Watching the Watchers – Guarding the Keys to the Kingdom

Version 1.3

Released: April 26, 2012

Securosis, L.L.C. 515 E. Carefree Highway Suite #766 Phoenix, AZ 85085 **T** 602-412-3051

info@securosis.com www.securosis.com

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog, but has been enhanced, reviewed, and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Licensed by Xceedium

Xceedium, Inc., is the leading provider of Privileged Identity and Access Management solutions. The company's award-winning Xsuite platform protects organizations from the numerous threats that privileged users and trusted insiders pose to networks and data.

Xsuite is used globally by large enterprises and government agencies to meet stringent security and compliance requirements. Xsuite enables organizations to implement a zero trust security model — ensuring that privileged users are able to access only explicitly authorized systems and commands. The enterprise-class solution provides organizations with a unified platform for controlling, auditing, and continuously monitoring all privileged access. Xsuite also vaults and manages privileged credentials, fully records user sessions, and provides real-time alerting. The system extends security controls across IT infrastructure located on premise, in public or private cloud environments, or any combination thereof.

Xceedium's products are FIPS 140-2 Level 2 and Common Criteria EAL4+ certified. For more information, visit www.xceedium.com. For more on Xsuite, watch the Xsuite 2-Minute Explainer.

Copyright

Table of Contents

Introduction

It's about time we documented our research on privileged user management (PUM), which you overlook at your own risk. Administrators (those privileged users) have the keys to your kingdom. A sysadmin with malicious intent can mean a very bad day for you and your organization.

And no, this isn't just another recycled attempt to bring the *insider threat* back into vogue – much to the chagrin of the DLP vendors, who built their first wave of growth on it. First of all, privileged users (P-Users) don't necessarily need to be insiders. And most insiders have limited access and authorization entitlements, whereas administrators can basically give themselves whatever access want — that old privilege escalation thing. That's why we call this paper *Watching the Watchers* – because if not properly managed, administrators are *Above the Law*.

Business Imperatives Changing Privileges

We live in a brave new world of technology. What used to be within your site, in your data center, or running on your big iron, now may or may not be in any or all of those places. Even for stuff runs in your data center, you might not know exactly where and it may not be under your control. It might not be running on an operating system you understand. You might not control the pipes to the data. And you certainly can't tell business users and business partners that they need to go back to the old model, where you had visibility from the bare metal all the way to the data layer. Times have changed.

Even better, you might not even know who is responsible for managing those specific systems. With layers of virtualization abstracting just about all physical networks, storage, and servers, different folks assume responsibility for managing the pieces of what we call an *application.* Even the term 'application' is really a misnomer – applications can be almost anything, processing in numerous places, accessing data from anywhere, and presenting information to anyone anywhere. So let's start by defining a privileged user:

Privileged User: Anyone with admin (or `root`*) access to a device.*

By that definition every user is privileged on *some* device. But that's a bit broad, so we'll restrict our discussion, and this research, to users who manage critical devices – running applications, hosting databases, or pushing packets to the places they need to be. Sure, it's problematic if the P-user in charge of the receptionist's device (you know, the receptionist) is compromised. But it's much more serious if an administrator of your customer database host gets compromised.

Let's be a bit more specific about the business drivers and impact on privileged users:

- **Reduce Cost – Virtualization/Cloud:** Many organizations are under intense pressure to continue reducing costs wherever possible. That means embracing technologies such as virtualization to make better use of physical hardware and cloud computing to make due with less data center real estate. The impact of this driver is scale. Now you have a lot more things to manage, and they can be spun up and torn down at the click of a button or via script. Throw in the unbounded number of instances that can be run in the public cloud, and the only thing you can be sure of is a massive change management headache.

- **Reduce Cost – Outsource:** While data centers are virtualizing, organizations are contracting with (lower) cost management to do their (alleged) commodity work. You know, like managing databases and email. All kidding aside, it's common to see third parties managing wide swaths of organizations' IT infrastructure – giving nameless, faceless folks (perhaps on the other end of a SAML link) access to critical stuff.

- **Agility – New Apps:** If you think about a typical web app, it's more 'assembled' nowadays than built from the ground up. Parts may be yours, they could be pieces you got from someone else, and they might include data from somewhere else, integrated into your environment via a foreign API. It's hard to know what an *application* is nowadays. It's difficult to manage what we don't understand.

Anyone with access to manage a device that runs something important is a privileged user, and the rush to become more efficient and leveraged can result in errors, shortcuts, and general violation of good operational practices.

There are plenty more business drivers but you get the picture. Anyone with access to manage a device that runs something important (or is a component of something important) is a privileged user, and the change management issues inherent in this escalating complexity require administrators continue becoming more efficient and leveraged. Which can result in errors, shortcuts, and general violation of good operational practices. Let's look at some specific threats presented by these privileged users.

P-User Risk Assessment

We're old school. We still like to assess risk, or at least run through a quick mental exercise to figure out how many ways we can get killed. So let's do that with this explosion of devices managed by privileged users. Of course this isn't an exhaustive list – more a back-of-the-envelope exercise to uncover some of the biggest threats to our environment if privileged users are compromised. And while we are at it, let's define a new term, *PUpwnage*, for compromise of privileged user. It just rolls off the tongue, right?

1. **Compromised devices:** This one is obvious. If a privileged user is compromised (PUpwned), the attackers gain access to any device they manage, and the fun begins.

2. **Data leakage:** PUpwnage can result in any and all data being stolen from the devices they control.

3. **Create accounts:** PUpwnage allows attackers to create both user and admin accounts on devices, and to pivot through the environment moving from one compromised device to another – stealing data as they march along.
4. **Pollute applications and/or data:** PUpwnage also results in application attacks, such as changing code to break functionality, creating backdoors, deleting or changing data, and otherwise breaking your applications.
5. **Operational mayhem (shut down devices):** Yup, PUpwnage allows attackers to shut down devices, max out device utilization (effectively a denial of service), wipe disks, or pretty much anything else.
6. **Fiasco in the clouds:** If you run a bunch of stuff in the cloud and someone gains access to your cloud management console, they can do everything else we mentioned under operational mayhem. But a new attack, unique to the cloud, is an economic denial of service. Spinning up instances and consuming cloud resources, costs you real money and can continue until you either max out your credit or the provider shuts you down. Fun, right?

