Balancing the Shadows

Max Schuchard, Alexander W. Dean, Victor Heorhiadi, Nicholas Hopper, Yongdae Kim
University of Minnesota
Minneapolis, MN
{schuch, adean, victor, hopper, kyd } @ cs.umn.edu

ABSTRACT

In this paper, we examine the ShadowWalker peer-to-peer anonymity scheme. ShadowWalker attempts to provide anonymity via circuits built using random walks over a secured topology. ShadowWalker's topology is secured through the use of *shadows*, peers that certify another node's routing information. We demonstrate two flaws in ShadowWalker. First, an attacker can compromise the underlying topology of ShadowWalker as a result of an insufficient numbers of shadows. We show that the failure of the underlying topology directly results in the failure of ShadowWalker to provide anonymity guarantees. Second, the dependence on untrusted nodes to certify other nodes allows an attacker to launch a selective denial of service attack. We show that there is an inherent tension between protecting against these two attacks: weakening the first attack strengthens the second attack and vice versa. We introduce a mechanism that generalizes ShadowWalker's lookup defense, and show that this mechanism can be tuned to simultaneously provide strong protection against both these attacks. Last, we implement ShadowWalker and provide performance measurements from a prototype deployment on PlanetLab.

Categories and Subject Descriptors: C.2.0 COMPUTER COMMUNICATION NETWORKS: Security and protection

General Terms: Security

Keywords: anonymity, peer-to-peer, eclipse attack, selective denial of service, ShadowWalker

1. INTRODUCTION

Anonymity systems are a critical tool for those seeking online privacy. They are used to avoid surveillance, preserve freedom of speech, and aid censorship resistance. Over the past decade usage of these systems has become far more widespread: for example, Tor [5], the most popular and well known low-latency anonymity system, is estimated to have roughly 100,000 simultaneous users. This number will likely continue to rise as more and more Internet users become increasingly aware of online privacy issues. It is easily within the realm of possibility that the number of Tor users could rise to that of other commonly used peer-to-peer systems, which support millions of simultaneous users. Should this happen, at the current ratio of clients to relays, nearly all of the bandwidth in the system would be spent distributing network state. This is a result of Tor's requirement that each node must know of all relays.

One possible solution to this lack of scalability is the use of a distributed peer-to-peer network structure. Most existing peer-to-peer anonymous communication systems open up new attack surfaces [1, 4, 11, 20] that do not exist in more centralized designs while remaining vulnerable to many of the same attacks that plague their centralized counterparts. In response to these attacks, researchers have attempted to create a fully decentralized system that is not vulnerable to these attacks.

One of the products of this effort is ShadowWalker [12], a peer-to-peer anonymous communication system based on random walks over a secure, structured topology. ShadowWalker secures its topology by having each node's state mirrored by its surrounding nodes, called *shadows*. Any response to a lookup is accompanied by digital signatures from all of a node's shadows (its *neighborhood*) attesting to the correctness of the response. So long as each neighborhood contains at least one honest node, an adversary should not be able to maliciously influence query results.

In this paper we outline two potential attacks on ShadowWalker. First, we show by simulation that if an adversary can acquire a neighborhood composed entirely of adversarial nodes, then he can launch an "eclipse" attack, actively corrupting honest nodes' routing tables so that, after a very short time, nearly all random walks on the topology are "captured" by adversarial nodes. We demonstrate both analytically and via simulation that an attacker can easily acquire such a neighborhood due to ShadowWalker's small neighborhood size. We present a simple solution to this attack: dramatically increasing the neighborhood size.

Second, we demonstrate a selective denial of service (DoS) attack on ShadowWalker. In order to be considered a valid hop in a random walk, a node must provide signatures from all of its shadows attesting to the correctness of its routing table. We show how adversarial nodes can refuse to provide signatures to legitimate nodes, preventing them from being valid hops in random walks. Our solution to the first attack will actually strengthen this attack as it causes adversarial nodes to shadow several times more nodes than they normally would have. We demonstrate that a very weak attacker can control nearly all steps in all random walks over the network by strategically refusing to sign routing tables. Our solution to this issue is to reduce the number of digital signatures a node needs from its shadows in order to be considered valid.

Thus, we show an inherent tension between resistance to these two attacks: resisting eclipse attacks requires rejecting routing tables with few signatures, while resisting selective DoS requires accepting routing tables with few signatures. We examine the trade-

WPES'10, October 4, 2010, Chicago, Illinois, USA.

off between these parameters and show that they can be tuned to simultaneously make both attacks unlikely, at the cost of dramatically increasing the cryptographic and communication overhead.

To evaluate the overhead incurred by our solution, we construct the first working prototype of ShadowWalker. We discuss several techniques that are required to translate the protocol into a functioning system and demonstrate that a ShadowWalker network using neighborhood sizes we recommend is functional. We deploy our implementation across PlanetLab and measure time requirements for topology maintenance and random walks for circuit construction.

2. BACKGROUND

2.1 Anonymity Systems

Anonymity systems are designed to create unlinkability between two communicating parties. They achieve this goal by mixing together the traffic of a large collection of users, an idea first proposed by Chaum [3]. While an adversary will know that a party is communicating with *someone*, he will, hopefully, not be able to determine who that someone is, beyond a member of a large set.

Modern low-latency systems achieve mixing by forwarding traffic from multiple simultaneous users through several relays. As an example, consider the operation of Tor [5]. Each Tor user acquires a list of all relays in Tor once every hour. Using this list, a Tor client builds a virtual circuit, commonly consisting of three relays. Traffic proceeds to flow from the user to the first relay – the *entry node* – and from there through each other relay in the circuit, until it reaches the last relay – the *exit node*. At this point traffic is forwarded to the actual destination. Return traffic takes the same path in reverse, entering Tor at the exit node and reaching the user via the entry node. This circuit-based system works well; however, it is not without shortcomings. For example, an adversary who controls the first and last hops in a circuit can link the sender and receiver via timing analysis.

If we assume that a user selects relays in an unbiased manner, we can quickly determine what fraction of circuits will be compromised in this manner. If an adversary controls a fraction f of the network, then the chance of the first hop being an adversary is simply f. Out of those circuits, a fraction f will also have an adversarial exit node, leading to a fraction f^2 of all circuits being compromised in this manner. Centralized systems like Tor ensure that relays are selected in an unbiased manner by providing clients with a full view of the network. This full view is costly, though, in terms of bandwidth. In an effort to alleviate this cost, researchers have turned toward peer-to-peer systems as a method of relay discovery.

2.2 Peer-to-Peer Anonymous Communication Systems

Early peer-to-peer anonymous communication systems focused on random walks. Examples of these systems include Crowds [15], MorphMix [16], and Tarzan [6]. In these systems, nodes have knowledge about some fraction of the network. Nodes build circuits by conducting a random walk, starting with a node they know, then walking to a neighbor of that node, and repeating this process for some distance.

