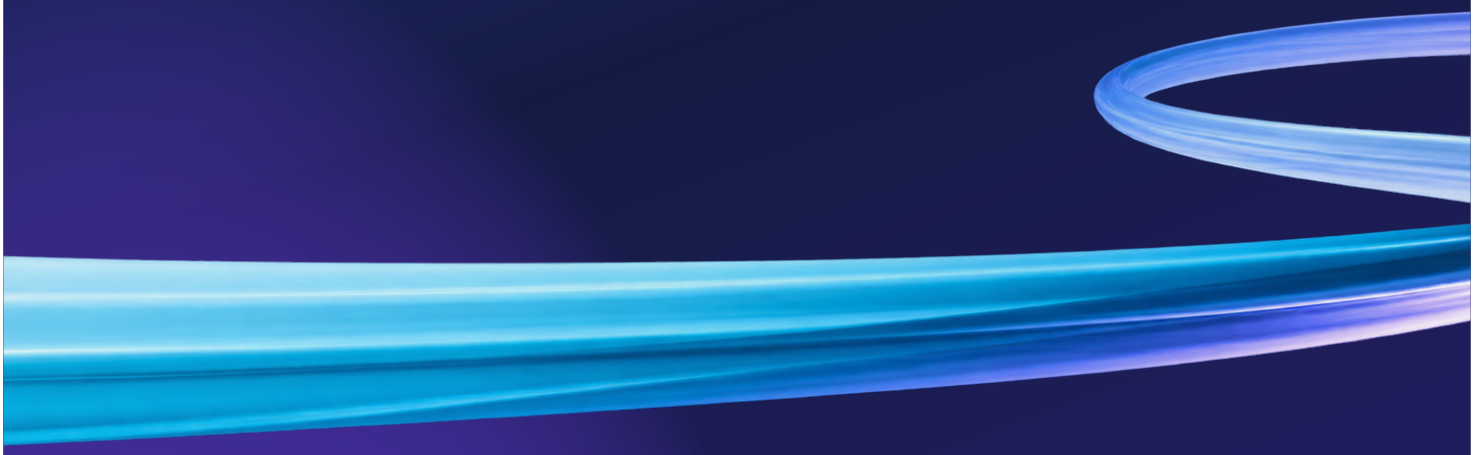
xtype | servicenow

AN XTYPE E-BOOK

Closing the Gaps Auditors Find First

Privileged access and change governance in ServiceNow



WHAT'S INSIDE

Contents

	Introduction	3
1	The two findings auditors write up first	4
2	Who really has power on your platform	7
3	Change governance: the gap between instances	10
4	Emergency access is not the problem. Silence is.	12
5	What good looks like, and what the rules now require	14
6	Seeing both gaps in one place: xtype Compliance	17
	The Privileged Access & Change Governance Review	20
	Start with your own data	21
	About xtype	22

Product screens are from xtype Compliance for ServiceNow, shown with illustrative demo data.

INTRODUCTION

Two findings, one risk

Two findings appear in audit reports more than any others: too many people hold high access, and changes reach production without independent review.

ServiceNow has good controls for both inside each instance. Proving that those controls worked, across every instance and over a full audit period, is much harder.

This short guide explains why, shows what a well-run platform looks like, and maps each practice to the rules that now require it.

IN THIS GUIDE

Chapter 1 names the two findings and why native controls alone cannot prove them.

Chapters 2 and 3 take each finding in turn: who really holds power, and how change crosses instances.

Chapter 4 deals with emergency access, and why the record matters more than the event.

Chapter 5 lists six habits and maps them to SOX, DORA and NYDFS Part 500.

Chapter 6 shows how xtype Compliance makes both gaps visible.

01 CHAPTER 1

The two findings auditors write up first

Audit reports are more similar than people expect. Across industries and across years, two findings appear again and again.

The first: too many people have high access, and no one checks it often enough. Accounts are created for a project and never removed. People change jobs but keep old permissions. Nobody is sure who still needs what.