As enjoyable as it would be to come up with another 20 problems posed by compromised privileged users, someone very wise once told us "never sell past the close", so we'll assume you are convinced of the dangers of losing control of your P-Users. But we need to discuss the other issue, which is that P-Users aren't really that unique because many administrators share accounts and credentials. They justify this behavior with many reasonable sounding excuses such as 1) we need access if Admin1 gets hit by a bus, or 2) it's too time consuming to create admin credentials for each admin on each device. The end result isn't much different than a crack den: pretty messy, and at some point the cops show up to turn out the lights.

Compliance has remains a major driver of all privileged user management. Specifically, PCI-DSS demands unique IDs and that account credentials aren't shared. SOX requires separation of duties to ensure no one (not even administrators) has the ability to change an organization's financial controls without oversight. And we see the need to audit database administrators as a clear driver for Database Activity Monitoring. That's one specific use case for privileged user management; we prefer to focus on the security risks and complexity drivers for the technology, but we can't minimize the old reliables: compliance and audit.

P-Users aren't really unique because many administrators share accounts and credentials. The end result isn't much different than a crack den: pretty messy, and at some point the cops show up to turn out the lights.

To reiterate: privileged user management involves ensuring (only) the right folks access the right resources at the right times. And that you know exactly what they did. Easier said than done, but that's why we are writing this paper – to map out a specific set of activities that can help reduce the risk of P-User pwnage.

The Elephant in the Room

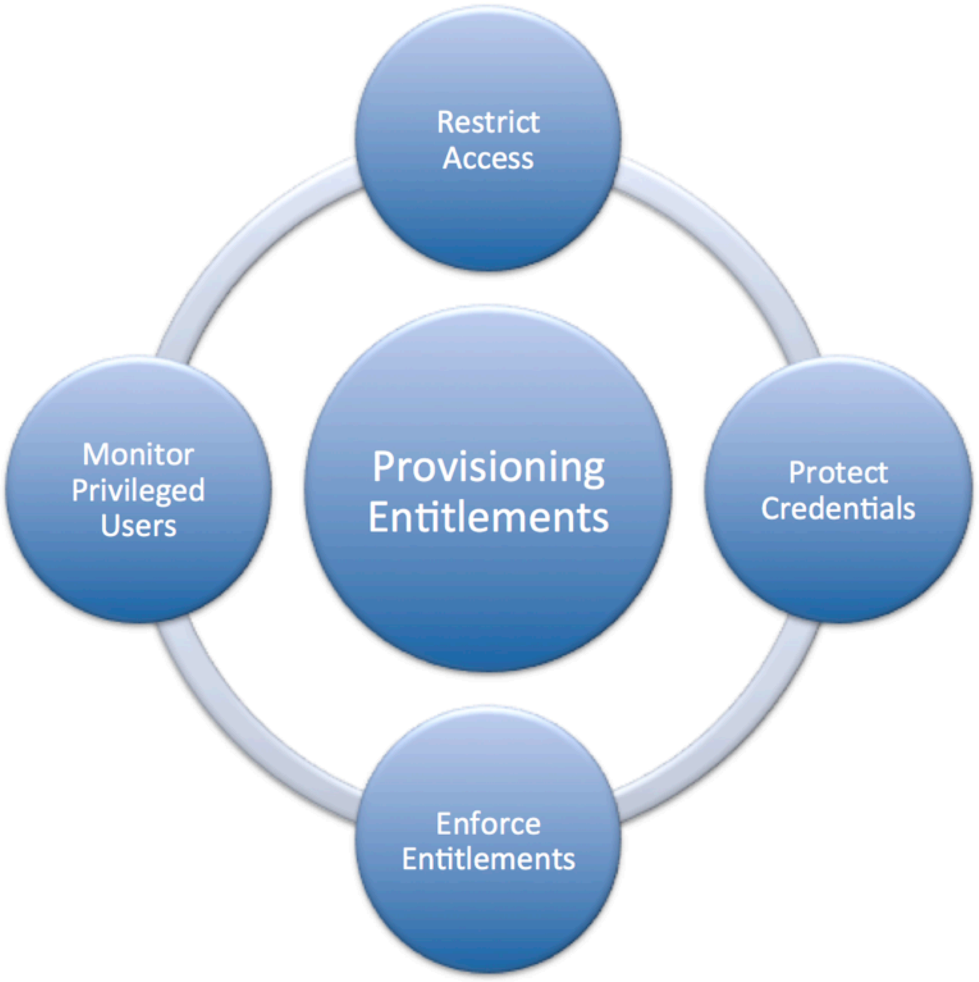
The cold hard reality is that most organizations don't formally manage privileged users. They may do some identity management, but that's for everyone. Given the potential issues, why doesn't *everyone* watch the watchers? Inevitably a higher priority project, or a new device with shinier lights, shows up to divert attention and budget away from this non-revenue-generating and non-cost-saving activity. Until a rogue admin locks you out of your data center or your forensics guys find the back door into your logistics system, anyway. But without throwing around a bunch of FUD we realize there needs to be a logical, phased approach to solving the problem. Big bangs don't work with large organizations with lots to do. So this paper will lay out a lifecycle for managing your privileged users.

We will go beyond the identity management aspects (provisioning, entitlements, and authentication) of managing users to also focus on the operational requirements and issues. This means keeping admins (and other folks) from establishing management sessions with devices they aren't authorized to manage. If they *are* authorized you need to make sure they can't share or otherwise misuse credentials. And finally you want to audit what they do on the devices – critical both for forensics after an attack and to substantiate controls for compliance.

The Privileged User Lifecycle

As we described above, organizations can't afford to ignore the issue of privileged users (P-Users) any more. A compromised P-user can cause all sorts of damage and so needs to be actively managed. Let's now talk about solutions. Most analysts favor models to describe things, and we call ours the *Privileged User Lifecycle*.

But pretty as the lifecycle diagram is, first let's scope it to define beginning and ending points. Our lifecycle starts when the privileged user receives escalated privileges, and ends when they are no longer privileged or leave the organization, whichever comes first. Here is the whole lifecycle:



Provisioning Entitlements

The Privileged User Management lifecycle starts when you determine someone gets escalated privileges. That means you need both control and an audit trail for granting these entitlements. Identity Management is a science all by itself, so this research cannot tackle it in depth – we will just point out the connections between (de-)provisioning escalated privileges and the beginning and end of the lifecycle. These privileged users have the keys to the kingdom so you need tight controls over the provisioning process, including separation of duties and a defined workflow with adequate authorization.

> Privileged users have the keys to the kingdom you need tight controls over the provisioning process, including separation of duties and a defined workflow which includes adequate authorization.