Later peer-to-peer anonymous communication systems utilized a distributed hash table (DHT) to build circuits. In a DHT, nodes are assigned an ID and a range of values they are responsible for. Nodes have knowledge of only a fraction of the total nodes in the network, stored in their routing tables. When a node attempts to locate the node responsible for a value, it first asks the node it knows which is closest to the value. This node, in turn, either asks or returns the closest node it knows about. The lookup proceeds until the node owning the value is found. This method of locating nodes requires the maintenance of a far smaller amount of state than would otherwise be needed, in exchange for some number of additional network round trips.

Applying this idea to anonymity systems appears straightforward. A simple example is the behavior of Salsa [13]. In Salsa, relays are assigned an ID and a range of values they own. When a client wishes to build a circuit, it generates a series of random numbers. It then looks up the relay that owns the first value in the series and starts constructing the virtual circuit through that relay. Afterword, it asks the first relay to find the relay that owns the second value and, once found, extends the circuit through this second relay. It proceeds to extend the circuit in this telescoping manner to each value in the list, creating a circuit without global knowledge of the network.

As stated previously, while these peer-to-peer systems do provide a lower overhead, they also open up new attacks. These attacks are made possible in large part by the implicit trust placed in the nodes queried while doing a lookup. Any node that is queried during the lookup is given the opportunity to attempt to tamper with the results of that lookup. Two simple examples are an attacker lying about the result of the lookup, instead reporting the closest colluding node to the value requested; the attacker also may cause any lookup that will not end with a colluding node to fail by dropping the lookup. By performing these attacks, an adversary can influence the number of circuits he compromises. Instead of the case in Section 2.1, where the client is selecting relays with equal probability, the attacker can inflate the chance that his relays are selected, increasing the fraction of compromised circuits above the f^2 compromised in more centralized systems.

2.3 ShadowWalker

ShadowWalker [12] is a peer-to-peer anonymous communication system that retains the DHT used in more recent systems but returns to the earlier idea of random walks to build circuits. In order to avoid the attacks mentioned in Section 2.2, ShadowWalker performs these walks over a secure restricted topology. In ShadowWalker, neighboring nodes vouch for responses to lookups in an attempt to prevent adversarial nodes from capturing those lookups. Vouching nodes, or *shadows*, are selected in a deterministic manner and will provide digital signatures on routing tables used to both maintain DHT routing and perform random walks.

2.3.1 Finger Tables and Shadows

At the lowest level, ShadowWalker's topology is based on a restricted DHT, such as Chord [19] or Koorde [**?**]. For the remainder of this work, we focus on ShadowWalker deployed over Chord. The ID/value space of Chord is typically pictured as a ring which wraps modulo 2^{idbits}. Nodes are assigned an ID and are responsible for all values they are the *successor* for; those between their ID value (inclusive) and the ID value of the next smallest node, their *predecessor* (exclusive). Routing tables, called *finger tables* in Chord, consist of a number of slots equal to the number of bits in IDs. A node fills its ith slot by searching for the successor of its ID plus 2^{i-1}, its *finger*. Thus, a node is aware of one node half the distance around the Chord ring, one node a quarter the distance, one node an eighth, etc. Unlike DHTs with unrestricted routing tables, such as Kademlia [8] or Pastry [17], there is a "correct" node for each slot of the finger table, an important property used by ShadowWalker. Nodes are responsible for building their finger table when they first join the network, and updating their fin-

ger table at regular intervals, called *stabilization intervals*. Mittal and Borisov [12] suggest that stabilization should occur once every second.

Lookups for the node responsible for a given value proceed as follows. At each step in the lookup, a node first checks if it is responsible for the value itself; if so, it returns itself. It next checks if its successor is responsible for the value; if so, it returns that node. Otherwise, it consults its finger table and returns the furthest node from itself that precedes the value. In this way, the lookup successively steps toward nodes that precede the value by less and less, eventually reaching the direct predecessor of the value, who is aware of the direct successor of the value.

On top of this framework, ShadowWalker uses a redundant structured topology. In ShadowWalker, each node, in addition to its fingers, has a set of shadows. All nodes in the network have r shadows, composed of the node's $\lceil r/2 \rceil$ successors and $\lfloor r/2 \rfloor$ predecessors. When a node builds or stabilizes its finger table, it must ask its shadows to verify it. Since the finger table is deterministic, the shadows are able to reconstruct the finger table directly with lookups to the correct values. If the shadows agree with the node's version of the finger table, they will provide digital signatures attesting to this fact. These signatures will be used later by the node to prove that it is not providing an incorrect result for a lookup. Additionally, nodes must discover the shadows of their fingers in order to accomplish lookups securely. Lookups are split into two different categories in ShadowWalker: lookups used to maintain finger tables, called *secure lookups*, and lookups used to build circuits, called *circuit construction*.

2.3.2 Secure Lookups

In order to ensure that adversaries cannot poison routing tables of other nodes as they either attempt to build or maintain them, ShadowWalker insists that every response to a lookup is attested to by some fraction of shadows (a *consensus fraction*). When a node is responding to a query it must also respond with the signatures from its shadows.

In secure lookups, if there is a discrepancy between the finger tables, the authors state the nodes should select the "closest" node. Traditionally this refers to the node closest to the ID being searched for; this actually increases an attacker's ability to capture lookups. We instead interpret closest to be defined as the closest to the querying node, which corresponds to the node closest to theoretical finger slot (i.e. closest to $ID + 2^{i-1}$), in an effort to be more pessimistic toward attack capability. Nodes are aware of the shadows they should expect at each hop in the lookup as they are part of the previous step's finger table; in the case of the first hop, the node comes for the requester's own finger table, which already holds the shadows of each finger.

2.3.3 Circuit Construction

Circuit construction proceeds in a slightly stricter manner than secure lookups. In order to build a circuit in ShadowWalker a node performs a random walk across the network. The two nodes it reaches at the end of this walk will be the entry and exit nodes of the circuit. In order to perform this random walk a node first selects one of its fingers at random and obtains a copy of its finger table. It then proceeds to select a node at random from that finger table and, via the previous node, ask it for its finger table. This is repeated in a telescoping manner as many times as is needed. Mittal and Borisov [12] show that the random walk should be roughly 6 nodes long, since additional nodes beyond 6 provide only limited increases in security [1].

[1]This was based on an analysis of de Bruijn graphs, not Chord

At each step in the random walk, nodes present not only their finger table, but also the finger tables as built by their shadows, and the signatures on each of those. Unlike secure lookups, disagreement between these finger tables is not tolerated. If a node does not have an agreeing finger table and corresponding signature from *all* of its shadows, the finger table is not accepted and the walk is aborted. This far stronger requirement is done in an effort to ensure that relays are chosen uniformly at random, meaning that it would have roughly the same security properties as Tor.

