BeyondCorp: Design to Deployment at Google

Article · March 2016

CITATIONS
4

READS
496

4 authors, including:

Max Saltonstall
Google Inc.
2 PUBLICATIONS **4** CITATIONS

SEE PROFILE

BeyondCorp
Design to Deployment at Google

BARCLAY OSBORN, JUSTIN MCWILLIAMS, BETSY BEYER, AND MAX SALTONSTALL

Barclay Osborn is a Site Reliability Engineering Manager at Google in Los Angeles. He previously worked at a variety of software, hardware, and security startups in San Diego. He holds a BA in computer science from the University of California, San Diego. barclay@google.com

Justin McWilliams is a Google Engineering Manager based in NYC. Since joining Google in 2006, he has held positions in IT Support and IT Ops Focused Software Engineering. He holds a BA from the University of Michigan, Ann Arbor. jjm@google.com

Betsy Beyer is a Technical Writer for Google Site Reliability Engineering in NYC. She has previously provided documentation for Google Data Center and Hardware Operations teams. Before moving to New York, Betsy was a lecturer in technical writing at Stanford University. She holds degrees from Stanford and Tulane. bbeyer@google.com

Max Saltonstall is a Program Manager for Google Corporate Engineering in New York. Since joining Google in 2011 he has worked on advertising products, internal change management, and IT externalization. He has a degree in computer science and psychology from Yale. maxsaltonstall@google.com

The goal of Google's BeyondCorp initiative is to improve our security with regard to how employees and devices access internal applications. Unlike the conventional perimeter security model, BeyondCorp doesn't gate access to services and tools based on a user's physical location or the originating network; instead, access policies are based on information about a device, its state, and its associated user. BeyondCorp considers both internal networks and external networks to be completely untrusted, and gates access to applications by dynamically asserting and enforcing levels, or "tiers," of access.

We present an overview of how Google transitioned from traditional security infrastructure to the BeyondCorp model and the challenges we faced and the lessons we learned in the process. For an architectural discussion of BeyondCorp, see [1].

Overview

As illustrated by Figure 1, the fundamental components of the BeyondCorp system include the Trust Inferer, Device Inventory Service, Access Control Engine, Access Policy, Gateways, and Resources. The following list defines each term as it is used by BeyondCorp:

- Access requirements are organized into ***Trust Tiers*** representing levels of increasing sensitivity.
- ***Resources*** are an enumeration of all the applications, services, and infrastructure that are subject to access control. Resources might include anything from online knowledge bases, to financial databases, to link-layer connectivity, to lab networks. Each resource is associated with a minimum trust tier required for access.
- The ***Trust Inferer*** is a system that continuously analyzes and annotates device state. The system sets the maximum trust tier accessible by the device and assigns the VLAN to be used by the device on the corporate network. These data are recorded in the Device Inventory Service. Reevaluations are triggered either by state changes or by a failure to receive updates from a device.
- The ***Access Policy*** is a programmatic representation of the Resources, Trust Tiers, and other predicates that must be satisfied for successful authorization.
- The ***Access Control Engine*** is a centralized policy enforcement service referenced by each gateway that provides a binary authorization decision based on the access policy, output of the Trust Inferer, the resources requested, and real-time credentials.
- At the heart of this system, the ***Device Inventory Service*** continuously collects, processes, and publishes changes about the state of known devices.
- Resources are accessed via ***Gateways***, such as SSH servers, Web proxies, or 802.1x-enabled networks. Gateways perform authorization actions, such as enforcing a minimum trust tier or assigning a VLAN.