The second: changes reach production without independent review. The person who built a change is also the person who moved it into production. Or a change was made directly in the live system, with no record of who approved it.

Both findings describe the same risk in different words: someone had the power to act alone.

ServiceNow gives you good tools to control both. What it does not give you is an easy way to prove it across your whole stack.

ServiceNow is not weak here

Platform owners already rely on strong native controls, and this guide assumes you do too:

- **Elevated roles:** the security_admin role must be switched on for each session, so the most sensitive power is not always active.
- **Change approvals:** change requests send work through review and approval before release.
- **Update set preview:** each update set is checked for conflicts and missing parts before it is committed.
- **Instance Security Center:** each instance is measured against recommended security settings.
- **Audit tables:** changes to audited records are written down, field by field.

Each of these works well at the moment of action, inside one instance.

What they do not give you is one connected record: who held power over time, how each change moved across every instance, and a file you can give an auditor for a full period. That is where the gaps in this guide appear.

Here is why

A ServiceNow stack is not one system. It is several separate instances (usually development, test and production), and each one only knows about itself. Work moves between them, but no instance holds the full story.

Access is also given in several different ways: directly to a person, through a group, or through a role that contains another role. And the platform's own change history does not last forever; some logs are removed automatically after a short time, and many teams limit how long audit data is kept to control table size.

So when the question comes, the answer is usually a person, a script, and three days of work.

SCENARIO · THE QUESTION IN THE ROOM

An external auditor asks a simple question: "In March, who had administrator access to production, and who gave it to them?"

The platform team knows who has access today. They can export that list in ten minutes. But March was seven months ago. People joined and left. The platform's own record of those changes is gone.

The team spends three days rebuilding the answer from tickets, emails and memory. The auditor writes up the finding anyway: not because access was wrong, but because the company could not prove it was right.

That last sentence is the point of this guide: your stack is probably secure, but you cannot prove it. The next five chapters describe what to fix, and in what order.

BEFORE YOU READ ON

Words used in this guide

Instance: one copy of ServiceNow. Most companies have several.

Role: a set of permissions. “admin” is the most powerful one.

Update set: a package of work that moves from one instance to another.

Stack: the collection of ServiceNow instances a company runs. Governed together, it becomes a single estate.

Group: a set of people. A group can hold roles, so everyone in it receives those roles.

Deployment: the act of moving that work into another instance, such as production.

02 CHAPTER 2

Who really has power on your platform

Start with a question that sounds easy: who has administrator access here?

It is not easy, for three reasons. First, “admin” is not the only powerful role. Roles such as security_admin, HR admin or a DevOps admin can also read sensitive data or change how the platform behaves. Some powerful roles do not look powerful from their name.

Second, access arrives by different paths. One person may hold a role directly. Another may receive the same role because they belong to a group. A third may hold it because one role contains another. A simple list of “who has admin” does not show how any of them got it. An auditor will ask exactly that.

So what counts as a powerful role?

Do not judge a role by its name. Judge it by what it can do. A role is powerful if it can:

- change code that runs on the platform, such as business rules and scripts, including through delegated development, where people who are not administrators are allowed to build;
- change the access rules (ACLs) that decide what other roles can see and do;
- read or change sensitive data, such as HR or financial records;
- or contain another role that can do any of these, sometimes several layers deep.

Write this list down, agree it with security and compliance, and review it on a schedule. Many teams find powerful roles that are not called admin at all. And remember that power can also grow through the access rules themselves, without anyone’s roles changing. Chapter 6 shows how to see this.

Third, most high access is standing access: it is always on, whether it is being used or not.

This is the quiet problem. An account created for a project two years ago still has full rights today. Nobody removed it, because nobody was sure it was safe to remove.

SCENARIO · THE ACCOUNT THAT NEVER SIGNED IN

A platform owner reviews privileged accounts for the first time in a year. For example: seventy-two accounts hold powerful roles. More than half have not been used in months. Twenty-three have never signed in at all.