3. THE ECLIPSE ATTACK

3.1 The Attack

The secure lookup system provided in ShadowWalker relies on the assumption that adversaries will never obtain a neighborhood of $r + 1$ adjacent malicious nodes in the Chord ID space, a state we term a *corrupted neighborhood*. If an adversary succeeds in acquiring such a neighborhood, the malicious node with a full set of malicious shadows can return incorrect results for queries. This is made possible by the fact that it can construct any finger table it desires and still provide signatures supporting this table.

With the ability to lie about lookup results from one node, an adversary can launch an attack that attempts to poison the routing tables of other nodes in manner similar to the Eclipse attack proposed by Singh et al. [18]: an attacker provides incorrect responses to lookups in order to place malicious nodes into routing tables in place of honest nodes. This attack is viable in ShadowWalker because each step in the lookup depends on the previous step to provide the correct shadows for the node currently being queried. If a lookup passes through a corrupted neighborhood, the malicious node at its center can capture the lookup, returning another malicious node as the next step. Additionally, since it also reports the next node's shadows, this next malicious node will be able to behave as though it is at the center of a corrupt neighborhood. While the next malicious node in the lookup may actually have a set of shadows that contain legitimate nodes, these legitimate nodes are never considered since they are not reported by the previous node. This will ultimately result in the malicious nodes successfully returning a corrupted result that appears valid.

An adversary in this position can attempt to capture lookups for finger slots of legitimate nodes, increasing the number of finger table entries he controls. However, to capture node n's ith finger table slot, an adversary must capture the lookups of both n and its r shadows. This complication is overcome by the fact that lookups in DHTs tend to converge when they get close to their destination [7]. This results in a slower startup time for this attack, but the creation of subsequent corrupted neighborhoods increases the chance for both a node and its shadows' lookups to be captured.

The attacker can use this attack to increase the number of circuits he compromises. The attacker achieves this in two ways. First, the attacker will dramatically increase the number of finger table slots in legitimate nodes that contain a malicious node, increasing the chance his nodes are selected at random. Second, nodes at the center of corrupt neighborhoods can return finger tables that are filled completely with corrupt nodes at the center of corrupt neighborhoods, ensuring that all subsequent nodes in the circuit will be malicious.

We created a simulator to measure the effectiveness of this attack. The simulator models the behavior of the network from the instant that an attacker acquires a corrupt neighborhood. Malicious nodes actively attempt to corrupt as many finger table slots as possible during the course of the attack. The results of this simulation can be seen in Figures 1 and 2.

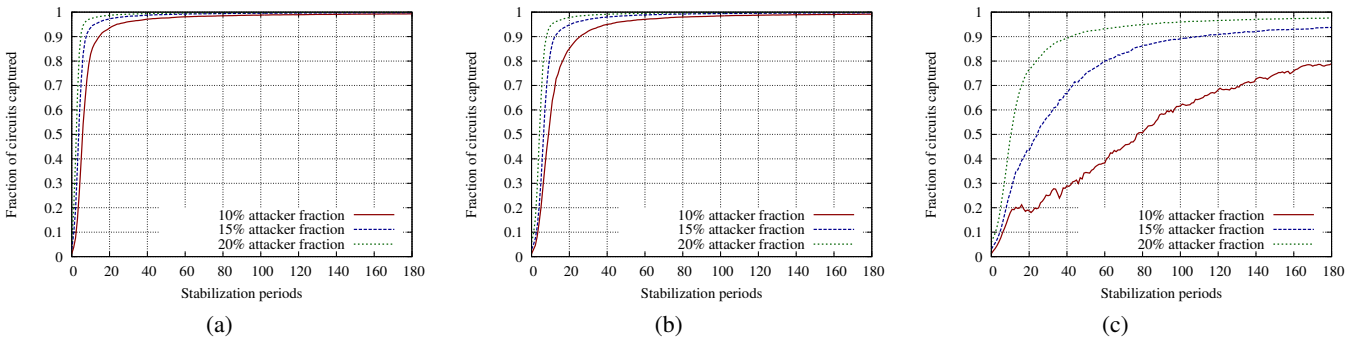

Figure 1: Circuit capture fractions during an eclipse attacks in a network of 1,600 nodes. In all cases the attack starts at time 0, where the attacker is assumed to have a compromised neighborhood. In Figure 1(a) nodes have 2 shadows, in Figure 1(b) they have 3, and in Figure 1(c) they have 4

In our simulation, a randomly distributed network of n nodes is created with attacker fraction f and one compromised neighborhood. Legitimate nodes churn in and out of the network using an exponential churn model with mean uptime of 30 stabilization intervals. Any time a node rebuilds its finger table, whether when joining the network or when stabilizing, each finger table entry has a chance to be captured by the attacker. A finger table entry is captured if all lookups for the finger are captured by the attacker, after which point any lookup that passes through that entry is also captured. Every completed stabilization interval, a series of circuits constructions is attempted, with the number of circuits captured by the attacker reported in Figure 1. We also report, in Figure 2, the fraction of circuits compromised after 60 stabilization intervals as a function of f.

In all of the eclipse attack simulations, the attacker was able to compromise far greater than f^2 of the circuits constructed. With 2 shadows, an attacker fraction of 0.1 had compromised 83% of circuits after only ten stabilizations, as seen in Figure 1(a). With the suggested stabilization interval of one second [12], the attack has only taken ten seconds from the time a compromised neighborhood was attained by the attacker. Increasing the number of shadows to 4, shown in Figure 1(c), slows the spread of the attack, but it is still highly successful. An attacker fraction of 0.1 has compromised 38% of circuits after 60 stabilizations, and compromises 62% of circuits after 100 stabilizations. Simulations run on larger networks had similar results, as expected since the attack is not dependent on network size.

Of course, in order to launch this attack, an attacker first needs to acquire a corrupted neighborhood. Although an attacker could achieve this by launching a Sybil attack or finding a flaw in ID assignment, we do not consider these cases. Instead, we consider the chance an attacker could acquire one such neighborhood by simply joining a collection of malicious nodes and waiting until they achieve one via nodes churning in and out of the network. The chance that an attacker has to achieve such a neighborhood is directly related to the number of shadows each node has. Mittal and Borisov [12] suggest that a network should either use 2 or 3 shadows; we also consider 4 shadows in our analysis. We generated a simulator to find out how long an attacker would have to wait in networks that use ShadowWalker's shadow sizes. The results can be seen in Figure 3 and Figure 4.

In our simulation, a randomly distributed network was constructed with n nodes and an attacker fraction f. Legitimate nodes churned in and out of the network (under an exponential churn model with a mean uptime of 1.0), and the time taken for a compromised neighborhood to develop was recorded.

