Comments On Selecting Ephemeral Ports[*]

Mark Allman
International Computer Science Institute
mallman@icir.org

ABSTRACT

Careless selection of the ephemeral port number portion of a transport protocol's connection identifier has been shown to potentially degrade security by opening the connection up to injection attacks from "blind" or "off path" attackers—or, attackers that cannot directly observe the connection. This short paper empirically explores a number of algorithms for choosing the ephemeral port number that attempt to obscure the choice from such attackers and hence make mounting these blind attacks more difficult.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Protocol verification; C.2.3 [**Network Operations**]: Network monitoring

General Terms

Measurement,Security,Experimentation

Keywords

TCP, port numbers

1. INTRODUCTION

Transport-layer connections[1] are identified by a set of information that provides the specific rendezvous point for the task at hand, which includes IP addresses, transport protocol number (e.g., 6 for TCP) and service port number (e.g., 80 for web traffic). A final component of the identifier is the ephemeral port number—which is chosen by the connection originator as the source port and does not need to be well established apriori to instantiate communication. The only strong requirement for choosing an ephemeral port is that it be able to disambiguate the connection from all other active connections, as well as those from the recent past in some transport protocols. However, previous work has shown that poor choices for ephemeral port numbers can lead to security vulnerabilities. [14] discusses the case of long-running BGP sessions setup using port 179 for not only the service port, but also as the ephemeral port for the transfer of routing data. The attack calls for an attacker to use a map of the network topology to predict two hosts that would plausibly be exchanging BGP data. The attacker would then have all the information needed to forge a seemingly valid reset packet that would terminate BGP sessions and undermine the Internet's routing system.

Traditional end systems have chosen ephemeral port numbers by using a counter initialized to 1024 (just above the reserved port range) and incrementing the counter each time an ephemeral port number is allocated. While better than simply hard-coding an ephemeral port into an application, this provides for a predictable progression of ephemeral port choices. For instance, if an attacker can get a glimpse of a host's current position in the port space—by traffic monitoring or engaging with the host itself—then that information could be leveraged to potentially impair subsequent connections. While this attack is clearly harder to successfully mount than the previously described BGP attack, the well-defined progression of ephemeral port allocations has led to calls for more advanced ways to *obfuscate* the ephemeral port choices to thwart attacks [6].

While a straightforward way to address a predictable progression of ephemeral port allocation is to choose randomly, this introduces the possibility of *collisions*. In some transport protocols (e.g., TCP [8]) one of the endpoints retains state about terminated connections for a period of time such that new connections cannot be established with the same connection identifier. This serves to ensure that old packets are not considered part of a new connection. Since only one of the endpoints keeps this state the other endpoint can attempt to instantiate a new connection only to see that attempt fail. At first glance the collision rate of a selection algorithm can be readily calculated. For instance, given that some connection chose port x there is a probability of $\frac{1}{N}$ of a subsequent connection randomly choosing port x from a pool of size N. However, the collision rate also *crucially depends on the underlying traffic pattern.* For instance, if at a given time a host tries to make 100 connections to 100 different services there is no chance of collision. However, if those connections are all made to the same remote service then via birthday analysis the chance of experiencing a collision is 7.4% (assuming a 63 KB ephemeral port pool). The problem of collisions is suggested to be bad enough in some cases that [6] discusses algorithms designed specifically to minimize collisions and [10] proposes keeping state on both endpoints involved in a connection to prevent collisions.

In this short paper we present an evaluation of various port selection algorithms based on measurement data from two networks. While these algorithms have been discussed, conjectured about and anecdotally tested we are aware of no systematic evaluation of how the techniques perform

[*] ACM Computer Communication Review, April 2009.

[1] We use the term "connection" loosely in this paper to include request/response transactions over a connectionless protocols (e.g., UDP).

in a general environment. Our goal is to provide empirically rooted guidance to implementers when they choose an ephemeral port selection algorithm. In addition, we shed light on the question as to whether extra state should be kept to avoid collisions.