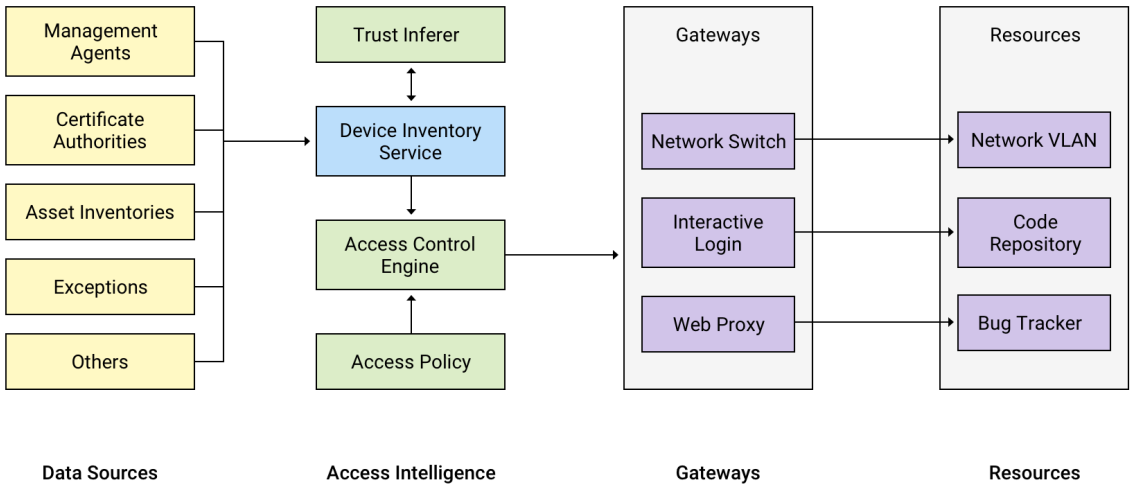

Figure 1: Architecture of the BeyondCorp Infrastructure Components

Components of BeyondCorp

Using the components described below, BeyondCorp integrated various preexisting systems with new systems and components to enable flexible and granular trust decisions.

Devices and Hosts

An inventory is the primary prerequisite to any inventory-based access control. Depending on your environment and security policy, you may need to make a concerted effort to distinguish between devices and hosts. A *device* is a collection of physical or virtual components that act as a computer, whereas a *host* is a snapshot of the state of a device at a given point in time. For example, a device might be a laptop or a mobile phone, while a host would be the specifics of the operating system and software running on that device. The Device Inventory Service contains information on devices, their associated hosts, and trust decisions for both. In the sections below, the generic term "device" can refer to either a physical device or a host, depending on the configuration of the access policy. After a basic inventory has been established, the remainder of the components discussed below can be deployed as desired in order to provide improved security, coverage, granularity, latency, and flexibility.

Tiered Access

Trust levels are organized into *tiers* and assigned to each device by the Trust Inferer. Each resource is associated with a minimum trust tier required for access. In order to access a given resource, a device's trust tier assignment must be equal to or greater than the resource's minimum trust tier requirement. To provide a simplified example, consider the use cases of various employees of a catering company: a delivery crew may only require a low tier of access to retrieve the address of a wedding, so they don't need to access more sensitive services like billing systems.

Assigning the lowest tier of access required to complete a request has several advantages: it decreases the maintenance cost associated with highly secured devices (which primarily entails the costs associated with support and productivity) and also improves the usability of the device. As a device is allowed to access more sensitive data, we require more frequent tests of user presence on the device, so the more we trust a given device, the shorter-lived its credentials. Therefore, limiting a device's trust tier to the minimum access requirement it needs means that its user is minimally interrupted. We may require installation of the latest operating system update within a few business days to retain a high trust tier, whereas devices on lower trust tiers may have slightly more relaxed timelines.

To provide another example, a laptop that's centrally managed by the company but that hasn't been connected to a network for some period of time may be out of date. If the operating system is missing some noncritical patches, trust can be downgraded to an intermediate tier, allowing access to some business applications but denying access to others. If a device is missing a critical security patch, or its antivirus software reports an infection, it may only be allowed to contact remediation services. On the furthest end of the spectrum, a known lost or stolen device can be denied access to all corporate resources.

In addition to providing tier assignments, the Trust Inferer also supports network segmentation efforts by annotating which VLANs a device may access. Network segmentation allows us to restrict access to special networks—lab and test environments, for example—based on the device state. When a device becomes