Identity management is repository-centric, so any controls you implement throughout the lifecycle need native integration with the user repository. It doesn't work well to store user credentials multiple times in multiple places.

Another aspect of the provisioning process involves defining the roles and entitlements for each administrator, or more likely for groups of administrators. We favor a *default deny* model, which basically denies any management capabilities to administrators, assigns capabilities by explicit authorization to manage device(s), and defines what they can do on each specific device. Although the technology to enforce entitlements can be complicated, as we will discuss later, defining the roles and assigning administrators to the proper groups can be even more challenging. This typically involves gaining significant consensus among the operations team, which is always fun but also critical for P-User management.

Now we get to the fun stuff: actively managing what specific administrators can do. In order to gain administrative rights to a device an attacker (or rogue administrator) needs access, entitlements, and credentials. So the next aspect of our lifecycle address these issues.

Restrict Access

Let's first tackle restricting access to devices. The key is to allow administrators to establish management sessions only to devices they are entitled to manage. Managing any other device should be blocked to that P-User. That's the meaning of default deny in this context. Network isolation is one of the oldest defense tactics you know. If a P-User can't logically get to a device, they can't manage it nefariously.

There are quite a few ways to isolate devices, both physically and logically, including proxy gateways and device-based agents; we will discuss a number of these tactics later. When restricting access you also need to factor in authentication, as logging into a proxy gateway and/or managing particularly sensitive devices should require multiple factors.

Obviously integrating private and public cloud instances into the P-User management environment requires different tactics, as you don't necessarily have control over the network to govern access. But the cloud is too enticing to simply avoid. We will also delve into tactics to restrict access to cloud-only and hybrid environments later.

Protect Credentials

Once a P-User has network access to a device, they still need credentials to manage it, so administrator credentials need appropriate protection. The next step in the lifecycle typically involves setting up a password vault to store administrator credentials and provide a system for one-time passwords. There are a number of architectural decisions involved in vaulting privileged user passwords that impact the other controls in place: restricting access and enforcing entitlements.

Enforce Entitlements

If an administrator has access and the credentials, the final aspect of controls involve determining what they can do. Many organizations opt for a *carte blanche* policy, providing `root` access and allowing P-Users to do whatever they want. Others take a finer-grained approach, defining the specific commands the P-User can perform on any class of device. For instance, you might allow the administrator to update the device or install software/applications, but not delete a logical volume or load an application. This all depends on how much granularity you use when provisioning the entitlements.

Technically, this approach requires some kind of agent capability on the managed device, or running sessions through a proxy gateway which can intercept and block commands as necessary. We will discuss architectures later when we dig into this control.

Privileged User Monitoring

Finally, keep a close eye on what all the P-Users do when they access devices. That's why we called this paper "Watching the Watchers" — the lifecycle doesn't end after implementing the controls. Privileged User Monitoring can mean a number of different things, from collecting detailed audit logs on every transaction, to actually capturing video of each session. There are multiple benefits to detailed monitoring, including forensics and compliance.

We should also mention the deterrent benefit of privileged user monitoring. Human nature dictates that people are more diligent when they know someone is watching. So Rich can be happy that human nature hasn't changed. Yet. When administrators know they are being watched they are more likely to behave properly – not just from a security standpoint but also from an operational standpoint.

No Panacea

Of course the privileged user lifecycle is not a panacea. A determined attacker will find a path to compromise your systems, regardless of how tightly you manage privileged users. No control is foolproof, and there are ways to gain access to protected devices and to defeat password vaults. So we will examine the weaknesses in each of these tactics later, as well. As with everything else in security, you aren't looking for perfection – but to make it a bit harder for attackers to gain `root` on your critical devices.

Restrict Access

We recommend an initial focus on *Restricting Access*, mostly because it reduces your attack surface, and the earlier you can eliminate risk to your key servers, the better. You achieve the general objective of restricting access by implementing controls to ensure administrators only access devices they have authorization to manage. There are a few ways to handle restriction:

> You achieve the general objective of restricting access by implementing controls to ensure administrators only access devices they have authorization to manage.

1. **Device-centricity (Status Quo):** Far too many organizations rely on their existing controls, which include authentication and other server-based access control mechanisms. But given the number of ways to escalate privileges or bypass the operating system defenses with known or unknown vulnerabilities, it's clear that the status quo doesn't cut it.
2. **Network Isolation:** Tried and true network segmentation approaches enable you to isolate devices (typically by group) and only allow authorized administrators access to the networks on which they reside.
3. **Privileged User Management (PUM) Proxy Gateway:** This entails routing all management communications through a privileged user management proxy server or service, which enforces access policies. The devices only accept management connections from the proxy server, and do not allow direct management access.

There are benefits and issues with each approach, so ultimately you need to choose an appropriate compromise. Let's dig into each approach and highlight what's good and what's not so good.

Device-centricity (Status Quo)

There are really two levels within this category; the first is common authentication, which in this context is not effectively *restricting access*. Obviously you could do a bit to make the authentication more difficult, including strong passwords and/or multi-factor authentication. You would also integrate with an existing identity management platform (IDM) to keep entitlements current. But ultimately you are relying on credentials as a way to keep unauthorized folks from managing your critical devices. And basic credentials can be defeated.

The other level under this category is server access control capabilities, which are fairly mature. This involves loading an agent onto each managed device and enforcing the access policy on the device. The agent-based approach offers fairly solid security – the risk shifts to compromise of the agent. Of course there is management overhead to distribution and management of the agents, as well as the additional computational load on the device imposed by the agent.

But any device-based approach is in opposition to one of our core philosophies: "If you can't see it, it's much harder to compromise." Device-centric access approaches don't affect visibility (to attackers) at all. This is suboptimal, because in the real world new vulnerabilities appear every month on all operating systems – and many of them can be exploited via zero-day attacks. Those attacks provide "back doors" into servers, giving attackers control without requiring legitimate credentials – regardless of agentry on the device. Any device-based method fails if the device is rooted somehow.

Network Isolation

This entails using network-layer technologies such as virtual LANs (VLANs) and network access control (NAC) to isolate devices and restrict access based on who can connect to specific protected networks. The good news is that many organizations (especially those subject to PCI) have already implemented some level of segmentation. It's just a matter of building another *enclave,* or trust zone, for each group of servers to protect.

Segmentation requires the attacker to know exactly what they are looking for and where it resides, and to have a mechanism for gaining access to the protected segment.