2. RELATED WORK

We are not aware of previous empirical evaluations of ephemeral port selection schemes. However, our work does relate to several other avenues explored in the community. The impetus for carefully choosing ephemeral ports is in mitigating the risk posed by blind attacks whereby the attacker cannot observe a given flow but can inject traffic into the flow if the connection identifier can be guessed. As discussed above, [14] sketches one concrete example of this kind of attack. Meanwhile, [12] provides a general discussion of blind attacks and mitigation strategies. In addition to port obfuscation, [9] suggests a challenge/response technique for control packets to increase the probability that the actual peer meant to send the given control packet and not an off-path attacker. An additional class of mechanisms have been developed to ensure the integrity of the segments within a connection using cryptographic mechanisms [5, 4, 13]. These techniques rely on signing and/or encrypting packets at the network or transport layer such that even if a blind attacker can guess the ports involved in some connection the attacker lacks the cryptographic material to craft legitimate packets to inject into the stream and therefore any attempt to do so will fail.[2] We note that there is no widespread use of these cryptographic techniques. However, for some traffic there are anecdotes that suggest these cryptographic techniques may be more heavily used (e.g., for BGP traffic). Finally, [1] discusses vulnerabilities stemming from an attacker predicting TCP's initial sequence number and a mitigation strategy, which is similar in spirit to the selection of an obscure ephemeral port number.

3. ALGORITHMS

We employ six port selection algorithms, as follows:

Algorithm 0: This algorithm represents the traditional behavior of many systems whereby the first ephemeral chosen is the value of a counter that is initialized to 1024 (just above the reserved port area), increased by one on each ephemeral port allocation and wrapping back to 1024 after port 65,535 is used.[3] In the case of a collision, the algorithm simply tries the next value of the counter.

Algorithm 1 [6]: This algorithm picks an ephemeral port at random from across the port space. In the case of a collision, the chosen port is incremented by one until a non-colliding port is found.

Algorithm 2 [6]: This algorithm picks an ephemeral port at random from across the port space. In the case of a collision, a new random port is chosen until a non-colliding port is found.

Algorithm 3 [6]: This algorithm is much like Algorithm 0 in that it keeps a host-wide ephemeral port counter C that is initialized to 1024 and incremented by one for each port allocation. Additionally, each time an ephemeral port is needed the host calculates an offset O using some hash function across the source IP address, destination IP address, destination port number and some host-held secret. The port is then chosen as $(C + O)\ mod\ S$ where S is the size of the ephemeral port space. In the case of a collision, the calculation is run again based on the updated value of C. Using both a counter and a service-specific offset this algorithm strives to both obfuscate port selections and maintain a low collision rate. In our simulations we use MD5 [11] as the hash function.

Algorithm 4 [6]: This algorithm builds on Algorithm 3. However, rather than keeping one counter per host Algorithm 4 uses a table of m counters, K_{1-m}, which is initialized with random numbers. O is calculated as in Algorithm 3. Which counter to use, x, is then chosen as $G(O)\ mod\ m$ (using some hash function $G()$). The ephemeral port is then chosen as $(K_x + O)\ mod\ S$ where S is again the size of the ephemeral port space. K_x is then incremented by one. In this way, all connections do not share the same counter and therefore the resulting ephemeral port is more obscured. When a collision occurs a new port is chosen based on the updated value of K_x. As with Algorithm 3, [6] gives no guidance on hash function and therefore we use MD5 for both hash calculations. Further, [6] gives no guidance on the magnitude of m and therefore we vary the table size in our experiments.

Algorithm 5: We introduce a final algorithm that tracks the last ephemeral port number allocated L on each host. In our simulations we initialized L to 1024, but to aid obfuscation L could be initialized to any value in the port space. When a new ephemeral port is needed an increment is randomly chosen from the range $[1, N]$ and added to L (with wrapping when L grows too large). N is a tunable host parameter. This algorithm is meant to ensure that a particular connection cannot collide with previous connections started in the recent past. For instance, if N is 1000 and the port pool is 63 KB we would expect the average increment to be 500 and therefore to be able to create 126 connections before L wraps and the host has even a chance of a collision. The size of N therefore represents a tradeoff between using smaller values to prevent collisions on the one hand and using larger values to aid obfuscation. We vary N in our simulations.

Algorithms are referred to in the remainder of the paper as $\mathcal{A}_n$ where n is the number of the algorithm. For Algorithm 4 we use the notation $\mathcal{A}_{4-m}$ where m is the size of the table employed. For Algorithm 5 we use the notation $\mathcal{A}_{5-s}$ where s is the maximum random increment used in the given simulation.

