Errata Slip

revised 10/1/10

Two papers in the OSDI '10 Proceedings have been updated:

"Depot: Cloud Storage with Minimal Trust," by Prince Mahajan, Srinath Setty, Sangmin Lee, Allen Clement, Lorenzo Alvisi, Mike Dahlin, and Michael Walfish, The University of Texas at Austin (Tuesday session on Cloud Storage, pp. 307–322 of the Proceedings)

"TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," by William Enck, The Pennsylvania State University; Peter Gilbert, Duke University; Byung-gon Chun, Intel Labs; Landon P. Cox, Duke University; Jaeyeon Jung, Intel Labs; Patrick McDaniel, The Pennsylvania State University; Anmol N. Sheth, Intel Labs (Wednesday session on Mobility, pp. 393–407 of the Proceedings)

Please see the changes below for each paper.

Changes for "Depot: Cloud Storage with Minimal Trust"
An updated version of this paper can be downloaded from http://www.usenix.org/events/osdi10/tech/full_papers/Mahajan.pdf. The changes include fixes for some typographical errors and minor text adjustments for readability. Please see the following major changes:

1. Section 4.3 omitted text explaining that locally maintained version vectors store the hash of recent updates in addition to logical clocks. The additional text belongs before the paragraph labeled "Identifying a fork" on page 6 and should be a stand-alone paragraph that reads:

   Version-and-hash vectors Each node *N*'s locally maintained *N.VV[M]* contains not only the highest logical clock that *N* has observed for *M* but also a hash of *M*'s update at that logical clock. As a result, if a faulty node creates logically different updates with the same accept stamp, other nodes can detect the discrepancy through update exchange.

2. For clarity, replace the first row in Figure 3, "Summary of main evaluation results," with the following two rows:

Depot adds modest latency relative to a baseline system. Depot's additional GET latency is comparable to checksumming data with SHA-256. For PUTS, 99-percentile latency for 10KB objects increases from 14.8 ms to 27.7 ms.	§7.1
Depot's main resource overheads are client-side storage and client- and server-side CPU use.	§7.1

3. Section 7 omitted a sentence. At the end of the paragraph that begins "Our default configuration is as follows," add a sentence of clarification: "Since receipts require signature checks, our evaluation slightly understates overhead."

4. Figure 6 omitted its legend. There should be a "B", "H", "S", "St", and "D" underneath each group of bars, and the caption should read:

   Per-request average resource use of Baseline (B), B+Hash (H), B+H+Sig (S), B+H+S+Store (St), and Depot (D) in the 100/0 (GET) and 0/100 (PUT) workloads with 10KB objects. The bar heights represent resource use normalized to Depot. The labels indicate the absolute per-request averages. (C) and (S) indicate resource use at clients and servers, respectively. (C-S) and (C-S) are client-server and server-server network use, respectively. For storage costs, we report the cost of storing a version of an object.

5. The discussion of resource overheads in the original was incomplete (Sections 7.1–7.2). To sharpen and clarify it, replace from the paragraph in 7.1 "Figure 5 depicts the results" to the first paragraph of 7.2 with the following:

   Figure 5 depicts the results. For the GET runs, the difference in means between Baseline and B+Hash are 0.0, 0.2, and 15.2 ms for 3B, 10KB, and 1MB, respectively, which are explained by our measurements [45] of mean SHA-256 latencies in the cryptographic library that Depot uses: 0.1, 0.2, and 15.7 ms for those object sizes. Similarly, the means of RSA-Verify operations explain the difference between B+Hash and B+H+Sign for 3B and 10KB, but not for 1MB; we are still investigating that latter case. Depot's GET latency is lower than that of the strongest

two baselines because Depot clients verify signatures in the background, whereas the baselines do so on the critical path. Note that for GETs, Depot does not introduce much latency beyond applying a collision-resistant hash to data stored in an SSP—which prudent applications likely do anyway.

For PUTs, the latency is higher. Each step from B+Hash to B+H+Sign to B+H+S+Store to Depot adds significantly to mean latency, and for large requests, going from Baseline to B+Hash does as well. For example, the mean latency for 10KB PUTs ascends 3.8 ms, 3.9 ms, 8.5 ms, 9.7 ms, 13.0 ms as we step through the systems.

We can explain the observed Depot PUT latency with a model based on measurements of the main steps in the protocol [45]. For example, for 10KB PUTs, the client hashes the value (mean measured time: 0.2 ms), hashes history (≈ 0.1 ms), signs the update (4.2 ms), stores the body (2.6 ms, with the DB cache enabled), stores the update (≈ 1.5 ms), and transfers the update and body over the 1 Gbps network (≈ 0.1 ms); the server verifies the signature (0.3 ms), hashes the value (0.2 ms), hashes history (≈ 0.1 ms), and stores the body (2.6 ms) and update (≈ 1.5 ms). The sum of the means (13.4 ms) is close to the observed latency (13.0 ms). The model is similarly accurate for the 3B experiments but off by 20% for 1MB; we hypothesize that the divergence owes to queues that build in front of BDB during periodic log exchange.

These PUT latencies could be reduced. For example, we have not exploited obvious pipelining opportunities. Also, we experiment on a 1Gbit/s LAN; in many cloud storage deployments, WAN delays would dominate latencies, shrinking Depot's percentage overhead.

Resource use Figure 6 depicts the average use of various resources in the experiments above for 10KB objects. We measure CPU use at the end of a run, summing the user and system time from `/proc/<pid>/stat` on Linux and dividing by the number of requests. We measure network use as the number of bytes handed to TCP.

Depot's overheads are small for network use, server storage, and server CPU on GETs. They are also small for client CPU on GETs, relative to the B+H+Sign baseline. The substantial client storage overheads result from clients' storing data for the PUTs that they create and metadata for all PUTs. The substantial PUT CPU overheads are due to additional Berkeley DB accesses and cryptographic checks, which happen intensively during gossiping. Since the request rate is low relative to the gossip rate, each request pays for a lot of gossip work. With increased request rate (and/or larger objects), this CPU overhead is lower, as shown by the measurements summarized immediately below.

Throughput Most of our evaluation is about Depot's underlying costs as opposed to the performance of the prototype, so we treat throughput only briefly. We ran separate measurements in which we saturated a single Depot server with requests from many clients. For 10KB GETs, a single Depot server can handle 11k requests per second, at which point network bandwidth is the bottleneck. For 10KB PUTs, peak throughput is 700 requests per second. This disappointing number is not surprising given the resource use measured above, but a well-tuned version ought to see sequential disk bandwidth with the bottleneck being signature checks (0.3 ms per core).

7.2 Dollar cost

Is Depot's added consumption of CPU cycles and client-side storage truly costly? To answer this question, we must weight Depot's resource use by the costs of the various resources. To do so, we convert the measured overheads from the prior subsection into dollars (to pick a convenient currency). We use the following cost model, loosely based on what customers pay to use existing cloud storage and compute resources.

Changes for "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones"
An updated version of this paper can be downloaded from http://www.usenix.org/events/osdi10/tech/full_papers/Enck.pdf.

1. Table 2: Moved "Barcode Scanner" to the "Camera only" row.
2. Table 2: Renamed "Layer (Productivity)" to "Layar (Lifestyle)."
3. Table 2: Added a footnote to the "Applications" header that says, "Listed names correspond to the name displayed on the phone and not necessarily the name listed in the Android Market."
4. Table 2: Moved "3001 Wisdom Quotes Lite" to the "(Productivity)" category in the same row.
5. Removed redundant sentence from end of second paragraph of introduction: "This lack of transparency forces users to blindly trust that applications will properly handle private data."