As mentioned earlier, it's much harder to break into something you can't see. Segmentation requires the attacker to know exactly what they are looking for and where it resides, and to have a mechanism for gaining access to the protected segment. Of course this is *possible* – there have been ways to defeat VLANs for years – but vendors have closed most of the very easy loopholes.

More problematic to us is that this relies on the networking operations team. Managing entitlements and keeping devices on the proper segments in a dynamic environment such as your data center, can be challenging. It is definitely possible, but it's also difficult, and it puts direct responsibility for access restriction in the hands of the network ops team. That can and does work for some organizations, but organizationally it is complicated and somewhat fragile.

The other serious complication is cloud computing – for both private and public clouds. The cloud is key and everybody is jumping on the bandwagon, but unfortunately it largely removes visibility at the physical layer. If you don't really know where specific instances are running this approach becomes difficult or completely unworkable. We will discuss this in detail later when we discuss the cloud in general.

PUM Proxy

This approach routes all management traffic through a proxy gateway (PUM proxy). Administrators authenticate to the proxy, hopefully using strong authentication. The authenticated administrator gets a view of the devices they can manage and establishes a management session to the desired device. Another possible layer of security involves a lightweight agent on every managed devices to handle the handshake and mutual authentication with the PUM proxy, and to block management connections from unauthorized sources.

This approach is familiar to anyone who has managed cloud computing resources via vCenter (in VMware land) or a cloud console such as the one for Amazon Web Services. You log in and see the devices/instances you can manage, and proceed accordingly.

This satisfies our preference for providing visibility to only those devices that can legitimately be managed. It also provides significant control over granular administrative functions, as commands can be blocked in real time (it is a man in the middle, after all). Another side benefit is the *deterrent effect:* administrators *know* all their activity is running through a central device and typically heavily monitored (as we will discuss later).

But any proxy presents issues, including a possible single point of failure, and additional latency for management sessions. Some additional design and architecture work is required to ensure high availability and reasonable efficiency. It's a bad day for the security team if they prevent ops from doing their jobs. And periodic latency testing is called for to make sure the proxy doesn't impair productivity. Finally, as with virtualization and cloud consoles, if you own the proxy server, you own **everything** in the environment. So the proxy's security is paramount.

The PUM proxy satisfies our preference for providing visibility to only those devices that can legitimately be managed. It also provides significant control over granular administrative functions, as commands can be blocked in real time (it is a man in the middle, after all).

Each of these approaches is best in different environments, and each entails its own compromises. For those just starting to experiment with privileged user management, a PUM proxy is typically the path of least resistance for getting started. It's a question of what works best for you, based on the sophistication of required controls and IT culture.

Protect Credentials

The next part of the lifecycle addresses protection of the keys or credentials of these privileged users (P-Users), to prevent them from falling into the wrong hands. The best access and entitlement security controls fail if someone can impersonate a P-User. But the worst risk isn't even compromised credentials — it's not having unique credentials in the first place. You must have seen the old admin password sharing scheme, right? It was used, mostly out of necessity, many moons ago. Administrators needed access to the devices they managed. But at times they needed help, so they asked a buddy to take care of something and just gave him/her the credentials. What could possibly go wrong?

We have already mentioned that shared administrative credentials open Pandora's Box. Once the credentials are in circulation you can't get them back – which is a problem when an admin leaves the company or no longer has those privileges. You can't deprovision shared credentials, so you need to change them.

PCI, as the low bar for security (just ask Global Payments), recognizes the issues with sharing IDs, so Requirement 8 is all about making sure anyone with access to protected data uses a unique ID and that their use is audited – so you can attribute every action to a particular user.

> You can't deprovision shared credentials, so you need to change them.

But that's not all! (in our best infomercial voice). What about the risk that some endpoints could be compromised? Even administrative endpoints — which would make sending admin credentials to that endpoint unsafe. And what happens when developers hard-code credentials into applications? Why go through the hassle of secure coding – just embed the password right into the application! That password never changes anyway, so what's the risk? We need to protect credentials just as much as whatever they control.

Credential Lockdown

How can we protect these credentials? Locking them away in a *vault* satisfies many of the requirements. First, if the credentials are stored in a vault, it is harder for admins to share them. We won't put the cart before the horse, but a vault makes it fairly easy (and transparent) to change passwords after every access, eliminating the sticky-note-under-keyboard risk.

Going through the vault for every administrative credential access creates an audit trail of who used which credentials and when, and often which specific devices they were managing. That kind of stuff makes auditors happy.

Depending on the deployment of the vault, the administrator may never even see the credentials, as they can be automatically entered on the server under the proxy approach to restricting access. This also provides single sign-on to all managed devices, as the administrator authenticates (presumably using multiple factors) to the proxy, which interfaces directly to the vault again, transparently to the user. So even an administration device teeming with malware cannot leak critical credentials.

Similarly, an application can make a call to the vault, rather than hard-coding credentials into the app. Yes, the credentials still end up on the application server, but that's still much better than hard-coding the password. So are you sold yet? If you worry about credentials being access and misused, a password vault offers a good mechanism for protecting them.

Define Policies

As with most things in security, using a vault requires both technology and process. We will tackle the process first, because without a good process even the best technology has no chance. Before you implement anything you need to define the rules of (credential) engagement. Start by answering some questions.

1. **Which systems and devices need to be involved in the password management system?** This may include servers (physical and/or virtual), network and security devices, infrastructure services (DNS, directory, mail, etc.), databases, and/or applications. Ideally your vault will natively support most of your targets, but broad protection is likely to require some integration on your end. Make sure any solution you look at has some kind of API to facilitate this integration.

2. **How does each target use the vault?** Then you need to decide who can access each target (likely by group), how long they are allowed to use the credentials and manage the devices, and whether they need to present additional authentication factors for access. You will also define whether multiple administrators can access managed devices simultaneously and whether to change the password after each check-in/check-out cycle. Finally, you may need to support external administrators (for third party management or business partner integration), so keep that in mind as you work through these decisions.

3. **What kind of administrator experience makes sense?** Then you need to figure out the P-User interaction with the system. Will it be via a proxy login, where the user never sees the credentials, or will there be a secure agent on the device to receive and protect the credentials? Figure out how the vault supports application-to-database and application-to-application interaction, as those are different than supporting human admins. You will also want to specify which activities are audited and how long audit logs are kept.

Securing the Vault

If you are putting the keys to the kingdom in this vault, make sure it's secure. You probably will not bring a product in and set your application pen-test ninjas loose on it, so you are more likely to rely on what we call the *sniff test.* Ask questions to see whether the vendor has done their homework to protect the vault.