4. DATA AND METHODOLOGY

To assess the set of algorithms described above we use the connection patterns found in two datasets to drive a simulation of the ephemeral port selection process. We step through a log of TCP connections and when each connection begins we allocate an ephemeral port on the client's behalf using one of the algorithms detailed above. This process takes into account the past activity of the system (i.e., active and recently active connections). Our goal is to assess the ephemeral port collision rate. We divide collisions into two functional categories. *Local collisions* are those whereby

[2] These techniques additionally protect against injection from on-path attackers who can see the connection endpoints and port numbers in use.

[3] We assume a 16-bit port field in the header, as this is common across many transport protocols and in particular it is the size of TCP port numbers used in this study.

an ephemeral port choice results in a 4-tuple that is identical to either a currently active connection or a previous connection for which the host is holding TIME-WAIT state (and therefore is readily detectable by the host itself). On the other hand, a *remote collision* is an ephemeral port choice that leads to a 4-tuple with no local connection state but yet collides with a connection the peer is holding in TIME-WAIT state. Remote collisions can be costly in terms of time because the remote host ignores incoming packets because they are considered to be from an old connection. In our simulations both kinds of collisions result in running the given algorithm again. Because of the inexpensiveness of detecting local collisions we only track remote collisions in the following analysis. For ease of exposition we simply refer to remote collisions as "collisions" in the remainder of the paper.

Our analysis is conducted over logs that include a connection's start time, duration, involved endpoints and service port. We make two choices that likely make our results a overestimate of reality. First, we assume the remote endpoint always holds the TIME-WAIT state and while there are proposals for techniques the remote host could use to shorten the TIME-WAIT state (e.g., [2, 3]) we assume these are not in use. Second, our definition of a connection is quite liberal in that not all "connections" are established and/or accomplish useful work. For instance, a SYN that elicits a reset from the server is considered a connection in our analysis when in reality no state is likely instantiated for such a "connection". Our data includes enough history and/or end state information to *perhaps* remove *some* of these connections from the analysis. However, the data includes enough inherent ambiguity and we do not understand how these connections will be treated by the end hosts and therefore instead of trying to derive how the bogus connections will be treated (which likely varies) we simply assume all connections create state.

(a) ICSI

Day	Conns.	Out. (%)	Hosts	3-Tuples
1	372K	51.2	38K	84K
2	407K	50.7	40K	95K
3	376K	49.0	38K	76K
4	382K	49.1	42K	87K
5	352K	44.7	35K	69K
6	290K	36.7	31K	66K
7	271K	42.7	29K	54K

(b) LBNL

Day	Conns.	Out. (%)	Hosts	3-Tuples
1	9.8M	28.5	325K	1.0M
2	15.7M	63.3	364K	1.9M
3	17.4M	56.8	356K	1.9M

Table 1: Data summary.

Our datasets come from the border of the International Computer Science Institute (ICSI) from August 11–18 2008 and the Lawrence Berkeley National Laboratory (LBNL) from September 14–16 2008. The data is split into day-long connection logs. The LBNL logs are generated by the Bro IDS system [7] as it monitors the network. The ICSI logs are generated by *tcpsum*[4] run across packet traces taken at the institute's border. Table 1 shows high-level characteristics of each dataset. The tables show (*i*) the overall number of connections, (*ii*) the percentage of outgoing traffic (which calibrates the NAT-based analysis in § 6), (*iii*) the number of unique hosts observed and (*iv*) the number of unique (source IP, destination IP, service port[5]) tuples in each dataset. This last item shows the number of connections that *cannot* experience collisions in our simulations since the first connection involving some three-tuple cannot collide using any algorithm.[6]

Finally, we make several notes about the simulations: (*i*) we assume an ephemeral port space of 63 KB, (*ii*) the length of the TIME-WAIT period varies in our experiments as, even though the period is standardized (at 4 minutes), we have anecdotally heard of a variety of times used in practice and (*iii*) we run 10 simulations with each algorithm for each day in our datasets with the exception of $\mathcal{A}_0$ and $\mathcal{A}_3$—neither of which involves randomness.

