O}(1)$ time on average with minimal data transferred to the client.

Onion Query

2.2 Design Objectives

OnioNS achieves all of our original requirements:

1. **The system must support anonymous registrations** — OnioNS Records do not contain any personal or location information. The PGP key field is optional and may be provided if the hidden service operator wishes to allow others to contact him. However, the operator may be using an email address and a Web of Trust disassociated from his real identity, in which case no identifiable information is exposed.

2. **The system must support privacy-enhanced lookups** — OnioNS performs Domain and Onion Queries through Tor circuits, and under our original assumption that circuits provide strong guarentees of client privacy and anonymity, resolvers cannot sufficiently distinguish users to track their lookups.

3. **Clients must be able to authenticate registrations** — OnioNS Records are self-signed, enabling Tor clients to verify the digital signature on the domain names and check the public key against the server's key during the hidden service protocol. This ensures that the association has not been modified in transit and that the domain name is authentic relative to the authenticity of the destination server.

4. **Domain names must be provably or have a near-certain chance of being unique** — Tor hidden services .onion addresses are cryptographically generated with a key-space of $58^{16} \approx 2^{93.727695922}$ and domain names within OnioNS are provably unique by anyone holding a complete copy of the Pagechain.

5. **The system must be distributed** — The responsibilities of OnioNS are spread out across many nodes in the Tor network, decreasing the load and attack potential

for any single node. The Pagechain is likewise distributed and locally-checked by all Mirrors, and although its head is managed by the Quorum, these authoritative nodes have temporary lifetimes and are randomly selected. Moreover, Quorum nodes do not answer queries, so they have limited power.

6. **The system must be simple and relatively easy to use** — Domain Queries are automatically resolved and require no input by the user. From the user's perspective, they are taken directly from a meaningful domain name to a hidden service. Users no longer have to use unwieldy .onion addresses or review third-party directories, OnioNS introduces memorability to hidden service domains.

7. **The system must be backwards compatible with existing protocols** — OnioNS does not require any changes to the hidden service protocol and existing .onion addresses remain fully functional. Our only significant change to Tor's infrastructure is the mechanism for distributing hashes for the Quorum Qualification protocol, but our initial technique for using the Contact field minimizes any impact. We also hook into Tor's TLD checks, but this change is very minor. Our reference implementation is provided as a software package separate from Tor per the Unix convention.

Finally, we meet our optional performance objectives:

1. **The system should not require clients to download the entire database** — Only Mirrors hold the Pagechain, and clients do not need to obtain it themselves to issue a Domain Query. Therefore clients rely on their existing and well-established trust of Tor routers when resolving domain names. However, clients may optionally obtain the Pagechain and post Domain Queries to localhost for greater privacy and security guarantees.

2. **The system should not introduce significant burdens to the clients** — Record verification should occur in sub-second constant time in most environments, and Ed25519 achieves very fast signature confirmation so verifying Page signatures at level 1+ takes trivial time. However, clients also verify the Record's proof-of-work,

so for some scrypt parameters the client may spend non-trivial CPU time and RAM usage confirming the one scrypt iteration required to check. We must therefore choose our parameters carefully to reduce this burden especially on low-end hardware.

3. **The system should have low latency** — Domain Queries without any packet delays over Tor low-latency circuits. Its exact performance is largely dependent on circuit speed and the client's verification speed.

We therefore believe that we have squared Zooko's Triangle; OnioNS is distributed, enables hidden service operators to select human-meaningful domain names, and domain names are guaranteed unique by all participants.

2.3 Performance

bandwidth, CPU, RAM, latency for clients to be determined...

demand on participating nodes to be determined...

Unlike Namecoin, OnionNS' *page*-chain is of L days in maximal length. This serves two purposes:

1. Causes domain names to expire, which reduced the threat of name squatting.

2. Prevents the data structure from growing to an unmanageable size.

2.3.1 Proof-of-work

2.3.2 Broadcast

2.3.3 Flood

2.3.4 Query

2.4 Reliability

Tor nodes have no reliability guarantee and may disappear from the network momentarily or permanently at any time. Old *quorums* may disappear from the network without consequence of data loss, as their data is cloned by current *mirrors*. So long as the *quorum* nodes remain up for the Δi days that they are active, the system will suffer no loss of functionality. Nodes that become temporarily unavailable will have out-of-sync *pages* and will have to fetch recent records from other *quorum* nodes in the time of their absence.

CHAPTER 3
FUTURE WORK

We introduce no changes to Tor's hidden service protocol and also note that the existence of a DNS system introduces forward-compatibility: developers can replace hash functions and PKI in the hidden service protocol without disrupting its users, so long as records are transferred and OnionNS is updated to support the new public keys.

web of trust for .onion keys to thwart Sybil attack

Some open problems that I need to address include:

1. How frequently should domains expire? Are there any security risk in sending a Renew request?
2. How should unreachable or temporarily down nodes be handled? I'd like to know the percentage of the Tor network that is reachable at any given time.
3. How many nodes in the Tor network should be assumed to be actively malicious? What are the implications of increasing this percentage?
4. What attack vectors are there from the committee nodes, and how can I thwart the attacks?
5. What are the implications of a node intentionally voting the opposite way, or ignoring the request altogether?
6. What would happen if a node was too slow or did not have enough storage space to do its job properly?
7. What other open problems are there?
8. What related works are there in the literature that relate to the concepts I have created here?

REFERENCES