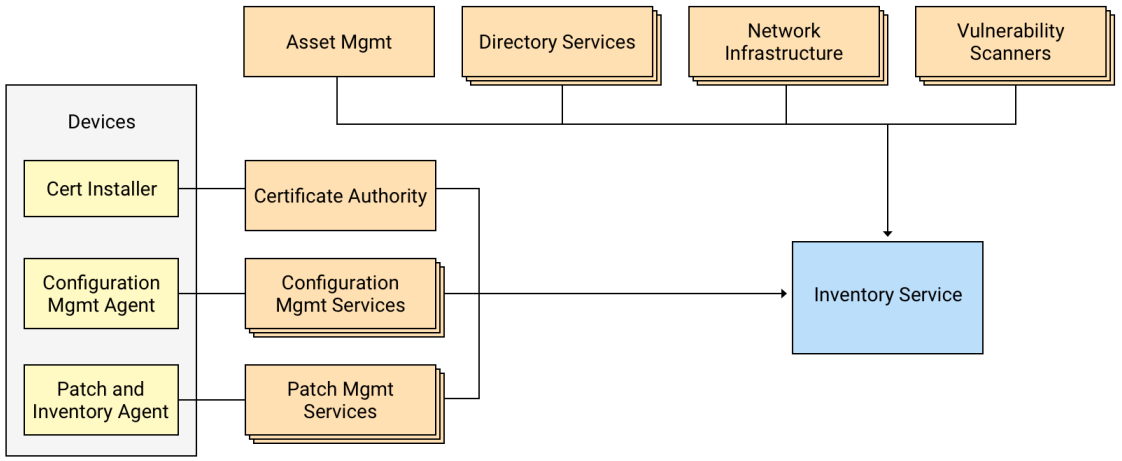

Figure 2: Device Inventory Service

untrustworthy, we can assign it to a quarantine network that provides limited resource access until the device is rehabilitated.

Device Inventory Service

The Device Inventory Service (shown in Figure 2) is a continuously updated pipeline that imports data from a broad range of sources. Systems management sources might include Active Directory, Puppet, and Simian. Other on-device agents, configuration management systems, and corporate asset management systems should also feed into this pipeline. Out-of-band data sources include vulnerability scanners, certificate authorities, and network infrastructure elements such as ARP tables. Each data source sends either full or incremental updates about devices.

Since implementing the initial phases of the Device Inventory Service, we've ingested billions of deltas from over 15 data sources, at a typical rate of about three million per day, totaling over 80 terabytes. Retaining historical data is essential in allowing us to understand the end-to-end lifecycle of a given device, track and analyze fleet-wide trends, and perform security audits and forensic investigations.

Types of Data

Data come in two main flavors: observed and prescribed.

Observed data are programmatically generated and include items such as the following:

- The last time a security scan was performed on the device, in addition to the results of the scan
- The last-synced policies and timestamp from Active Directory
- OS version and patch level
- Any installed software

Prescribed data are manually maintained by IT Operations and include the following:

- The assigned owner of the device
- Users and groups allowed to access the device
- DNS and DHCP assignments
- Explicit access to particular VLANs

Explicit assignments are required in cases of insufficient data or when a client platform isn't customizable (as is the case for printers, for example). In contrast to the change rate that characterizes observed data, prescribed data are typically static. We analyze data from numerous disparate sources to identify cases where data conflict, as opposed to blindly trusting a single or small number of systems as truth.

Data Processing

TRANSFORMATION INTO A COMMON DATA FORMAT

Several phases of processing are required to keep the Device Inventory Service up to date. First, all data must be transformed into a common data format. Some data sources, such as in-house or open source solutions, can be tooled to publish changes to the inventory system on commit. Other sources, particularly those that are third party, cannot be extended to publish changes and therefore require periodic polling to obtain updates.

CORRELATION

Once the incoming data are in a common format, all data must be correlated. During this phase, the data from distinct sources must be reconciled into unique device-specific records. When we determine that two existing records describe the same device, they are combined into a single record. While data correlation may appear straightforward, in practice it becomes quite complicated because many data sources don't share overlapping identifiers.

For example, it may be that the asset management system stores an asset ID and a device serial number, but disk encryption escrow stores a hard drive serial number, the certificate

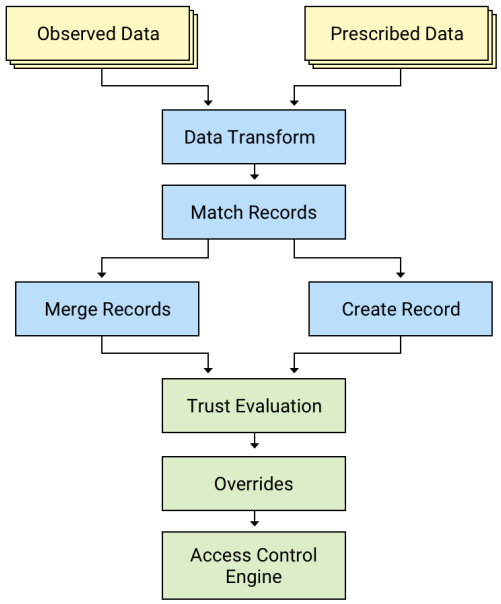

Figure 3: The data processing pipeline

authority stores a certificate fingerprint, and an ARP database stores a MAC address. It may not be clear that deltas from these individual systems describe the same device until an inventory reporting agent reports several or all of these identifiers together, at which point the disjoint records can be combined into a single record.