5. STANDARD CONNECTIVITY SIMULATIONS

The results in table 2 show the average[7] percentage of collisions experienced in simulations based on the ICSI and LBNL datasets. These simulations were run for each algorithm and parameter set we consider (rows) and for three values of TIME-WAIT state retention (columns). We see roughly an order of magnitude more collisions in the LBNL data than in the ICSI data—owing to the size and therefore heterogeneity disparity between the two institutions. The trends are, however, similar across the data. $\mathcal{A}_0$, $\mathcal{A}_3$ and $\mathcal{A}_4$ show no collisions[8] in both datasets. In addition, the prevalence of collisions in the remaining algorithms increase with the length of the TIME-WAIT period. $\mathcal{A}_1$ and $\mathcal{A}_2$ show similar collision rates, which is expected since the essence of each algorithm is the same (i.e., a random port selection). Small increments in $\mathcal{A}_5$ allow for random port choices with lower collision rates than $\mathcal{A}_1$ or $\mathcal{A}_2$. As the length increment increases in $\mathcal{A}_5$ the collision rate approaches that of $\mathcal{A}_1$ and $\mathcal{A}_2$. This is because the as the increment used by $\mathcal{A}_5$ increases the protection against wrapping—and therefore collisions—decreases to the point where instead of methodically stepping through the port space the choices become essentially random. These results also serve to underscore the point that while $\mathcal{A}_5$ can reduce the collision rate with a small increment, the cost is less obfuscation of the resulting choice.

Next we turn to assessing how many port selections a host may have to make when trying to setup a connection. Table 3 shows the average daily maximum number of port selections required to setup a connection. This serves to illustrate a bound on the problem of collisions. Again we see that the

[4] *tcpsum* is bundled with the *tcptb* library, available from http://www.icir.org/mallman/software/tcptb/.

[5] The service port is the destination port in the original SYN packet.

[6] Note, each day is treated independently and clearly collisions can happen in reality on the first connection observed on a given day due to activity on the previous day. Given that each log is 24 hours long we believe this "startup effect" to be small.

[7] We report averages in this paper. The simulations do not vary greatly, with the range being a small fraction of a percent across all our simulations.

[8] Throughout the paper we indicate no collisions with a "—" in the tables for readability.

(a) ICSI

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	0.006569	0.014639	0.025943
$\mathcal{A}_2$	0.006629	0.014632	0.025487
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	—	—	—
$\mathcal{A}_{5-100}$	0.000090	0.000144	0.000187
$\mathcal{A}_{5-500}$	0.000792	0.002722	0.005801
$\mathcal{A}_{5-1000}$	0.001649	0.004181	0.012758

(b) LBNL

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	0.058737	0.164135	0.299890
$\mathcal{A}_2$	0.058896	0.164355	0.299577
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	0.000065	0.000101	0.000092
$\mathcal{A}_{5-100}$	0.001945	0.024399	0.155937
$\mathcal{A}_{5-500}$	0.022260	0.112994	0.243165
$\mathcal{A}_{5-1000}$	0.035634	0.134483	0.268328

Table 2: Average percentage of collisions as a function of both port selection algorithm (rows) and the assumed length of TIME-WAIT state retention (columns).

(a) ICSI

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	2.057143	2.171429	2.471429
$\mathcal{A}_2$	2.028571	2.071429	2.314286
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	—	—	—
$\mathcal{A}_{5-100}$	1.277778	1.421053	1.578947
$\mathcal{A}_{5-500}$	1.833333	1.947368	2.052632
$\mathcal{A}_{5-1000}$	2.000000	2.052632	2.210526

(b) LBNL

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	3.700000	4.633333	5.666667
$\mathcal{A}_2$	3.400000	4.133333	4.766667
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	1.966667	2.000000	2.000000
$\mathcal{A}_{5-100}$	3.000000	3.766667	4.500000
$\mathcal{A}_{5-500}$	3.200000	4.133333	4.800000
$\mathcal{A}_{5-1000}$	3.366667	4.166667	5.000000

Table 3: Average maximum number of ephemeral port choices required per day.