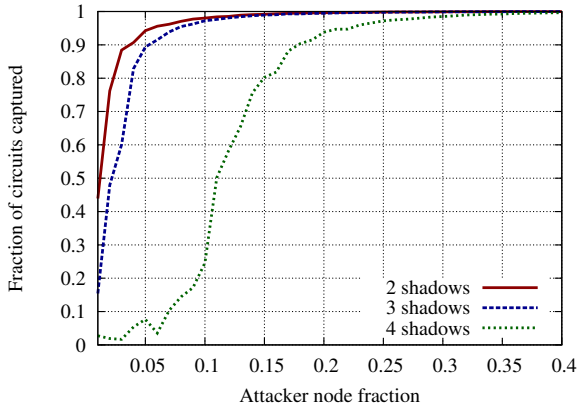

Figure 2: The fraction of circuits captured under an eclipse attack that has been running for 60 stabilization intervals for various attacker sizes. The network size was fixed at 1,600 nodes.

In the case of $r = 2$, seen in Figures 3(a) and 4(a), the simulated network nearly always starts with at least one compromised neighborhood. Increasing the number of shadows increases the difficulty of attaining a compromised neighborhood, as seen in Figures 3(c) and 4(c). However, even with 4 shadows, a 1,600 node network took a median of less than 3 churn intervals to become compromised with attacker fraction 0.15.

3.2 The Solution

We note that, without an effective mechanism for secure ID assignment, no defense against the eclipse attack can work: once an adversary achieves a corrupt neighborhood he will be able to launch a successful attack. Given such a mechanism, an adversary still has the chance to achieve a corrupt neighborhood via nodes randomly churning, as shown in Figure 5 and Figure 6. To approximate the difficulty of generating an attacker-compromised neighborhood, one may consider the probability of such a neighborhood existing in a randomly distributed network of n nodes. For an attacker that controls fraction f of the nodes in the network, the probability that all nodes in one neighborhood are attackers is f^{r+1}, where r is the number of shadows per node. Therefore, the prob-

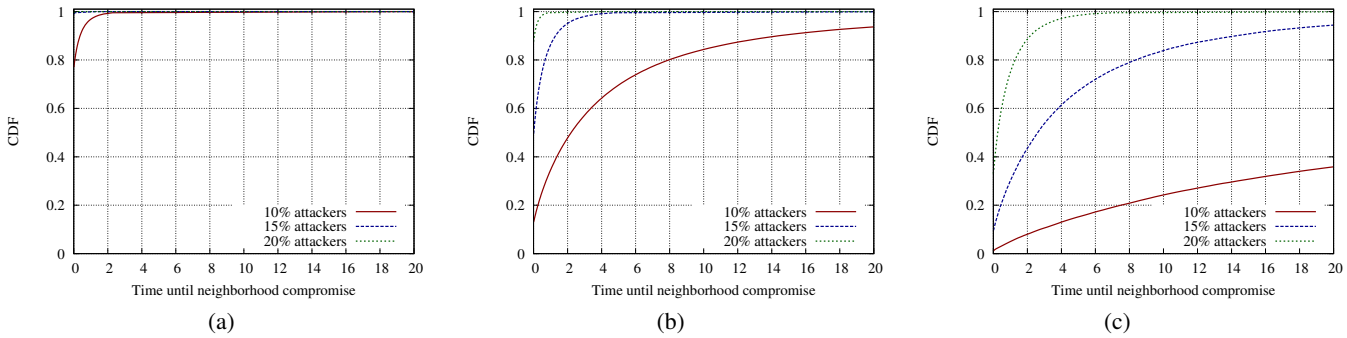

Figure 3: The time (measured with a unit of one mean node uptime) it took an attacker of various strengths to acquire a corrupt neighborhood passively for different neighborhood sizes. The time required is directly tied to node uptime, as node churn is what creates this network state. Network size was fixed at 1,600 nodes, neighborhood sizes are 2 shadows for Figure 3(a), 3 shadows for Figure 3(b), and 4 shadows for Figure 3(c).

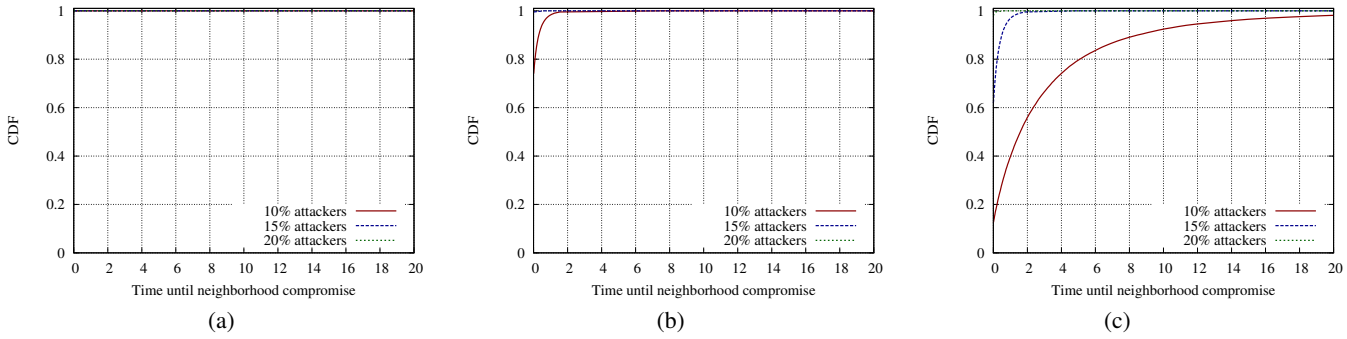

Figure 4: The same attack as shown in Figure 3, but launched against a larger network, in this case 15,000 nodes. Unlike the effect of the eclipse attack, acquiring the position needed to launch an eclipse attack is dependent on network size. Neighborhood sizes are 2 shadows for Figure 4(a), 3 shadows for Figure 4(b), and 4 shadows for Figure 4(c).

ability that at least one compromised neighborhood exists within a random network state is $(1-(1-f^{r+1})^n)$.

An obvious way to reduce this chance is to increase r, the number of shadows. As seen in Figure 6, with 20 shadows, the probability of a compromised neighborhood within the network only exceeds 10^{-10} when the attacker fraction exceeds 0.2, and neighborhood sizes of 40 or 60 shadows do not exceed a probability of 10^{-10} even with attacker fraction 0.4.

4. DENIAL OF SHADOWS ATTACK

4.1 The Attack

The second attack exploits the number of signatures required for a node to participate in circuit construction. The result of a secure lookup is considered valid if a node provides at least one signature. However, during the random walk for circuit construction, nodes must present a full set of signatures. An attacker who wishes to launch attacks on the availability of relays in ShadowWalker need only refuse to provide a node with a matching signature in order to make it non-viable for circuits. ShadowWalker attempts to address this issue by making the shadow relationship symmetric, i.e. if node A is a shadow for node B, then node B is a shadow for node A. In this way, if a node wishes to prevent another node from taking part in circuits, the removed node can reciprocate and remove the attacking node as well.