You should understand the security architecture of the vault. You need to know how they protect things.

You should understand the security architecture of the vault. Yes, you may have to sign a non-disclosure agreement to see the details, but it's worth it. You need to know how they protect things. Discuss the threat model(s) the vendor uses to implement that security architecture. Make sure they didn't miss any obvious attack vectors. You also need to poke around their development process a bit to make sure they have a proper SDLC and actually *test* for security defects before shipping. Don't laugh – it's not funny. You would be shocked at how many "security companies" don't do this.

Not that you need to be a protocol ninja, but you need to understand how the communications happen, in case the vault ships the protected credential to a compromised endpoint or app. What protocols are used, what does the endpoint/app agent look like, and how does the agent protect the credentials? In a perfect world the credentials could never make it to a compromised device. But in the real world these are questions you need to ask.

It's also good to know whether the vendor has contracted with an application penetration testing outfit to try to break their web interface, especially if you plan to use the vault as a proxy providing single sign-on. If you break the interface you break the system. This is another area often overlooked by security vendors.

Discuss deployment architectures with both the vendor and some large reference customers. Single points of failure are a very real concern with vaults. And if the vault fails, your IT ops group is out of business. So design the deployment to ensure you have recoverability.

With all that done, you have a vault you could be comfortable putting gold bullion in.

Enforce Entitlements

If you have implemented the first two aspects of the *Privileged User Lifecycle*, any administrator managing a device is authorized to be there and uses strong credentials. But what happens when they get there? Do they get free reign? Should you just give them `root` or full `Administrator` rights and have done with it? What could possibly go wrong with that?

Clearly you should make sure administrators only perform authorized functions on managed devices. This protects against a couple scenarios you probably need to worry about:

1. **Insider Threat:** A privileged user is the ultimate insider, as he/she has the skills and knowledge to compromise a system and take what they want, cover their tracks, etc. So it makes sense to provide a bit more specificity over what admins can do and block them from everything else.
2. **Separation of Duties:** Related to the Insider Threat, optimally you should make sure no one person has the ability to take down your environment. So you can logically separate duties such that one group can manage the servers but not the storage. Or one admin can provision a new server but can't move data onto it.
3. **Compromised Endpoints:** You also can't assume any endpoint isn't compromised. So even an authenticated and authorized user may not be who you think they are. You can protect yourself by restricting what the administrator can do. So even in the worst case, where an intruder is in your system as an admin, they can't wreck everything.

Smaller organizations may lack the resources to define administrator roles with real granularity. But the more that larger enterprises can restrict administrators to particular functions, the harder it gets for a bad apple to take everything down.

Either adopt a whitelist approach: defining legitimate commands and blocking everything else; or block specific commands (a blacklist).

Policy Granularity

You need to define roles and responsibilities – what administrators can and can't do – with adequate granularity. We won't go into detail on the process of setting policies, but you will either adopt a whitelist approach: defining legitimate commands and blocking everything else; or block specific commands (a blacklist), such as restricting folks in the network admin group from deleting or snapshotting volumes in the data center.

Depending on your needs you might also define far more granular polices, similar to the policy options available for controlling access to the password vault. For example you might specify that a sysadmin can only add user accounts to devices during business hours, but can add and remove volumes at any time. Or you could define specific types of commands authorized to flow from an application to the back-end database to prevent unauthorized data dumps.

But granularly brings complexity. In a rapidly changing environment it can be hard to nail down a legitimate set of allowable actions for specific administrators. So overdoing granularity has consequences similar to those with application whitelisting. The higher up the application stack you go, the more integration is required, as homegrown and highly customized applications need to be manually integrated into the privileged user management system.

Location, Location, Location

As much fun as it is to sit around and set up policies, the reality is that nothing is protected until the entitlements are enforced. There are two main approaches to actually enforcing entitlements. The first involves implementing a proxy in between the admin and the system, which acts as a *man in the middle* to interpret and then either allow or block each command. Alternatively, entitlements can be enforced on devices by agents that intercept commands and enforce policy locally.

To enforce entitlements you can implement a proxy between the admin and the system, which acts as a *man in the middle,* or through an endpoint agent that intercepts commands and enforces policy locally.

We aren't religious about either approach — each has pros and cons. Specifically, the proxy implementation is simpler – you don't need to install agents on every device, so you don't have to worry about OS compatibility (as long as the command syntax remains consistent) or deal with incompatibilities every time an underlying OS is updated. Another advantage is that unauthorized commands are blocked before reaching the managed device, so even if the attacker has elevated privileges, management commands can only come through the proxy. On the other hand the proxy serves as a choke point, which may introduce a single point of failure.

Similarly, an agent-based approach offers advantages such as preventing attackers from accessing a backdoor on devices by defeating the proxy or gaining physical access to the devices. The agent runs *on* each device, so even being at the keyboard doesn't kill it. But agents require management and consume processing resources on the managed systems. Pick the approach that makes the most sense for your environment, culture, and operational capabilities.

Monitor Privileged Users

If you thought you could avoid the auditors forever… well, not so much. The lifecycle ends with a traditional audit because verifying what administrators do with their privileges is just as important as the other steps. Admittedly, some organizations have a cultural issue with granular user monitoring because they actually want to *trust* their employees. Silly organizations, right? But in this case there is no monitoring slippery slope – we aren't talking about recording an employee's personal Facebook interactions or checking out pictures of Grandma. We're talking about capturing what an administrator has done on each specific device.

Before we get into the *how* of privileged user monitoring, let's look at *why* you would monitor admins. There are two main reasons:

We aren't talking about recording an employee's personal Facebook interactions or checking out pictures of Grandma. We're talking about capturing what an administrator has done on each specific device.

- **Forensics:** In case of a breach, you need to know what happened on the device, quickly. A detailed record of what an administrator did on a device can be instrumental to putting the pieces together – especially in the event of an inside job. Of course privileged user monitoring is not a silver bullet for forensics – there are a zillion other ways to get compromised – but if the breach begins with administrator activity you at least have a record of what happened, and the proverbial smoking gun.
- **Audit:** Another use is to make your auditor happy. Imagine the difference between showing the auditor a policy saying how you do things, and showing a screen capture of an account being provisioned or a change being committed. Monitoring logs are powerful demonstrations that the controls are in place.