LBNL data shows a wider range of results—highlighting the general notion that network data is heterogeneous and observations from more than one network are key in drawing general conclusions. The data in these tables follows the intuition developed above. $\mathcal{A}_1$ and $\mathcal{A}_2$ are the most prone to many collisions. The worst case for $\mathcal{A}_5$ increases with the increment. Finally, we see that the number of collisions increases as the TIME-WAIT retention period is lengthened. We stress that these tables show the worst case per day. In both datasets when an initial connection attempt yields a collision, one additional port selection suffices in over 95% of the cases regardless of algorithm.[9]

Finally, we assess the percentage of hosts that experience collisions. Table 4 shows the average percentage of hosts that experience at least one collision across our simulations. The general pattern is familiar and expected from the results presented above. For instance, the number of hosts involved increases with the TIME-WAIT period. Comparing these tables with the overall prevalence of collisions (table 2) we note that even though $\mathcal{A}_5$ trends towards an overall similar number of collisions as $\mathcal{A}_1$ and $\mathcal{A}_2$ as the increment is increased, the number of hosts impacted by $\mathcal{A}_5$ is less (by at least a factor of 2) than that of $\mathcal{A}_1$ and $\mathcal{A}_2$. This shows that stepping through the space avoids collisions on hosts with low connection-initiation rates, but cannot avoid collisions when the rate is aggressive such that wrapping is a routine operation.

6. NATED CONNECTIVITY SIMULATIONS

A second situation we consider is that of an enterprise network using a network address translator (NAT) at the border. We run a second set of simulations in which we replace the originator's IP address with a single institution-wide address for each locally-originated connection (at ICSI or LBNL). This has the effect of removing a dimension of the connection identifier and hence may make ephemeral port selections more prone to collisions by using a single port pool across multiple end hosts. Our simulations likely represent an upper bound on the collisions a network would experience because networks may have various proxying and relaying services (e.g., for email and web transactions) and therefore while we use a single external IP address we expect many situations to have at least a few such IP addresses. Finally, in the simulations presented in this section we use a TIME-WAIT period of 240 seconds to show a worst case. We conducted simulations with shorter TIME-WAIT periods and the trends illustrated in the last section hold for NATed connectivity.

Table 5 shows the results of our simulations involving a NAT for the ICSI and LBNL traffic respectively. As with the non-NAT case, the baseline algorithm $\mathcal{A}_0$ shows no collisions in the ICSI simulations. However, we do observe a negligible collision rate in the LBNL data. We find that there are a small number of hosts that do connect to particular servers relatively frequently, as well as services that are simply popular (e.g., Google). When all internal hosts make connections through a NAT and the NAT simply cycles through the port space the probability of collision increases over all the machines independently cycling through the port space. This happens with greater propensity at LBNL than at ICSI simply because LBNL has more hosts, users and traffic competing for the same size ephemeral port pool. Across the entire 3 days of LBNL data we observe an

[9] We find only one exception to this in one simulation of $\mathcal{A}_{5-1000}$ using the ICSI dataset where 10 collisions occurred and only 8 were fixed by the second attempt.

(a) ICSI

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	0.029100	0.044700	0.053600
$\mathcal{A}_2$	0.030300	0.043300	0.052600
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	—	—	—
$\mathcal{A}_{5-100}$	0.000700	0.001100	0.001600
$\mathcal{A}_{5-500}$	0.003000	0.006800	0.011400
$\mathcal{A}_{5-1000}$	0.006300	0.013100	0.020100

(b) LBNL

Alg.	30 sec	120 sec	240 sec
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	0.187200	0.271300	0.319900
$\mathcal{A}_2$	0.190500	0.272000	0.318800
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	0.000300	0.000300	0.000300
$\mathcal{A}_{5-100}$	0.001400	0.002900	0.005400
$\mathcal{A}_{5-500}$	0.019900	0.070000	0.091500
$\mathcal{A}_{5-1000}$	0.057100	0.103300	0.132800

Table 4: Percentage of hosts experiencing collisions.