The question of what, exactly, constitutes a device becomes even more muddled when you factor in the entire lifecycle, during which hard drives, NICs, cases, and motherboards may be replaced or even swapped among devices. Even more complications arise if data are manually entered incorrectly.

TRUST EVALUATION

Once the incoming records are merged into an aggregate form, the Trust Inferer is notified to trigger reevaluation. This analysis references a variety of fields and aggregates the results in order to assign a trust tier. The Trust Inferer currently references dozens of fields, both platform-specific and platform-agnostic, across various data sources; millions of additional fields are available for analysis as the system continues to evolve. For example, to qualify for a high level of trust, we might require that a device meets all (or more) of the following requirements:

- Be encrypted
- Successfully execute all management and configuration agents
- Install the most recent OS security patches
- Have a consistent state of data from all input sources

This precomputation reduces the amount of data that must be pushed to the gateways, as well as the amount of computation that must be expended at access request time. This step also allows us to be confident that all of our enforcement gateways are using a consistent data set. We can make trust changes even for inactive devices at this stage. For example, in the past, we denied access for any devices that may have been subject to Stagefright [2] before such devices could even make an access request. Precomputation also provides us with an experiment framework in which we can write pre-commit tests to validate changes and canary small-percentage changes to the policy or Trust Inferer without impacting the company as a whole.

Of course, precomputation also has its downsides and can't be relied on completely. For example, the access policy may require real-time two-factor authentication, or accesses originating from known-malicious netblocks may be restricted. Somewhat surprisingly, latency between a policy or device state change and the ability of gateways to enforce this change hasn't proven problematic. Our update latency is typically less than a second. The fact that not all information is available to precompute is a more substantial concern.

EXCEPTIONS

The Trust Inferer has final say on what trust tier to apply to a given device. Trust evaluation considers preexisting exceptions in the Device Inventory Services that allow for overrides to the general access policy. Exceptions are primarily a mechanism aimed at reducing the deployment latency of policy changes or new policy primitives. In these cases, the most expedient course of action may be to immediately block a particular device that's vulnerable to a zero-day exploit before the security scanners have been updated to look for it, or to permit untrusted devices to connect to a lab network. Internet of Things devices may be handled by exceptions and placed in their own trust tier, as installing and maintaining certificates on these devices could be infeasible.

Deployment

Initial Rollout

The first phase of the BeyondCorp rollout integrated a subset of gateways with an interim meta-inventory service. This service comprised a small handful of data sources containing predominantly prescribed data. We initially implemented an access policy that mirrored Google's existing IP-based perimeter security model, and applied this new policy to untrusted devices, leaving access enforcement unchanged for devices coming from privileged networks. This strategy allowed us to safely deploy various components of the system before it was fully complete and polished and without disturbing users.

In parallel with this initial rollout, we designed, developed, and continue to iterate a higher-scale, lower-latency meta-inventory solution. This Device Inventory Service aggregates data from

over 15 sources, ingesting between 30–100 changes per second, depending on how many devices are actively generating data. It is replete with trust eligibility annotation and authorization enforcement for all corporate devices. As the meta-inventory solution progressed and we obtained more information about each device, we were able to gradually replace IP-based policies with trust tier assignments. After we verified the workflows of lower-tiered devices, we continued to apply fine-grained restrictions to higher trust tiers, proceeding to our ultimate goal of retroactively increasing trust tier requirements for devices and corporate resources over time.

Given the aforementioned complexity of correlating data from disparate sources, we decided to use an X.509 certificate as a persistent device identifier. This certificate provides us with two core functionalities:

- If the certificate changes, the device is considered a different device, even if all other identifiers remain the same.
- If the certificate is installed on a different device, the correlation logic notices both the certificate collision and the mismatch in auxiliary identifiers, and degrades the trust tiers in response.

Thus, the certificate does not remove the necessity of correlation logic; nor is it sufficient to gain access in and of itself. However, it does provide a cryptographic GUID which enforcement gateways use to both encrypt traffic and to consistently and uniquely refer to the device.