Sold? Good, but how do you move from concept to reality? You have a couple options, including:

1. **SIEM/Log Management:** As part of your other compliance efforts, you likely send most events from sensitive devices to a central aggregation point. That SIEM/Log Management work can also be used to monitor privileged users. By setting up some reports and correlation rules for administrator activity you can effectively figure out what administrators are doing. By the way, this is a major use cases for SIEM and log management.

2. **Configuration Management:** A similar approach is to pull data out of a configuration management platform which tracks changes across managed devices. An advantage of PUM over pure configuration management or SIEM/Log Management is the ability to go beyond monitoring to actually block unauthorized changes.

Screen Capture

If a picture is worth a thousand words, how much would you say a video is worth? One advantage of routing administrative sessions through a PUM proxy is the ability to capture *exactly* what admins are doing on every device. With a video capture of the session and the associated keystrokes, there can be no question of intent – no *inference* of what actually happened. You'll *know* what happened – you just need to watch the playback.

> One advantage of routing your administrative sessions through a proxy is the ability to capture *exactly* what admins are doing on every device. With a video capture of the session and the associated keystrokes, there can be no question of administrator intent.

For screen capture you can deploy an agent on the managed device or you could route sessions through a proxy. We started discussing the P-User Lifecycle by focusing on how to restrict access to sensitive devices. After discussing a number of options we explained why proxies make a lot of sense for making sure only the *right* administrators access the *correct* devices at the *right* times. So it's appropriate that we come full circle and end our lifecycle discussion back where we started.

Let's look at performance and scale first. Video is fairly compute intensive and consumes a tremendous amount of storage. The good news is that an administrative session doesn't require HD quality to catch a bad apple red-handed. So significant compression is feasible and can save a significant chunk of storage – whether you capture with an agent or through a proxy. But there is a major difference in device impact between these approaches. An agent takes resources for screen capture from the managed device, which impacts its performance – probably significantly. With a proxy, the resources are consumed by the proxy server rather than the managed device.

The other issue is the security of the video – ensuring there is no tampering with the capture. Either way you can protect the video with secure storage and/or other means of making tampering evident, such as cryptographic hashing. The main question is how you get the video *into* secure storage. An agent needs a secure transport to the storage. With a proxy the storage can be integrated into (or very close to) the proxy device.

We believe a proxy-based approach to monitoring privileged users makes the most sense in most cases, but there are certainly cases where an agent could suffice.

Clouds Rolling In

As much as we enjoy playing masters of the obvious, we don't really need to discuss the move to cloud computing. It's happening. It's disruptive. Blah blah blah. People love to quibble about the details but it's obvious to everyone. And of course when the computation and storage behind your essential IT services may not reside in a facility under your control things change a bit. The idea of a privileged user evolves in the cloud context, by adding another layer of abstraction from the cloud management environment. So regardless of your current level of cloud computing adoption, you need to factor the cloud into your PUM initiative.

Or do you? Let's play a little devil's advocate here. When you think about it, doesn't cloud computing just give you the ability to do a lot more, faster? You know, provision new systems, roll out new applications, reconfigure networks, and add storage. Handling these functions in the cloud means you no longer have to deal with the pesky IT ops group asking inconvenient questions like "Why?"

You still have the same operating systems running as guests in public and/or private clouds, but with a greatly improved ability to spin up machines, faster than ever before. If you are able to provision and manage the entitlements of these new servers, it's all good, right? In the abstract, yes. But the same old same old doesn't work nearly as well in the new regime. We respect the noble ostrich, but unfortunately burying your head in the sand doesn't really remove the need to think about privileged users on cloud computing resources. So let's walk through some ways cloud computing differs fundamentally from the classical world of on-premise physical servers.

The same old same old doesn't work nearly as well in the new regime. We respect the noble ostrich, but unfortunately burying your head in the sand doesn't really remove the need to think about privileged users on cloud computing resources.

Cloud Risks

First of all, any cloud initiative adds another layer of management abstraction. You manage cloud resources though either a virtualization console (such as vCenter or XenCenter) or a public cloud management interface. This means a new set of privileged users and entitlements require management. Additionally, this cloud stuff is (relatively) new, and management capability continues to evolve rapidly, but it hasn't yet caught up with the tools and processes for management of physical servers on a local physical network – and that immaturity poses a risk.

For example, without entitlements properly configured, anyone with access to the cloud console can create and tear down *any* instance under that account. Or they can change access keys, add access or entitlements, change permissions, etc. – *for the entire virtual data center*. Again, this doesn't mean you shouldn't proceed and take full advantage of cloud initiatives. But take care to avoid unintended consequences stemming from the flexibility and abstraction of the cloud.

We also face a number of new risks driven by the flexibility of provisioning new computing resources. Any privileged user can spin up a new instance, which might not include proper agentry and instrumentation to plug into the cloud management environment. You don't have the same coarse control of network access as before, so it's easier for new (virtual) servers to pop up, which means it's also easier to be exposed accidentally. Management and security largely need to be implemented *within* instances – you cannot rely on the cloud infrastructure to provide them. So cloud consoles absolutely demand suitable protection – *at least* as much as the most important server under their control.

You will want to take a similar lifecycle approach to protecting the cloud console as you do with traditional devices.

The Lifecycle in the Clouds

To revisit our earlier points for the new context, the Privileged User Lifecycle involves restricting access, protecting credentials, enforcing entitlements, and monitoring P-user activity – but what does that look like in the cloud context?

Restrict Access (Cloud)

As in the physical world, you have a few options for restricting access to sensitive devices, which vary dramatically between private and public clouds. You can implement access controls within the network, on the devices themselves via agents, or by running all connections through a proxy and only allowing management connections from the proxy.

- **Private cloud console:** Those tactics generally work but there are a few caveats. Network access control gets a lot more complicated due to the inherent abstraction of the cloud. Agentry requires pre-authorized instances which include properly configured software. A proxy requires an additional agent of some kind on each instance, to restrict management connections to the proxy. That actually matches the traditional data center – but now it must be tightly integrated with the cloud console. As instances come and go, knowing which instances are running and which policy groups each instance requires becomes the challenge. To fill this gap third party cloud management software providers are emerging to add finer-grained access control to private clouds.

- **Public cloud console:** Restricting network access is obviously a non-starter in a public cloud. Fortunately you can set up specific security groups to restrict traffic and have some granularity on which IP addresses and protocols can access the instances, which would be fine in a shared administrator context. But you do not have the ability to restrict access to specific users on specific devices (as required by most compliance mandates) at the network layer, because you have little control over the network. That leaves agentry on the instances, but with little ability to stop unauthorized parties from accessing instances. Another less viable option is a proxy, but you can't really restrict access *per se* – the console literally resides on the Internet. To protect instances in a public cloud you need to insert protections into other segments of the lifecycle.