(a) ICSI

Alg.	Mean	Avg. Max	Hosts
$\mathcal{A}_0$	—	—	—
$\mathcal{A}_1$	0.026779	2.428571	0.000276
$\mathcal{A}_2$	0.025918	2.300000	0.000287
$\mathcal{A}_3$	—	—	—
$\mathcal{A}_{4-5}$	—	—	—
$\mathcal{A}_{4-10}$	—	—	—
$\mathcal{A}_{5-10}$	—	—	—
$\mathcal{A}_{5-100}$	0.001949	1.947368	0.000030
$\mathcal{A}_{5-500}$	0.017328	2.166667	0.000056
$\mathcal{A}_{5-1000}$	0.018825	2.210526	0.000075

(b) LBNL

Alg.	Mean	Avg. Max	Hosts
$\mathcal{A}_0$	0.006795	13.333333	0.000003
$\mathcal{A}_1$	0.317740	5.666667	0.002172
$\mathcal{A}_2$	0.322352	5.000000	0.002203
$\mathcal{A}_3$	0.006795	13.333333	0.000003
$\mathcal{A}_{4-5}$	0.000059	1.875000	0.000002
$\mathcal{A}_{4-10}$	0.000002	1.250000	0.000001
$\mathcal{A}_{5-10}$	0.025706	4.266667	0.000003
$\mathcal{A}_{5-100}$	0.187828	4.800000	0.000035
$\mathcal{A}_{5-500}$	0.268561	4.933333	0.000673
$\mathcal{A}_{5-1000}$	0.291006	5.000000	0.000888

Table 5: Simulation results for NAT simulations.

average of 94 connections/second initiated from within the lab. This means a NAT doing simple cycles would work through the entire 63 KB ephemeral port space in roughly 11.5 minutes. With a 4 minute TIME-WAIT period it is clearly possible that the frequency of accessing a particular service does not have to be significant to increase the collision rate at this traffic level. While the chance of a collision with $\mathcal{A}_0$ is overall low ($< 0.01\%$), we note that when a collision happens it can be problematic to find an available port. The highest average daily worst case is just over 13 attempts! This is directly attributable to high-rate applications and the sequential allocation strategy.

The collision rates for $\mathcal{A}_1$ and $\mathcal{A}_2$ are similar to that shown in the non-NAT case. The NAT results for $\mathcal{A}_1$ and $\mathcal{A}_2$ show that a relatively few "heavy hitters" are responsible for most of the collisions since replacing the source IP address in traffic originated by ICSI and LBNL does not dramatically increase the overall collision rate. Further, we see a small increase in the maximum number of collisions any one connection experiences with the NAT, which shows that some coalescing has occurred in certain cases. The number of hosts involved in the collisions, however, decreased in the NAT case. This is natural because we have coalesced all local hosts into a single host—thereby reducing the number of hosts in the simulation and also critically the number of hosts experiencing collisions.

In the case of $\mathcal{A}_3$ we observe no collisions in the ICSI experiments, just as for the non-NAT case. However, $\mathcal{A}_3$ shows a negligible number of collisions in the LBNL simulations ($< 0.01\%$), in contrast to the non-NAT case. $\mathcal{A}_3$ shares the general cycling of the port space with $\mathcal{A}_0$, except $\mathcal{A}_3$ uses an offset as discussed in § 3. This offset effectively allows for a set of 63 KB different cycles to be happening at the same time. This has the effect of spreading out the use of the port space and therefore $\mathcal{A}_3$ shows fewer collisions than $\mathcal{A}_0$ in the LBNL simulations. This is because the port chosen effectively becomes more randomized and all allocations are not dependent on the same cycling. We see that for $\mathcal{A}_3$—just as for $\mathcal{A}_0$—when a collision happens the worst case can be quite bad, with an average worst case of 13 collisions over our simulations.

The collision rate observed for $\mathcal{A}_4$ is negligible in all cases we tested—regardless of NAT or data source. When compared with $\mathcal{A}_3$ we observe a reduction in the collision rate in the LBNL simulations—even though in both cases the collision rate is negligible. This shows that there is little gain in complicating ephemeral port selection by using multiple counters that independently cycle through the port space. In addition, $\mathcal{A}_4$ shows the best worst case performance (with averages of less than 2, which indicates that on some days we experienced no collisions).

$\mathcal{A}_5$ shows behavior similar to the non-NAT case. As the increment is increased the collision rate converges towards that of $\mathcal{A}_1$ and $\mathcal{A}_2$ while involving less hosts in collisions. We observe a small increase in the worst case number of collisions experienced when establishing a connection through a NAT due to the coalescing effect.

