DOI:10.1145/2687878

Article development led by acmqueue
queue.acm.org

Trust, but verify.

BY GEETANJALI SAMPEMANE

Internal Access Controls

EVERY DAY SEEMS to bring news of another dramatic and high-profile security incident, whether it is the discovery of longstanding vulnerabilities in widely used software such as OpenSSL or Bash, or celebrity photographs stolen and publicized. There seems to be an infinite supply of zero-day vulnerabilities and powerful state-sponsored attackers. In the face of such threats, is it even worth trying to protect your systems and data? What can systems security designers and administrators do?

While these threats are very real, they are not the biggest ones faced by most organizations. Most organizations do not face targeted attacks from hostile governments or criminals intent on stealing users' data; their systems are more likely to be unavailable because of ill-timed software updates or misconfiguration.[2,3,4]

People tend to overreact to dramatic events like terrorist attacks, but they underestimate mundane threats. This is made worse by the fact the threat landscape is evolving; security advice that was once reasonable becomes obsolete. For example, users are routinely advised to use long, complex passwords, but account compromise caused by password reuse is probably a bigger threat these days than brute-force password cracking, so choosing different passwords for different sites is a better strategy than creating a complex password, memorizing it, and using it everywhere.

In a former life, I helped organizations connect to the Internet, and, as part of that process, warned administrators of new threats they now faced. Those conversations convinced me that practical systems security was still too difficult for most people to get right. In the years since, Internet connectivity has become more routine, but methods for securing systems have not kept pace.

This article argues in favor of relatively mundane tools that systems security designers and administrators can use to protect their systems and detect attacks. The principles proposed here are good *internal* access controls: regular automated monitoring and verifying of access configurations, and auditing user access to data. At Google we use these techniques as part of our security strategy, but the principles are applicable to any organization with data to protect.

The Problem

Systems security administrators, who have more incentive than the average user to get security right, have a hard job. With the increasing proliferation of mobile devices, and increased expectation of anytime/anywhere access, there are only a few high-security environments where users can be prohibited from bringing their personal phones or devices into the corporate environ-

CREDIT TK

Lorem ipsum dolor sit amet consect adipiscing nunc enim mauris sed massa

ment. Keyboard loggers and malware on personal machines can thus be a path to attack enterprise systems. These devices can be used to exfiltrate data, deliberately or accidentally.

Even when users are restricted to using corporate-owned and -managed devices for work, they still tend to reuse passwords on different systems, and this can provide a vector of attack. Stashes of username/passwords stolen from compromised servers can be retried on other sites, so users who have reused a username/password on multiple sites can contribute to a bigger problem. People remain vulnerable to social engineering or phishing attacks. Improved authentication systems, such as having a second factor or one-time passwords, help some, but the vast majority of systems do not use those yet.

It is therefore reasonable to assume that some user accounts will get compromised, and it is important to design a system to be resilient to that. Such a system also offers the benefit of providing some protection against malicious insiders. Insider attacks have the potential to cause great damage, since people cause them with authorized access and, often, knowledge of systems and processes. Designing protections against insider attacks, however, can be difficult without making the system very cumbersome to use or making users feel untrusted and, therefore, uncooperative with security measures.

Users of the system often do not understand the threat models, so they end up viewing security measures as hoops they have to jump through. Better explanations of the rationale for restrictions may make users more cooperative and dissuade them from looking for ways around the hoops.

Another common problem is misconfigured security controls. As systems and security software grow more complex, the chance of administrators misunderstanding them increases. This can lead to an increase in successful attacks based on such flaws as overlooked default passwords or misconfigured firewall rules.

Why Have Internal Access Controls?

The case for good internal access controls, also called defense in depth, is easy to understand but surprisingly hard to get right in practice. Internal access controls make it harder for attackers to break in (it is not just the firewall that needs to be breached) and limits damage if a system is attacked (one phished password will at most get the attackers what that user has access to, not necessarily everything on the internal network). Given that a common way systems are attacked is via compromised legitimate user accounts, limit-

CREDIT TK

ing the damage a single compromised (or malicious) user can get away with undetected is a useful goal.