Fortunately we see some innovation in cloud management, including the ability to *manage on demand*. This means the protocols used to manage instances (usually via `ssh` on Linux instances) are blocked by default. Only when management is required does the cloud console open up management ports via policy, and only for authorized users at specified times. That approach addresses a number of the challenges of always-on and always-accessible cloud instances, and represents a promising model for cloud management.

Protect Credentials (Cloud)

To think about protecting credentials for cloud computing resources we use an expanded concept of *credentials*. We now need to worry about three types of credentials:

1. Credentials for the cloud console(s)
2. Credentials for instances
3. Credentials for API access

The real question is which of these groups can (and should) be stored in a password vault as described in the *Protect Credentials* section. Optimally the answer is yes: everything goes in the vault. But it's rarely so simple. The most straightforward credential to store in the vault is the console credential. In a private cloud, access to the vault is no different than access to traditional data center devices, so that's not an issue.

It's a bit more complicated with a public cloud, as a device-based approach would require you to log into the proxy or have the credentials transferred to an agent on the device. Either way it's achievable and recommended given the power of the cloud console. Another option is to rely on federation to allow existing trusted credentials to be used for federated access to the cloud console. For example, the Identity Management features of Amazon AWS support federation, so you can use existing credentials for access to the console.

Accessing cloud instances through a vault is possible but requires some work. Basically, to start an instance, you need to have the credentials sent to and stored in the vault. So the instance needs a way to bootstrap the credential generation and storage process – which clearly demands automation. The proxy (or agent on an administrator's device) manages the keys for access, as well as the administrative credentials. This may add some latency – especially in a public cloud – and so needs to be weighed against the security risks of sharing `root` access.

Finally, what about API access? This would require a similar capability to the current application support in password vaulting products. Credentials would be stored in the vault and the application would call the vault to get the credential, which would then be securely transferred to the application, which would use the credential. For the cloud management API the automation would need to call the vault to get the API credentials and then use them accordingly. So this requires a bunch of integration by either you or the vault vendor. We are not currently aware of any vault vendor that has done this integration with a cloud services API yet, but in light of the popularity and adoption curve of the cloud, we expect this capability soon.

Enforce Entitlements (Cloud)

Once the administrator gains access to the instance, how can you enforce finer grained controls? Until you refine entitlements anyone with access to the cloud console can do pretty much anything. Some of the more mature and robust public cloud providers do offer finer grained control, but it's largely a manual process. Private clouds are still maturing rapidly, meaning many capabilities simply aren't there yet, including limitations on administrative capabilities such as those offered by virtualization platform vendors. Thus the emergence of third-party offerings to fill this gap for both private and public clouds. Of course we expect these finer-grained controls to gradually be integrated into cloud management consoles. But for now to get real control over who can do what within your cloud environment you need a third-party offering.

For command blocking, to control what administrators can do to instances – as described in the *Enforce Entitlements* section – the approach is the same as in the physical world. You need to either install an agent on the instance to pull the policy from the management console, or route management traffic through a proxy that blocks unauthorized commands. Once again the cloud complicates things a bit – routing management traffic through a central point adds latency – but again, that may be the price of security.

Monitor Privileged Users (Cloud)

As with enforcing entitlements, privileged user monitoring in the cloud involves the same decision points as in the physical world. Routing through the proxy to record sessions entails the same tradeoffs: latency and possible inconvenience, traded off for security. And recording via a device agent can again exact a performance toll on the instance.

But the real tradeoff is about storing logs and sessions. Do you aggregate in the cloud or send back to a central location, perhaps increasing latency or generating additional network traffic? Of course there are storage security and integrity requirements for any cloud-based repository, which may favor a centralized on-premise option that can be better controlled.

The Need for Consistency

Regardless of how you decide to manage privileged users for cloud instances, we cannot stress the importance of consistency enough. The eventual goal is to set one policy for privileged users and enforce it consistently *everywhere*. Right now, due to the immaturity of cloud computing, that requires multiple tools for the various steps in the PUM lifecycle. Over time we expect better integration between PUM offerings focused on traditional data centers and those built for the cloud. But beware the vendor tendency to *cloud-wash* their offerings: they have an unfortunate tendency to claim superior support for cloud computing when *actually* offering the exact same product running on cloud instances.

Look for integration with cloud consoles and APIs as a start. Management tools without such support miss an entire area of exposure.

Integration

In today's enterprise IT environment nothing stands alone – not in the management stack anyway – so privileged user management needs to play nicely with the other management tools. Various levels of integration are required, as some functions need to be attached at the hip, while others can be mere acquaintances.

Identity Integration

Given that the 'U' in PUM stands for *user,* clearly Identity infrastructure is one of the categories that needs to be tightly coupled. What does that mean? We described the provisioning/entitlements requirement in the Privileged User Lifecycle. But Identity is a discipline itself, so we cannot cover it in real depth in this paper.

Given that the 'U' in PUM stands for *user,* clearly Identity infrastructure is one of the categories that needs to be tightly coupled.

In terms of integration, your PUM environment needs to natively support your enterprise directory. It doesn't really work to have multiple authoritative sources for users. Privileged users are, by definition, a subset of the user base, so they reside in the main user directory. This is critical, for both provisioning new users and deprovisioning those who no longer need specific entitlements. Again, the PUM environment handles enforcing entitlements, but the groupings of administrators are stored in the enterprise directory.

Another requirement for identity integration is support for two-factor authentication. PUM protects the keys to the kingdom, so if a proxy gateway is part of your PUM installation, it's *essential* to ensure a connecting privileged user is *actually* the real user. That requires some kind of multiple-factor authentication to protect against an administrator's device being compromised and an attacker thereby gaining access to the PUM console. That would be a bad day. We don't have any favorites in terms of stronger authentication methods, though we note that most organizations opt for tried-and-true hard tokens.

Management Infrastructure

Another area of integration is the enterprise IT management stack. You know, the tools that manage data center and network operations. This may include configuration, patching, and performance management. The integration is mostly about pushing alert to an ops console. For instance, if the PUM portal is under brute force password attack, you probably want to notify ops folks to investigate. The PUM infrastructure also represents devices, so there will be some device health information that could be useful to ops. If a device goes down or an agent fails, alerts should be sent over to the ops console.

