Caching Trust Rather Than Content

M. Satyanarayanan
School of Computer Science
Carnegie Mellon University

1. Position Statement

Caching, one of the oldest ideas in computer science, often improves performance and sometimes improves availability [1, 3]. Previous uses of caching have focused on *data content.* It is the presence of a local copy of data that reduces access latency and masks server or network failures. This position paper puts forth the idea that it can sometimes be useful to merely *cache knowledge sufficient to recognize valid data.* In other words, we do not have a local copy of a data item, but possess a substitute that allows us to verify the content of that item if it is offered to us by an untrusted source. We refer to this concept as *caching trust.*

Mobile computing is a champion application domain for this concept. Wearable and handheld computers are constantly under pressure to be smaller and lighter. However, the potential volume of data that is accessible to such devices over a wireless network keeps growing. Something has to give. In this case, it is the assumption that all data of potential interest can be *hoarded* on the mobile client [1, 2, 6]. In other words, such clients have to be prepared to cope with cache misses during normal use. If they are able to cache trust, then any untrusted site in the fixed infrastructure can be used to stage data for servicing cache misses — one does not have to go back to a distant server, nor does one have to compromise security. The following scenario explores this in more detail.

2. Example Scenario

An engineer with a wearable computer has to visit a distant site for troubleshooting. Because of limited client cache capacity, it is impossible for him to hoard all the repair manuals and proprietary company documents he may require at the site. He therefore has to be prepared to cope with cache misses while on site. Unfortunately, that site only has occasional connectivity via a satellite link to the servers at home. Further, satellite communications are restricted to off-peak hours to reduce cost; at other times, the site is effectively disconnected.

At the remote site there is excellent, high-bandwidth short-range wireless coverage. There are also many machines with ample disk capacity available for temporary use by the engineer. It would be convenient to use one of these machines as a surrogate server, staging data in bulk from the real servers to the surrogate so that cache misses can be serviced efficiently on site. Unfortunately, security is lax at the remote site. The engineer cannot be confident that the surrogate will not be tampered with. Under these circumstance, how can the engineer be assured that the data he accesses at the remote site is indeed authentic?

3. Integrity and Privacy

A common trust model is to assume that servers are physically secure and trusted, and that the client-server communication channel is encrypted for privacy. Staging data at an untrusted surrogate hurts both integrity and privacy. The challenge is to preserve these properties even when the surrogate is physically compromised. This can be accomplished either using private or public key encryption. For brevity, the discussion below focuses on a private key approach. The corresponding public key approach is easy to derive.

Integrity is the easier of the two security properties to preserve. We envision an approach in which the user hoards the fingerprints (such as MD5 checksums [5]) of all files of potential interest directly from the server before leaving on his trip. Since fingerprints are much smaller than file contents, this is only a small burden on the disk capacity of the client. When a cache miss occurs at the remote site, the corresponding data is fetched from the surrogate and its fingerprint is computed by the client; the data is accepted only if the computed and cached fingerprints match.

The problem becomes more complex if data can change at the server after the user leaves home. In that case, the user needs to obtain fresh fingerprints. This requires a trusted channel from client to server, but a low-bandwidth modem link may suffice. A public key approach would be simpler in this regard, since digitally signed updates can be sent over an untrusted channel.

It is simple to extend this idea to privacy. In addition to a fingerprint, the client also hoards a per-file private encryption key. The server encrypts each file before staging it on the surrogate. To handle a cache miss at the remote site, the client fetches the data from the surrogate, decrypts it, verifies its fingerprint and then uses the data. The volume of cached keys can be reduced by using a single private encryption key for all files, at the price of total exposure if that key is broken.

This solution to the privacy problem is not fully satisfactory. It is not possible to ensure purging of staged data from the surrogate because it lies outside the administrative domain of the client and server. With enough time and effort, the keys of staged files can be broken and their contents revealed. The keys can be chosen to be strong enough that breaking them will take much longer than the expected duration of surrogate use. However, it is not feasible to guarantee the privacy of staged data indefinitely. This approach may therefore be restricted to situations where privacy is not an issue, or where there is a well-defined time bound on privacy of information.

4. Status and Plans

We are in the early stages of building a system that uses the idea of caching trust. Our work is being done in the context of the Aura Project at Carnegie Mellon, a new research initiative whose theme is ‘‘distraction-free, ubiquitous computing’’. Support for nomadic data access in Aura uses the Coda File System as a back end. Coda was recently extended to exploit surrogates for efficient update propagation over low-bandwidth networks [4]. We now plan to further extend the system to exploit surrogates for servicing cache misses, as described here. An important implementation question we hope to answer is whether the support for using surrogates securely can be fully encapsulated in a user-level proxy that runs on a Coda client, avoiding changes to Coda itself.

From a broader perspective, opportunistic exploitation of remote infrastructure is key to the long-term success of mobile computing. Unfortunately, security concerns loom large in such architectures. Caching trust may prove to be an important enabling technology for these architectures. The idea is particularly relevant to secure coprocessors and smartcards because their limited storage capacity may be adequate for caching trust but not data content.

Acknowledgements

This research was supported by the National Science Foundation (NSF) under grant number CCR-9901696, and the Defense Advanced Research Projects Agency (DARPA) via the Office of Naval Research (ONR) under contract number N66001-99-2-8918. Additional support was provided by IBM. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, DARPA, ONR, IBM, CMU, or the U.S. Government. Discussions with Jason Flinn, Jan Harkes and Adrian Pavlykevych were valuable in developing these ideas.

References

[1] Kistler, J. J., Satyanarayanan, M.
Disconnected Operation in the Coda File System.
ACM Transactions on Computer Systems 10(1), February, 1992.

[2] Kuenning, G.H., Popek, G.J.
Automated Hoarding for Mobile Computers.
In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, pages 264--275. Saint-Malo, France, October, 1997.

[3] Lampson, B.W.
Hints for Computer System Design.
In *Proceedings of the 9th ACM Symposium on Operating System Principles*. Bretton Woods, NH, October, 1983.

[4] Lee, Y., Leung, K.S., Satyanarayanan, M.
Operation-based Update Propagation in a Mobile File System.
In *Proceedings of the USENIX Annual Technical Conference*. Monterey, CA, June, 1999.

[5] Rivest, R.
The MD5 Message-Digest Algorithm, Internet RFC 1321.
Available from http://theory.lcs.mit.edu/~rivest/publications.html. 1992.

[6] Tait, C.D., Lei, H., Acharya, S., Chang, H.
Intelligent File Hoarding for Mobile Computers.
In *Proceedings of MobiCom'95: The First Annual International Conference on Mobile Computing and Networking*, pages 119-125. Berkeley, CA, November, 1995.