The problem is that systems typically start out small, with little or no valuable data, and internal access controls seem like overkill. A good firewall and unrestricted access to (the small number of) authorized users seems like more than enough. People get used to that unrestricted internal access, and processes and tools are developed under that assumption, so adding internal security barriers as the system grows can be disruptive and meet with resistance from users. Removing permissions can also break systems, often in unexpected ways. Retrofitting security into systems is difficult.

Most organizations have different kinds of valuable information that needs protecting—company-confidential code and documents, customer information, or data entrusted to them by their users (in the case of cloud service providers). Different employees need access to different subsets of this information, either for development and debugging services, or to provide customer service, or for routine activities such as indexing or backup. How does the organization ensure people have the right level of access they need and no more?

Most organizations have different kinds of valuable information that needs protecting—company-confidential code and documents, customer information, or data entrusted to them by their users.

Achieving the Right Granularity of Permissions

Administrative usability is often overlooked while designing access schemes. Very fine-grained permissions seem like a good idea, since they can grant exactly the necessary access, but it can easily become too much work to manage. Too many or too low-level permissions can also result in clutter and can be difficult to understand and reason about.

On the other hand, the problem with access that is too coarse-grained is it can grant too much access. One of the bigger problems with granting too much access is not malicious use but accidental use. Many systems do not enable permissions on an as-needed basis but, rather, have all the permissions a user is granted; this is the equivalent of always running as a superuser rather than as a regular user. Again the problem is one of granularity—having to specify every permission needed becomes tedious, so the tendency is just to leave permissions enabled.

Role-based access control systems[1] help with this by grouping related sets of permissions, but people who perform different roles still end up with a lot of access and not-always great ways of using the least-privileged access possible.

What can be done about this? Try to understand the system well enough to set up access controls at the right places, but also recognize that you will sometimes get this wrong and will grant more or less access than is needed. This may be because you want to simplify administration or because your mental model of permissions and usage is wrong. It is thus useful to have a system in place to review and monitor permissions, and correct the access configuration as appropriate.

Monitoring Access Configurations

Too often, access requests are reviewed at grant time and never again. People in an organization move across roles and projects, but old permissions do not always expire. Removing unused permissions rarely seems that urgent, and guessing wrong about whether something is unused can break running systems. Unused permissions are not dangerous as long as they remain unused, but they do make the access configuration harder to understand.

At Google we use regular monitoring of access configurations to identify unexpected or unwanted permission behavior. The principle of access-configuration monitoring is much like unit testing for code. Like any type of verification, this is most useful if the verification uses a different approach from the configuration—for example, viewing the permissions in the live production configuration rather than just viewing them as configured.

Administrators specify invariants about the access configuration that should be maintained, and automated test infrastructure periodically verifies these invariants hold. Preconfigured alerts can be raised if any problems are detected.

Access-configuration monitoring is useful for a few different purposes:

▸ *Catching differences between static and live configurations.* Some access systems require configuration changes

to be reviewed by administrators and then "pushed" to take effect. Occasionally, changes are pushed to live systems without changing the static configuration, or the configuration is changed and not pushed. This sort of situation can lead to unpleasant surprises when long-running systems are restarted.

▸ *Verifying the configuration is behaving as expected.* Most configuration languages have their quirks, so it is good to have tests to confirm they are doing what you expect them to do. A common example is firewall rules that block too much or too little traffic.

▸ *Tripwire-like monitoring to notify people of changes.* Typically, these are expected changes, but this can catch unauthorized or unexpected changes. It is important that these not be too noisy, or people who receive them will tune them out.

▸ *Catching drifts such as sudden (or even gradual) increases in the number of authorized people.* People often create an ACL (access-control list) for a particular reason, and, over time, tend to use it for other reasons, and the size grows. This sort of monitoring can be useful for recognizing when a group has grown too large, contains too many permissions, and should be split.

▸ *Verifying that separation of permissions holds.* For example, you may want to prevent any one person from having certain combinations of permissions (like being able to make changes to code and push them to production without review).

Auditing to Understand Access

Audit logs are a common part of systems security. Typically, all configuration changes and any access to sensitive data generate audit logs, which are hard to subvert. These are often a requirement for regulatory compliance.