Finally, you will want some kind of help desk integration. Some ops tickets may require access to the PUM console, so being able to address a ticket and close it out directly in the PUM environment can streamline operations.

Monitoring Infrastructure

The last area of integration is monitoring infrastructure. Yes, your SIEM/Log Management platform should be the target for any auditable event in the PUM environment. After all, a best practice for log management is to isolate the logs on a different device to ensure log records aren't tampered with in the event of a compromise. Frankly, if your PUM proxy is compromised you have bigger problems than log isolation, but you should still exercise care in protecting the integrity of the log files, and perhaps they can help you address those larger issues.

Sending events over to the SIEM also helps provide more depth for user activity monitoring. Obviously a key aspect of PUM is *privileged user monitoring*, but that only catches when users access server devices with their enhanced privileges. The SIEM watches a much broader slice of activity which includes accessing applications, email, etc.

Don't expect to start pumping PUM events into the SIEM and fairy dust to start drifting out of the dashboard. You still need to do the work to add correlation rules that leverage the PUM data and update reports, etc. We discuss the process of managing SIEM rule sets fairly extensively in both our Understanding and Selecting SIEM/Log Management and Monitoring Up the Stack papers. Check them out if you want more detail on that process.

Don't expect to start pumping PUM events into the SIEM and fairy dust to start drifting out of the dashboard. You still need to do the work to add correlation rules that leverage the PUM data and update reports, etc.

Summary

With all the pressure to increase productivity and decrease costs, IT organizations continue to evaluate new ways of provisioning, managing, and hosting critical IT resources. This includes tighter integration with business partners, increasing the level of outsourcing of management and hosting applications and databases in a variety of cloud computing architectures, and various other options. But the common thread in any specific architecture is the privileged user. Users with privileges to manage critical devices can cause outsized damage to any IT operation, so managing these privileged users more actively has become imperative for many organizations.

As with most operational functions, privileged user management (PUM) can be thought of in terms of a lifecycle, starting when a user is provisioned with escalated privileges and ending when those privileges are removed or the user leaves the organization, whichever comes first. In this paper we have made our case that the lifecycle includes restricting access, protecting credentials, enforcing appropriate entitlements, and finally monitoring exactly what users do on those devices. Fail in any of those key functions and the keys to your kingdom are exposed.

The earlier in the lifecycle you address the threat, the better. So we recommend you focus first on making sure the wrong folks can't access devices, either at the network layer or by implementing a proxy gateway to ensure administrators access only the devices they can legitimately manage. Then protect the credentials they use to log in with a password vault, and work to avoid transferring any credentials to compromised endpoint devices. Then manage what they can do via fine-grained control of commands, either via that proxy gateway or an agent on the managed devices. Finally, watch what they are doing to ensure you have a forensic audit trail and also for deterrence, which may include session recording.

Finally, don't forget to factor in how the cloud impacts PUM. The fundamentals are the same, but implementation of the controls differs substantially, as you no longer control the infrastructure and have limited visibility inside the clouds. Consistency is key to ensuring that privileged users are subjected to the same rules of engagement, regardless of whether the device is in your datacenter or a cloud. You need to strive for a single PUM policy, with the ability to enforce that policy regardless of where your computing resources reside.

If you have any questions on this topic, or want to discuss your situation specifically, feel free to send us a note at info@securosis.com or ask via the Securosis Nexus (http://nexus.securosis.com/).

About the Analyst

Mike Rothman, Analyst/President

Mike's bold perspectives and irreverent style are invaluable as companies determine effective strategies to grapple with the dynamic security threatscape. Mike specializes in the sexy aspects of security — such as protecting networks and endpoints, security management, and compliance. Mike is one of the most sought-after speakers and commentators in the security business, and brings a deep background in information security. After 20 years in and around security, he's one of the guys who "knows where the bodies are buried" in the space.

Starting his career as a programmer and networking consultant, Mike joined META Group in 1993 and spearheaded META's initial foray into information security research. Mike left META in 1998 to found SHYM Technology, a pioneer in the PKI software market, and then held executive roles at CipherTrust and TruSecure. After getting fed up with vendor life, Mike started Security Incite in 2006 to provide a voice of reason in an over-hyped yet underwhelming security industry. After taking a short detour as Senior VP, Strategy at eIQnetworks to chase shiny objects in security and compliance management, Mike joined Securosis with a rejuvenated cynicism about the state of security and what it takes to survive as a security professional.

Mike published The Pragmatic CSO <http://www.pragmaticcso.com/> in 2007 to introduce technically oriented security professionals to the nuances of what is required to be a senior security professional. He also possesses a very expensive engineering degree in Operations Research and Industrial Engineering from Cornell University. His folks are overjoyed that he uses literally zero percent of his education on a daily basis. He can be reached at mrothman (at) securosis (dot) com.

About Securosis

Securosis, LLC is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

Our services include:

- **The Securosis Nexus**: The Securosis Nexus is an online environment to help you get your job done better and faster. It provides pragmatic research on security topics that tells you exactly what you need to know, backed with industry-leading expert advice to answer your questions. The Nexus was designed to be fast and easy to use, and to get you the information you need as quickly as possible. Access it at <https://nexus.securosis.com/>.

- **Primary research publishing**: We currently release the vast majority of our research for free through our blog, and archive it in our Research Library. Most of these research documents can be sponsored for distribution on an annual basis. All published materials and presentations meet our strict objectivity requirements and conform to our Totally Transparent Research policy.

- **Research products and strategic advisory services for end users**: Securosis will be introducing a line of research products and inquiry-based subscription services designed to assist end user organizations in accelerating project and program success. Additional advisory projects are also available, including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessment.

- **Retainer services for vendors**: Although we will accept briefings from anyone, some vendors opt for a tighter, ongoing relationship. We offer a number of flexible retainer packages. Services available as part of a retainer package include market and product analysis and strategy, technology guidance, product evaluation, and merger and acquisition assessment. Even with paid clients, we maintain our strict objectivity and confidentiality requirements. More information on our retainer services (PDF) is available.

- **External speaking and editorial**: Securosis analysts frequently speak at industry events, give online presentations, and write and/or speak for a variety of publications and media.

- **Other expert services**: Securosis analysts are available for other services as well, including Strategic Advisory Days, Strategy Consulting engagements, and Investor Services. These tend to be customized to meet a client's particular requirements.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Additionally, Securosis partners with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis. For more information about Securosis, visit our website: <http://securosis.com/>.