One of them belongs to a contractor who finished work eleven months ago. The account was never used, so it never appeared in any activity report. It was also never removed.

Nothing bad happened. But for eleven months, a full administrator account sat unused, unwatched and outside the company's normal sign-in system. If someone had taken it, no one would have noticed.

There is one more detail worth checking

Some accounts sign in through the company login system, where rules such as multi-factor authentication are enforced. Others use a local ServiceNow password instead. A local password skips the company system completely, and with it every rule your security team built there.

A privileged account with a local password is a door beside the main entrance.

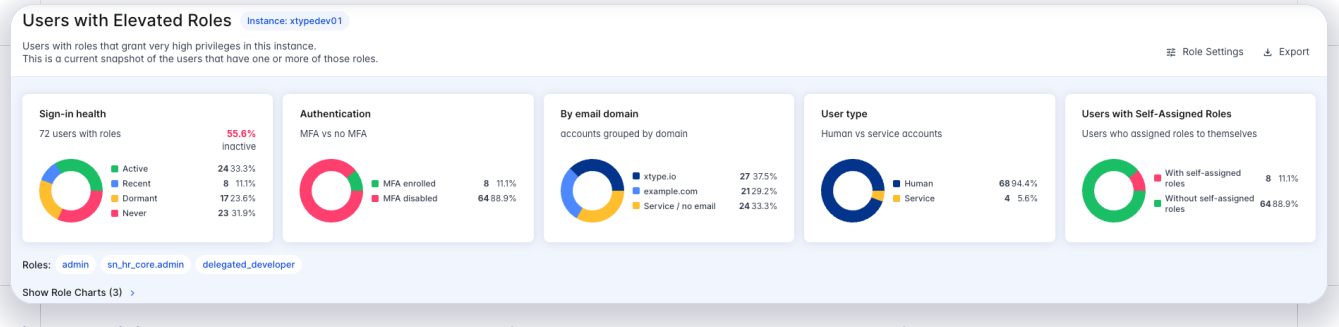


Figure 1: Privileged access at a glance. One screen for one instance: 72 accounts hold powerful roles, 55.6% are inactive and 23 have never signed in. Beside sign-in health sit multi-factor authentication status, accounts grouped by email domain, people versus service accounts, and the 8 users who gave a role to themselves. Each role can be opened to show how every person received it (see Figure 4). *Illustrative demo data.*

What to take away

Reviewing access is not one task. It is three short passes.

1 **Find accounts nobody uses.**

2 **Find accounts that skip the company login.**

3 **Find roles people gave to themselves.**

Those three passes catch most of what an audit will find.

03 CHAPTER 3

Change governance: the gap between instances

Now the second finding. The rule is simple, and almost every framework states a version of it: the person who builds a change should not be the only person who moves it into production.

Proving it is harder, because the building and the deploying happen in different places. The work is created in development. It is tested somewhere else. It is deployed into production. Each instance holds its own part of the story, and no instance holds the whole one.

A detail that surprises many platform owners

When work is deployed, the record left behind in ServiceNow often names the account that performed the technical step (frequently a service account or an automation user), not the person who started it. So a real problem can look completely clean in the native record: the builder is a person, the deployer appears to be "system", and the two names do not match. The control appears to pass. Nobody is hiding anything; the record simply does not know who pressed the button.

SCENARIO · THE CHANGE THAT LOOKED CLEAN

An engineer builds a change in development and deploys it to production the same afternoon. That breaks the separation rule.

But in the production record, the deployment shows a service account as the deployer. The builder is a person; the deployer is not. On paper, two different "people" were involved.

The report says compliant. The reality was not. Six months later, an incident review finds the change. The question becomes why the control did not catch it.

Two more gaps to remember

- **Changes created directly in production,** which skip the normal path completely. They are sometimes legitimate, but they should always be visible.
- **Unreviewed deployments at volume.** When hundreds of changes move each month, nobody can review them by reading lists.