Many systems, however, stop at generating the audit logs, using them only for postmortem analysis when something goes wrong. An "audit" in these systems is a sign of trouble. Therefore, access audits should be much more routine, and not a hostile process. Whenever an employee performs a nonroutine access, perhaps for troubleshooting or debugging, the access will be audited. In most cases, this may involve just documenting the reason for access. This develops a culture of accountability, where users expect to have to justify access to sensitive data.

Knowing that all accesses are audited makes granting permissions a little easier. Restricting access to very few people can make a system fragile. It would be more robust if more people were granted emergency access but did not have to use it. Having overbroad permissions, however, is generally a problem. Users could accidentally or maliciously misuse their accesses or become targets for social-engineering attacks because of it. Having good audit logs at the time of *use* of permissions mitigates this risk somewhat, since inappropriate access is unlikely to go undetected.

Routine access audits also help identify access patterns and can help tune access configuration. If all access is logged, it becomes possible to identify unused permissions reliably and prune them safely if needed. This catches the cases where people move jobs or roles without explicitly giving up permissions.

Auditing accesses that are actually used provides visibility into which accesses are needed for people to do their jobs. This allows for the development of better tools, sometimes reducing the amount of access that needs to be granted for a particular task.

Good tools are needed to prevent access audits from becoming bureaucratic nightmares. Routine access can be recognized, based on job roles or access history, and only unusual access patterns can be flagged for extra or manual review.

It is also worth noting that auditing accesses is not a substitute for good access controls; audits can recognize inappropriate access only after it has happened, unlike access controls, which prevent it. As just described, however, auditing all accesses can help tune access configurations. Having to justify access also helps prevent inappropriate access by authorized users. Further, in the unfortunate event of inappropriate access, audit logs can help administrators assess the damage.

Conclusion

While high-profile targeted attacks will continue, organizations can do a lot to protect their systems. Internal access controls at the right granularity, combined with access logging and auditing, can help detect and prevent unwanted access. Access configurations suffer from "bit rot," and users often accumulate unnecessary permissions over time; therefore, regular monitoring, a la unit tests for code, can help detect unwanted situations.

Making security goals and threats clear to system users may encourage their cooperation, rather than leaving them to view security as a nuisance to be worked around. Making the system and security configuration easy for administrators to understand will likely lead to fewer configuration errors, and well-designed monitoring can catch any remaining ones. Finally, making access audits routine can help system administrators understand access patterns and notice unusual access, whether it is a result of some nonroutine event or because a user account has been compromised. C

Related articles on queue.acm.org

A Decade of OS Access-control Extensibility
Robert N. M. Watson
http://queue.acm.org/detail.cfm?id=2430732

Standardizing Storage Clusters
Garth Goodson, Sai Susarla, and Rahul Iyere
http://queue.acm.org/detail.cfm?id=1317402

Monitoring and Control of Large Systems with MonALISA
Iosif Legrand, Ramiro Voicu, Catalin Cirstoiu, Costin Grigoras, Latchezar Betev, and Alexandru Costan
http://queue.acm.org/detail.cfm?id=1577839

References

1. Computer Security Resource Center. Role based access control and role based security. National Institute of Standards and Technology, Computer Security Division, 2014; http://csrc.nist.gov/groups/SNS/rbac/.
2. Hockenson, L. Facebook explains the cause behind its early Thursday downtime. Gigaom; https://gigaom.com/2014/06/19/facebook-explains-the-cause-behind-its-early-thursday-downtime/.
3. Moscaritolo, A. Verizon billing system hit by major outage. *PC Mag UK*, 2014; http://uk.pcmag.com/news/33726/verizon-billing-system-hit-by-major-outage.
4. Wikipedia. RBS Group computer system problems, 2012; http://en.wikipedia.org/wiki/2012_RBS_Group_computer_system_problems.

Geetanjali Sampemane (geta@google.com) belongs to the Infrastructure Security and Privacy group at Google. She started her career administering India's first connection to the Internet and then spent a few years working for the United Nations Development Program, helping developing countries connect to the Internet.