Mobile

Because Google seeks to make mobile a first-class platform, mobile must be able to accomplish the same tasks as other platforms and therefore requires the same levels of access. It turns out that deploying a tiered access model tends to be easier when it comes to mobile as compared to other platforms: mobile is typically characterized by a lack of legacy protocols and access methods, as almost all communications are exclusively HTTP-based. Android devices use cryptographically secured communications allowing identification of the device in the device inventory. Note that native applications are subject to the same authorization enforcement as resources accessed by a Web browser; this is because API endpoints also live behind proxies that are integrated with the Access Control Engine.

Legacy and Third-Party Platforms

We determined that legacy and third-party platforms need a broader set of access methods than we require for mobile devices. We support the tunneling of arbitrary TCP and UDP traffic via SSH tunnels and on-client SSL/TLS proxies. However, gateways only allow tunneled traffic that conforms with the policies laid out in the Access Control Engine. RADIUS [3] is one special case: it is also integrated with the device inventory, but it receives VLAN assignments rather than trust-tier eligibility semantics from the Trust Inferer. At network connection time, RADIUS dynamically sets the VLAN by referencing Trust Inferer assignments using the certificate presented for 802.1x as the device identifier.

Avoiding User Disruptions

One of our biggest challenges in deploying BeyondCorp was figuring out how to accomplish such a massive undertaking without disrupting users. In order to craft a strategy, we needed to identify existing workflows. From the existing workflows, we identified:

- Which workflows we could make compliant with an unprivileged network
- Which workflows either permitted more access than desirable or allowed users to circumvent restrictions that were already in place

To make these determinations, we followed a two-pronged approach. We developed a simulation pipeline that examined IP-level metadata, classified the traffic into services, and applied our proposed network security policy in our simulated environment. In addition, we translated the security policy into each platform's local firewall configuration language. While on the corporate network, this measurement allowed us to log traffic metadata destined for Google corporate services that would cease to function on an unprivileged network. We found some surprising results, such as services that had supposedly been decommissioned but were still running with no clear purpose.

After collecting this data, we worked with service owners to migrate their services to a BeyondCorp-enabled gateway. While some services were straightforward to migrate, others were more difficult and required policy exceptions. However, we made sure that all service owners were held accountable for exceptions by associating a programmatically enforced owner and expiration with each exception. As more services are updated and more users work for extended periods of time without exercising any exceptions, the users' devices can be assigned to an unprivileged VLAN. With this approach, users of noncompliant applications are not overly inconvenienced; the pressure is on the service providers and application developers to configure their services correctly.

The exceptions model has resulted in an increased level of complexity in the BeyondCorp ecosystem, and over time, the answer to "why was my access denied?" has become less obvious. Given the inventory data and real-time request data, we need to be able to ascertain why a specific request failed or succeeded at a specific point in time. The first layer of our approach in answering this question has been to craft communications to end users (warning of potential problems, and how to proceed with self-remediation or contact support) and to train IT Operations staff. We also developed a service that can analyze the Trust Inferer's

decision tree and chronological history of events affecting a device's trust tier assignment in order to propose steps for remediation. Some problems can be resolved by users themselves, without engaging support staff with elevated privileges. Users who have preserved another chain of trust are often able to self-remediate. For example, if a user believes his or her laptop has been improperly evaluated but still has a phone at a sufficient trust tier, we can forward the diagnosis request to the phone for evaluation.

Challenges and Lessons Learned

Data Quality and Correlation

Poor data quality in asset management can cause devices to unintentionally loose access to corporate resources. Typos, transposed identifiers, and missing information are all common occurrences. Such mistakes may happen when procurement teams receive asset shipments and add the assets to our systems, or may be due to errors in a manufacturer's workflow. Data quality problems also originate quite frequently during device repairs, when physical parts or components of a device are replaced or moved between devices. Such issues can corrupt device records in ways that are difficult to fix without manually inspecting the device. For example, a single device record might actually contain data for two unique devices, but automatically fixing and splitting the data may require physically reconciling the asset tags and motherboard serial numbers.

The most effective solutions in this arena have been to find local workflow improvements and automated input validation that can catch or mitigate human error at input time. Double-entry accounting helps, but doesn't catch all cases. However, the need for highly accurate inventory data in order to make correct trust evaluations forces a renewed focus on inventory data quality. Our data are the most accurate they've ever been, and this accuracy has had secondary security benefits. For example, the percentage of our fleet that is updated with the latest security patches has increased.