This defense does not provide any actual protection against this

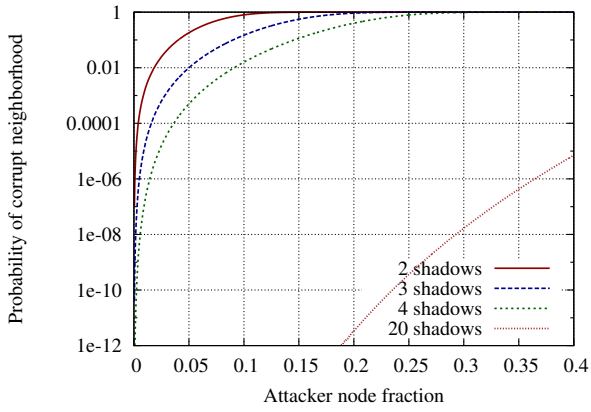

Figure 5: The probability of a 1,600 node network containing a corrupt neighborhood for various neighborhood sizes. Note that the y-axis is in logarithmic scale.

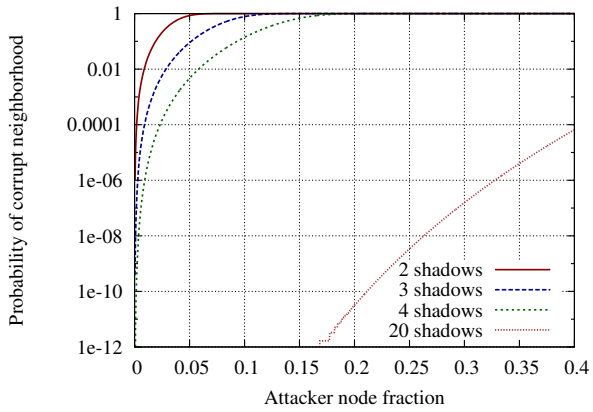

Figure 6: The probability of a 15,000 node network containing a corrupt neighborhood for various neighborhood sizes. Note that the y-axis is in logarithmic scale.

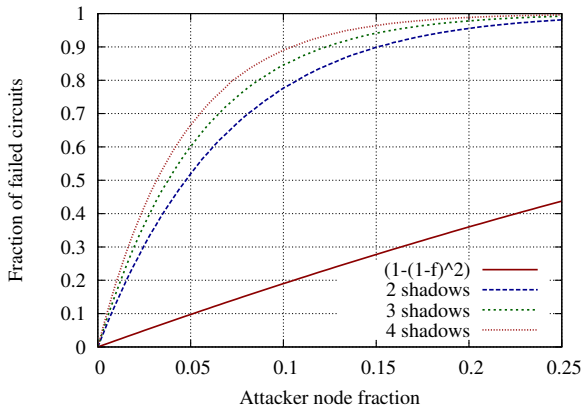

Figure 7: Results of a simulated attack where malicious nodes refuse to provide correct signatures in an effort to cause circuit construction to fail. In this simulation the networks being attacked is a 1,600 node network. We have provided the baseline for the number of circuits an attacker could cause to fail if he could not influence other nodes for reference.

attack for two reasons. First, the attacker benefits from a large force multiplier granted by the number of nodes that each attacking node will shadow. Each node will shadow at least 2 nodes and will have the ability to remove all of these nodes from circuit participation. The fact that this is not a simple $1:1$ trade means that an attacker who controls a small fraction of nodes will still be able to affect a large portion of the network. The second issue with this defense is that, while the attacking nodes are removed from circuit consideration, they still can participate in routing. This is made possible by the fact that secure lookups do not require a full set of shadows. An attacking node needs to simply ensure that it provides, and consequently receives, a signature from at least one node, and it will still be able to participate in the DHT portion of ShadowWalker.

At the most basic level an attacker can destroy relay availability as a whole. In this case, the attacker simply joins some number of malicious nodes to the network and refuses to provide correct signatures to all but one shadow. These attacking nodes will have sufficient signatures to continue functioning as part of the DHT, but will remove, roughly speaking, $r-1$ nodes, each, from circuit consideration.

We implemented a simulator to demonstrate this simple attack. In our simulation, a random network state was generated with the specified attacker fraction and number of nodes. All attackers denied signatures to nodes in their neighborhood, rendering a number of nodes (including those attackers) nonviable with respect to circuit construction. 100,000 circuit constructions were attempted, and the fraction that failed was recorded.

In this simulation, circuits were constructed as described in Section 2.3.3, taking the last two nodes in a six node random walk as the entry and exit points of the circuit. For a circuit to be built successfully, all six nodes on its random walk must be viable to participate in circuit construction. The probability that a random node is viable is $(1-f)^{r+1}$ with attacker fraction f and r shadows per node. A node is only viable if it is not an attacker, and has no attackers in its neighborhood. Therefore, the probability that the entire circuit is viable is $((1-f)^{r+1})^6 = (1-f)^{6(r+1)}$, and it is easy to see why a small attacker fraction is capable of disrupting a large fraction of circuit constructions.

As seen in Figure 7, with only 2 shadows per node, an attacker who controls only 10% of the nodes in the network is able to cause 78% of circuits to fail during construction. Increasing the number of shadows to 4 allows the same attacker fraction of 10% to cause 89% of circuit constructions to fail. Further increasing the number of shadows to the level recommended in Section 3.2 only improves the effectiveness of the attack, as seen in Figure 8, where all circuits are unusable when an attacker controls a mere 5% of the network.

Although this destructive attack is illustrative, there is a far more powerful attack an adversary could mount. An attacker can choose to have some of his nodes deny shadows while the rest behave normally, dramatically increasing the number of circuits he controls. This attack was introduced by Borisov et al. [1].

For a given attacker fraction f, the attacker can choose to have a subfraction f_d of his nodes refuse to sign their neighbors' finger tables; the remaining $(f-f_d)$ fraction of attacker nodes will behave normally. The probability of a legitimate node being viable is the same as the probability that the node has no dishonest attackers in its neighborhood, or $(1-f_d)^r$, and the expected fraction of viable legitimate nodes is then $(1-f)\times(1-f_d)^r$. The expected effective attacker fraction is the fraction of viable attacker nodes divided by the fraction of total viable nodes, or $(f-f_d)/((f-f_d)+(1-f)\times(1-f_d)^r)$. By taking a partial derivative with respect to f_d, the optimal value of f_d can be easily calculated to be $max(0, f-(1-f)/r)$, which implies that this attack is only expected to be successful when $f > 1/(r+1)$.