We again note that the worst case performance given in table 5 is just that: the worst case. We find that across all NAT simulations at least 92% of the cases when an initial connection attempt experiences a collision the next attempt is successful.[10]

7. SUMMARY

Across both datasets and standard and NATed connectivity we observe that all the algorithms offer low collision rates—at most 0.3%—and involve only a small number of

[10] There are two exceptions to this general finding. In both simulations, 86% of initial collisions were fixed by one additional re-try.

hosts—again at most 0.3%. $\mathcal{A}_0$ shows nearly no collisions in our study, but also offers the most predictable ephemeral port choices. $\mathcal{A}_1$ and $\mathcal{A}_2$ show the largest collision rates and perform quite similarly. $\mathcal{A}_3$ and $\mathcal{A}_4$ offer the lowest collision rates. Further, $\mathcal{A}_4$ does not require large tables to offer low collision rates. However, larger tables may provide for a higher level of obfuscation. We introduce $\mathcal{A}_5$, which we show offers a middle ground between the algorithms that depend on randomization ($\mathcal{A}_1$ and $\mathcal{A}_2$) and the deterministic algorithms that offer obfuscation but no randomization ($\mathcal{A}_3$ and $\mathcal{A}_4$). The size of the increment used in $\mathcal{A}_5$ allows for direct control of the tradeoff between the level of obfuscation and the collision rate.

Introducing a NAT into the network did not increase the collision rate appreciably. This indicates that a fairly few "heavy hitter" hosts and services are responsible for many of the collisions experienced. This may indicate a need for special handling of such hosts in some cases (e.g., by tracking the TIME-WAIT state on the client as suggested in [10]). We note that an institution-wide NAT or proxy naturally obscures ephemeral port selections simply because to the vast range of services such devices connect to and the overall rate of the traffic. Together these make it difficult at best for an off-path attacker to successfully guess a connection identifier.

Finally, we reiterate that in all cases we needed to make assumptions we chose the worst case (e.g., where TIME-WAIT state is being held, length of TIME-WAIT retention, the definition of a "connection", the number of external addresses, etc.). Therefore, the results presented in this short paper likely represent an upper bound on reality.

Acknowledgments

We thank Vern Paxson for the LBNL data, Joe Touch for comments on an earlier version of the draft and the members of the IETF's Transport Working Group for useful discussions. This work was partially funded by NSF grants 0205519 and 0433702.

8. REFERENCES

[1] S. Bellovin. Defending Against Sequence Number Attacks, May 1996. RFC 1948.

[2] R. Braden. TIME-WAIT Assassination Hazards in TCP, May 1992. RFC 1337.

[3] F. Gont. On the Generation of TCP Timestamps, Oct. 2008. Internet-Draft draft-gont-tcpm-tcp-timestamps-00.txt (work in progress).

[4] A. Heffernan. Protection of BGP Sessions via the TCP MD5 Signature Option, Aug. 1998. RFC 2385.

[5] S. Kent and K. Seo. Security Architecture for the Internet Protocol, Dec. 2005. RFC 4301.

[6] M. Larsen and F. Gont. Port Randomization, Aug. 2008. Internet-Draft draft-ietf-tsvwg-port-randomization-02.txt (work in progress).

[7] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24), 1999.

[8] J. Postel. Transmission Control Protocol, Sept. 1981. RFC 793.

[9] A. Ramaiah, R. Stewart, and M. Dalal. Improving TCP's Robustness to Blind In-Window Attacks, Nov. 2008. Internet-Draft draft-ietf-tcpm-tcpsecure-11.txt (work in progress).

[10] A. Ramaiah and P. Tate. Effects of Port Randomization With TCP TIME-WAIT State, July 2008. Internet-Draft draft-ananth-tsvwg-timewait-00.txt (work in progress).

[11] R. Rivest. The MD5 Message-Digest Algorithm, Apr. 1992. RFC 1321.

[12] J. Touch. Defending TCP Against Spoofing Attacks, July 2007. RFC 4953.

[13] J. Touch, A. Mankin, and R. Bonica. The TCP Authentication Option, Nov. 2008. Internet-Draft draft-ietf-tcpm-tcp-auth-opt-02.txt (work in progress).

[14] P. Watson. Slipping in the Window: TCP Reset Attacks. In *CanSecWest*, 2004.