Sparse Data Sets

As mentioned previously, upstream data sources don't necessarily share overlapping device identifiers. To enumerate a few potential scenarios: new devices might have asset tags but no hostnames; the hard drive serial might be associated with different motherboard serials at different stages in the device lifecycle; or MAC addresses might collide. A reasonably small set of heuristics can correlate the majority of deltas from a subset of data sources. However, in order to drive accuracy closer to 100%, you need an extremely complex set of heuristics to account for a seemingly endless number of edge cases. A tiny fraction of devices with mismatched data can potentially lock hundreds or even thousands of employees out of applications they need to be productive. In order to mitigate such scenarios, we monitor and verify that a set of synthetic records in our production pipeline, crafted to verify trust evaluation paths, result in the expected trust tier results.

Pipeline Latency

Since the Device Inventory Service ingests data from several disparate data sources, each source requires a unique implementation. Sources that were developed in-house or are based on open source tools are generally straightforward to extend in order to asynchronously publish deltas to our existing pipeline. Other sources must be periodically polled, which requires striking a balance between frequency of polling and the resulting server load. Even though delivery to gateways typically takes less than a second, when polling is required, changes might take several minutes to register. In addition, pipeline processing can add latency of its own. Therefore, data propagation needs to be streamlined.

Communication

Fundamental changes to the security infrastructure can potentially adversely affect the productivity of the entire company's workforce. It's important to communicate the impact, symptoms, and available remediation options to users, but it can be difficult to find the balance between over-communication and under-communication. Under-communication results in surprised and confused users, inefficient remediation, and untenable operational load on the IT support staff. Over-communication is also problematic: change-resistant users tend to overestimate the impact of changes and attempt to seek unnecessary exemptions. Overly frequent communication can also inure users to potentially impactful changes. Finally, as Google's corporate infrastructure is evolving in many unrelated ways, it's easy for users to conflate access issues with other ongoing efforts, which also slows remediation efforts and increases the operational load on support staff.

Disaster Recovery

Since the composition of the BeyondCorp infrastructure is non-trivial, and a catastrophic failure could prevent even support staff from accessing the tools and systems needed for recovery, we built various fail-safes into the system. In addition to monitoring for potential or manifested unexpected changes in the assignment of trust tiers, we've leveraged some of our existing disaster recovery practices to help ensure that BeyondCorp will still function in the event of a catastrophic emergency. Our disaster recovery protocol relies on a minimal set of dependencies and allows an extremely small subset of privileged maintainers to replay an audit log of inventory changes in order to restore a previously known good state of device inventory state and trust evaluations. We also have the ability in an emergency

to push fine-grained changes to the access policy that allow maintainers to bootstrap a recovery process.

Next Steps

As with any large-scale effort, some of the challenges we faced in deploying BeyondCorp were anticipated while others were not. An increasing number of teams at Google are finding new and interesting ways to integrate with our systems, providing us with more detailed and layered protections against malicious actors. We believe that BeyondCorp has substantially improved the security posture of Google without sacrificing usability, and has provided a flexible infrastructure that will allow us to apply authorization decisions based on policy unencumbered by technological restrictions. While BeyondCorp has been quite successful with Google systems and at Google scale, its principles and processes are also within the reach of other organizations to deploy and improve upon.

Resources

[1] Architectural discussion of BeyondCorp: http://research.google.com/pubs/pub43231.html.

[2] Stagefright: https://en.wikipedia.org/wiki/Stagefright_(bug).

[3] RADIUS: https://en.wikipedia.org/wiki/RADIUS.

Writing for *;login:*

We are looking for people with personal experience and expertise who want to share their knowledge by writing. USENIX supports many conferences and workshops, and articles about topics related to any of these subject areas (system administration, SRE, file systems, storage, networking, distributed systems, operating systems, and security) are welcome. We will also publish opinion articles that are relevant to the computer sciences research community, as well as the system adminstrator and SRE communities.

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of *;login:*, is to submit a proposal to login@usenix.org.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, mini-paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, the limit for articles is about 3,000 words, and we avoid publishing articles longer than six pages. We suggest that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but has not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

The final version can be text/plain, text/html, text/markdown, LaTex, or Microsoft Word/Libre Office. Illustrations should be EPS if possible. Vector formats (TIFF, PNG, or JPG) are also acceptable, and should be a minimum of 1,200 pixels wide.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at www.usenix.org/publications/login/publication_schedule.

COPYRIGHT

You own the copyright to your work and grant USENIX first publication rights. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.