As mentioned by Mittal and Borisov, for small ShadowWalker neighborhood sizes of 2, 3, and 4, this strategy is largely suboptimal. The exception is the case of 3 shadows, where the odd numbers of shadows allows nodes to violate the reciprocal shadow relationship. With an odd number of shadows, an attacker can freely deny a signature to its one nonreciprocating node, which is then incapable of affecting the attacker's complement of signed finger tables. This added vulnerability allows for the enhanced attack to remain effective at small neighborhood sizes for $r=3$, as seen in Figure 9. For $r=2$ and $r=4$ the attack is ineffective until larger fractions of the network are compromised.

Our changes in Section 3.2 make this attack highly optimal. The fraction of circuits an attacker can compromise under our elevated neighborhood sizes can be seen in Figure 10. In the case of 2 to 4 shadows, the best an attacker could hope for is compromising

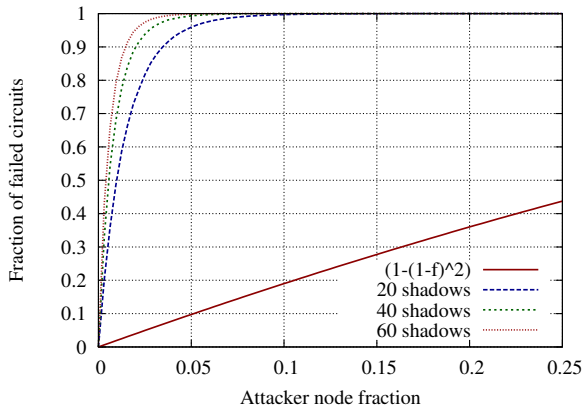

Figure 8: The same attack as Figure 7 but using neighborhood sizes recommended in Section 3.2. The increase in size reduces the requirements for an attacker to compromise circuits.

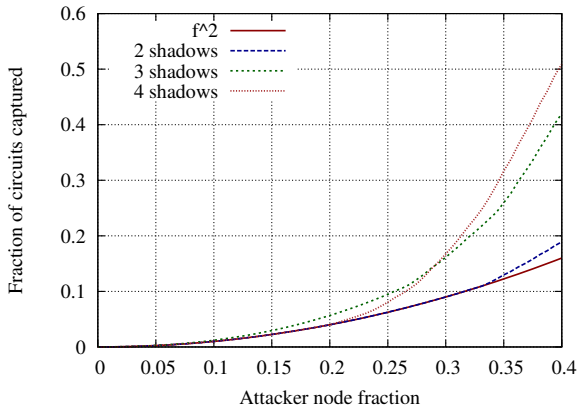

Figure 9: Results of a strategic denial of signature attack in which the attacker attempts to increase the fraction of circuits he controls by reducing the number of honest nodes that can participate in circuits. The attack was run against a network of 1,600 nodes. The baseline of f^2 is provided as a point of reference.

51% of the circuits when he controls 40% of the network. After increasing the neighborhood size to 20, an attacker needs less than 15% of the network in order to compromise the same 51%. Even worse, if 40 or 60 shadows are used, an attacker with that same 15% of the network will compromise nearly all of the circuits. In fact, with 60 shadows, less than 6% of the network needs to be malicious in order for the attacker to compromise more than half of all circuits.

The key issue allowing the attack to occur is the requirement of 100% consensus between shadows for circuit construction. Our solution is to require a smaller consensus fraction of agreeing nodes. This weakens the selective DoS attack, as it requires the attacker to control a large fraction of the network in order to prevent nodes from functioning. Under a reduced consensus where c members of the $(r+1)$-node neighborhood (the node and its r shadows) are required to agree, denial is optimal when $f > (r+1-c)/c$. However, the reduced consensus allows each attacker $(r+1-c)$ "free" denials of shadows, in that the attacker will remain a viable node. Thus, attackers may elect to eliminate some number of legitimate nodes without reprisal, but the overall effect of the attack is greatly weakened.

4.2 The Solution

We re-ran our simulations using decreased consensus fractions. The results can be seen in Figure 11. In the case of a 50% consensus fraction, an attacker sees very little success using selective DoS. Using higher consensus fractions will still allow an attacker to gain some benefit by selectively denying signatures, though the situation is no worse than base ShadowWalker.

At first glance, it would appear as though we have re-introduced our first failure mode using this fix. It is true that by reducing the consensus fraction needed, an adversary stands a greater chance of acquiring a corrupted neighborhood; however, we can demonstrate that the chance is still low, assuming large neighborhood sizes are used. The chance an attacker has to acquire such a neighborhood after taking reduced consensus fractions into account can be seen in Figure 12.

As should be expected, the higher the consensus fraction, the harder it is for an attacker to obtain a compromised neighborhood. The probability P_{NC} of neighborhood compromise, with attacker fraction f, r shadows and a required node consensus of c, is given by the cumulative binomial distribution: a neighborhood must have c or more successes out of $r+1$ trials, with a probability f of success. The probability that a randomly distributed network of n nodes contains at least one compromised neighborhood is $(1-(1-P_{NC})^n)$. By choosing an appropriate consensus fraction and neighborhood size, the probability of a compromised neighborhood can be reduced such that the network is resistant to a desired attacker fraction.

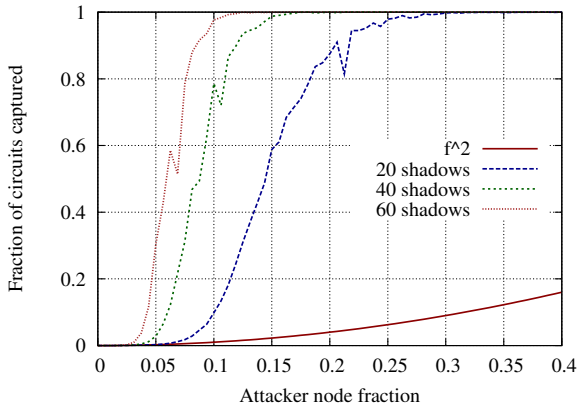

Figure 10: Results of a selective DoS attack launched against networks using expanded neighborhood sizes. The baseline of f^2 is included as a reference.

For example, consider the requirement that the network is resistant to an attacker fraction of 0.25. One could quickly establish that 60 shadows with a 75% consensus fraction would provide a network that has a less than 10^{-10} chance of having a corrupt neighborhood, while only providing an attacker roughly 2% more circuits via selective DoS. If a network resistant to an attacker fraction of 0.2 is needed then it can be quickly seen that 60 shadows is more than required. A network with 40 shadows and 75% consen-

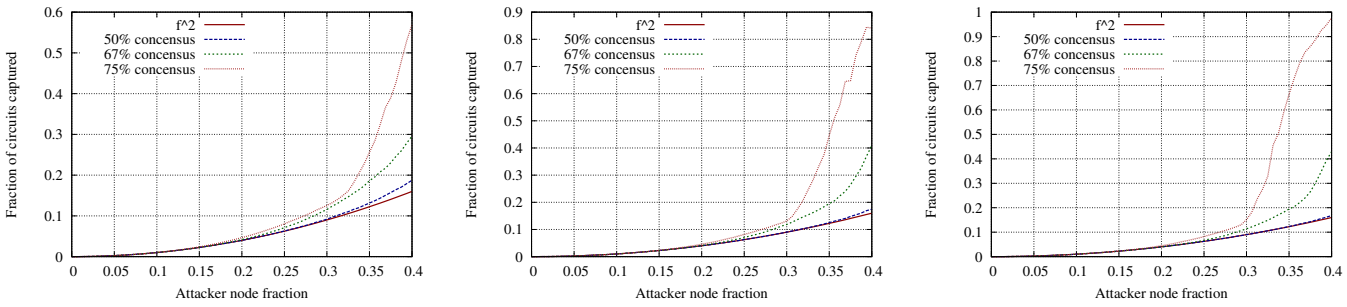

Figure 11: Results of the selective denial of signatures when applied to large neighborhood size networks that use fractional consensus. The network size in this case is 1,600 nodes. Neighborhood sizes are: 20 in Figure 11(a), 40 in Figure 11(b), and 60 in Figure 11(c).

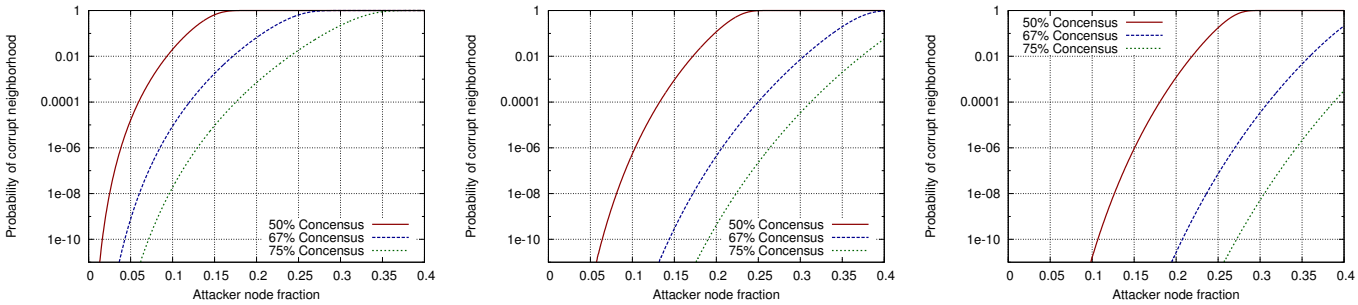

Figure 12: Probability of a corrupt neighborhood existing after relaxing consensus requirements in a 15,000 node network. In Figure 12(a) 20 shadows are used, in Figure 12(b) 40 shadows, and in Figure 12(c) 60 shadows.

sus provides a less than 10^{-10} chance of a corrupt neighborhood and effectively prevents the selective DoS attack.

5. IMPLEMENTATION

In an effort to ensure that ShadowWalker could still feasibly run with between 20 and 60 shadows, we implemented ShadowWalker and tested it on PlanetLab. During the course of the implementation and deployment, several interesting lessons were learned about ShadowWalker.

5.1 Deployment Lessons

It became clear while running our implementation on PlanetLab that small inconsistencies in network state could present large issues. At very high consensus requirements (90% - 100%), a handful of honest but unstable nodes caused other honest nodes to lack sufficient signatures. In some cases, the network recovered; in other cases, the network was partitioned. When the consensus fractions recommended in Section 4.2 were used, this problem virtually disappeared. The lower consensus requirements allowed ShadowWalker to be more tolerant of malfunctioning nodes, leading to increased network robustness.

Extremely short-lived nodes in large enough numbers still present issues to ShadowWalker, even with reduced consensus fractions. This is a direct result of the network not being able to stabilize fast enough to keep up with churn. In order to minimize this issue we recommend that the full ShadowWalker protocol only be run on relays, which tend to be longer lived when compared to clients. Clients can still perform lookups to build their finger table as they would normally, but do not announce themselves as valid nodes to the network. They will not appear on any node's finger table, will not be responsible for any IDs, and, consequently will not need shadows to verify their finger table. This dramatically reduces the overhead required to maintain the network.

5.2 PlanetLab Deployment

We deployed our functioning prototype over PlanetLab in order to test the performance scaling of different neighborhood sizes. Given the load and temperament of the PlanetLab testbed, it also provided an interesting opportunity to test the robustness of our prototype. We deployed ShadowWalker across 100 relatively well-functioning nodes in PlanetLab. Nodes participated in all ShadowWalker behavior: bootstrapping; stabilizing; requesting, and providing signatures; and building circuits. We measured various performance characteristics of the deployed nodes. The length of time it took for a node to request a finger table, receive it, and verify the signatures is show in Figure 13. As neighborhood size increased, so too did the time required to fetch a table, a direct result of the increased table size and number of cryptographic operations needed. In general, 80% or more of tables were fetched and validated within 2 seconds of requesting the table.

Closely related to finger table fetch times is the amount of time it took a node to perform the random walk used to build a circuit. This can be seen in Figure 14: a random walk takes roughly 6 times the amount of time required for a single finger table request. This makes sense, as the random walk is merely 6 finger table requests in succession. The median time required ranged from slightly more

than 2 seconds for 20 shadows to roughly 6 seconds for 60 shadows. This compares favorably with the original author's simulated median circuit construction time of 3.8 seconds.

Last, we measured the amount of time nodes took to perform a stabilization; this can be seen in Figure 15. During a stabilization, a node performs lookups for shadows, fingers, and the shadows of fingers. This graph shows the time required for a node to perform these lookups. These lookups can either be for itself or for one of its shadows. A neighborhood size of 20 nodes is noticeably faster than other sizes. On the other hand, 40 and 60 nodes are roughly the same; in some places, the larger neighborhood size is actually faster than the smaller size. This is a direct result of the fact that stabilization tends to be governed by the slowest nodes in the lookup, hence the larger finger table size and cryptographic load are not as noticeable. A stabilization period should be roughly the same length as the largest amount of time a network is willing to allow a node to spend doing stabilization lookups. From the results of Figure 15, we quickly see that Mittal and Borisov's stabilization period of 1 second is not feasible. Instead we recommend a stabilization period of between 30 and 90 seconds, depending on the network latency of nodes comprising the network.

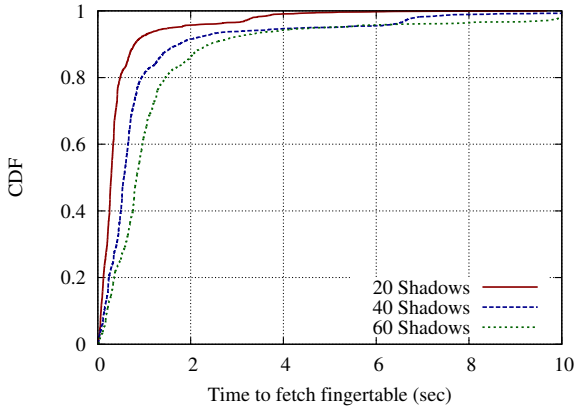

Figure 13: Measured time required to fetch and validate a finger table for various neighborhood sizes.

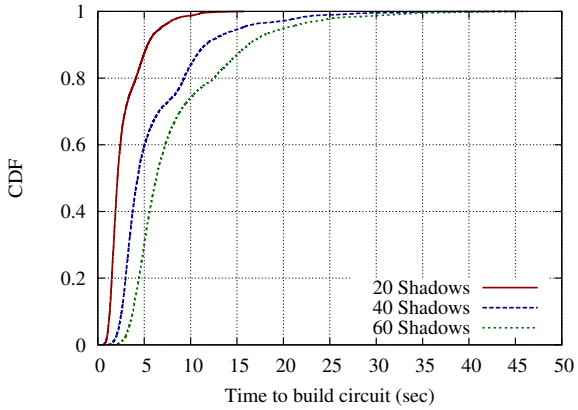

Figure 14: CDF of time required to build a circuit for various large neighborhood sizes as measured during our PlanetLab deployment.

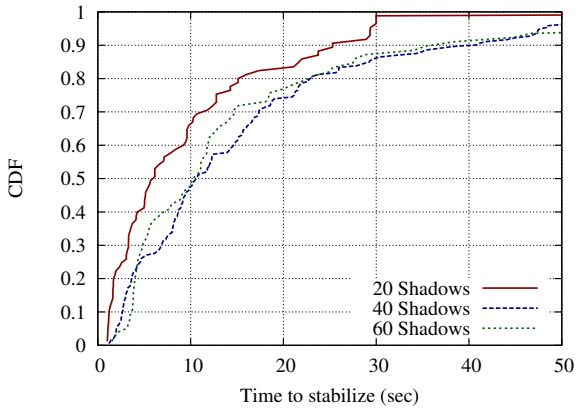

Figure 15: Observed time needed to build a finger table during stabilization for various neighborhood sizes.

6. RELATED WORK

Closely related works have performed a variety of attacks on these systems. For example, Mittal and Borisov [11] studied redundant DHT lookup mechanisms in AP3 [10] and Salsa [13], and showed that these systems offer a tradeoff between resistance to passive and active attacks based on the level of redundancy.

Tran et al. [20] focused on lookup capture and selective DoS attacks against DHT-based anonymous communication systems. Their conclusion was that DHTs are not a very suitable choice for anonymity networks, as there is a mismatch between security requirements for DHT routing and anonymity networks.

The essential basics for our attacks have appeared in prior literature. For example, selective DOS attacks have been described by Borisov et al. [1]. The Eclipse attack was discussed by Singh et al. [18], and the idea of route capture is ubiquitous among anonymity works since at least 2002 [16]. Although our work is based on these papers, the implementations of the attacks themselves are unique to ShadowWalker. This is a result of ShadowWalker's novel topology.

A body of work similar to ShadowWalker exists studying peer-to-peer anonymous communication systems. Several schemes have been proposed [21, 10, 14, 9, 16, 15, 6, 13, 22] that relied on distributed hash tables or random walks to provide scalability. Most of these works have been shown vulnerable to the attacks of [1, 11, 20].

7. CONCLUSION

In this work, we have investigated how to improve ShadowWalker, a DHT-based anonymity system. We have exposed two weaknesses in ShadowWalker and shown how to correct them. First, we looked at what happens when an adversary violates a key assumption of ShadowWalker: the existence of at least one honest node in each neighborhood. We have shown that an attacker who has fully compromised a neighborhood can capture a majority of circuits constructed in the network. We established that the underlying issue is the ease with which an attacker can acquire such a neighborhood in the original design of ShadowWalker, a direct result of the neighborhood size. We fixed this issue by dramatically increasing neighborhood size, minimizing the chance of an attacker violating this assumption. Second, we examined an attack where an adversary refuses to provide signatures to other nodes, making them un-

usable for circuit construction. We simulated two attacks, one in which an adversary disrupts the entire network and another where the attacker is strategic and uses signature denial to capture additional circuits. We resolved this issue by reducing the number of signatures required during circuit construction. We also ensured that this fix does not facilitate an attacker compromising neighborhoods. Last, we examined the challenges of implementing this altered version of ShadowWalker, demonstrating via PlanetLab experiments that our neighborhood sizes are realistic.

Acknowledgments This work was supported by the NSF under grants 0917154 and 0546162. We thank Prateek Mittal and Nikita Borisov for helpful discussions on ShadowWalker.

8. REFERENCES

[1] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 92–102, New York, NY, USA, 2007. ACM.

[2] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 299–314, New York, NY, USA, 2002. ACM.

[3] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[4] George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In *PETS '08: Proceedings of the 8th international symposium on Privacy Enhancing Technologies*, pages 151–166, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[6] Michael J. Freedman and Robert Morris. Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206, New York, NY, USA, 2002. ACM.

[7] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 381–394, New York, NY, USA, 2003. ACM.

[8] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[9] Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. Scalable onion routing with torsk. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 590–599, New York, NY, USA, 2009. ACM.

[10] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. Ap3: cooperative, decentralized anonymous communication. In *EW 11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 30, New York, NY, USA, 2004. ACM.

[11] Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 267–278, New York, NY, USA, 2008. ACM.

[12] Prateek Mittal and Nikita Borisov. Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 161–172, New York, NY, USA, 2009. ACM.

[13] Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 17–26, New York, NY, USA, 2006. ACM.

[14] K.P.N. Puttaswamy, A. Sala, C. Wilson, and B.Y. Zhao. Protecting anonymity in dynamic peer-to-peer networks. In *IEEE International Conference on Network Protocols (ICNP)(Oct. 2008)*, pages 104–113, 2008.

[15] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. Technical report, 1997.

[16] Marc Rennhard and Bernhard Plattner. Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 91–102, New York, NY, USA, 2002. ACM.

[17] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[18] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *EW 11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 21, New York, NY, USA, 2004. ACM.

[19] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[20] Andrew Tran, Nicholas Hopper, and Yongdae Kim. Hashing it out in public: common failure modes of dht-based anonymity schemes. In *WPES '09: Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2009. ACM.

[21] Yingwu Zhu and Yiming Hu. Tap: A novel tunneling approach for anonymity in structured p2p systems. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing*, pages 21–28, Washington, DC, USA, 2004. IEEE Computer Society.

[22] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: resilient anonymous routing. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 301–314, Berkeley, CA, USA, 2005. USENIX Association.