Change reaches production by more than one road

Update sets are the most common road. But many stacks also publish scoped applications through the Application Repository, deploy from source control or DevOps pipeline tools, change data directly, or run fix scripts in production. Each road needs its own evidence. A good first step is simply to list the roads your stack uses, and to know which ones your evidence covers.

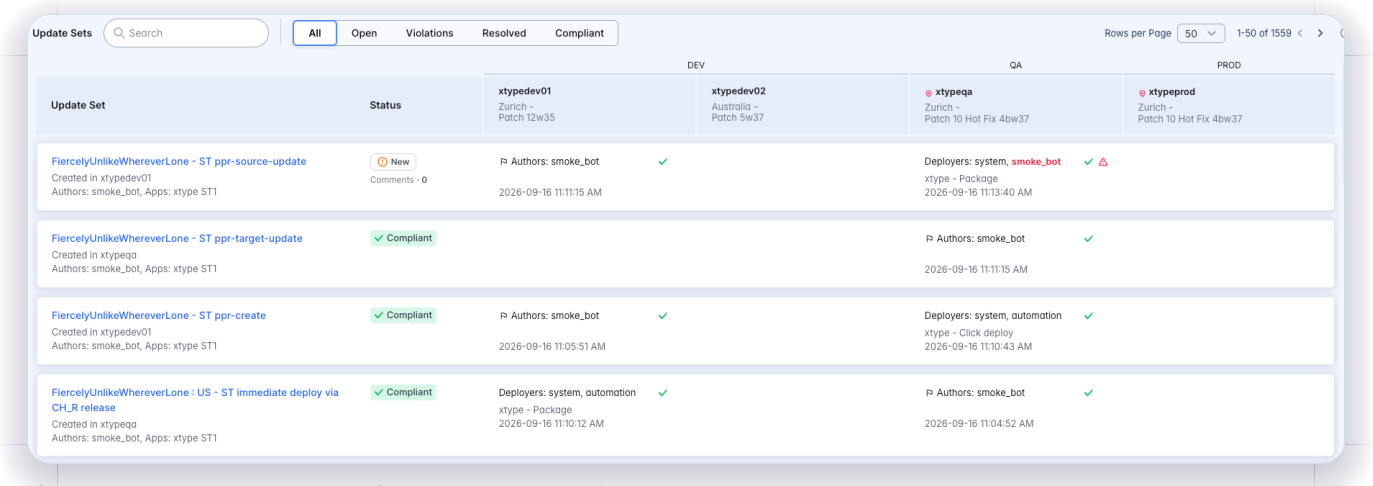


Figure 2: One change, across every instance. Each piece of work on one line as it moves through the stack: who created it, who deployed it, to which instance and when. Findings are raised only on the instances you choose to protect (marked with a shield). In the first row, the deployment to QA lists the platform's system account alongside smoke_bot, resolved from xtype's own deployment data; smoke_bot also authored the change, so the row is flagged. *Illustrative demo data.*

> **AN HONEST LIMIT, STATED PLAINLY**
>
> This shows who built the change and who deployed it. It does not show who approved it. Approval evidence still lives in your change process. Knowing the difference keeps your answers accurate when an auditor asks.

04 CHAPTER 4

Emergency access is not the problem. Silence is.

Every platform needs a way to act fast. A payroll run fails the night before payday. A production defect blocks customers. Someone needs powerful access immediately, and waiting for the normal process would cause real harm.

Auditors know this. They do not expect emergency access to be impossible. They expect three things instead: that it was recorded, that someone reviewed it afterwards, and that the reason was written down. This is often called break-glass access. When you break the glass, you must be able to explain why.

The failure is almost never the emergency itself. The failure is silence afterwards. Nothing was recorded. Nobody reviewed it. Six months later, no one remembers, and the event now looks exactly like misuse, because the two are indistinguishable without a record.

SCENARIO · THE 2 A.M. FIX

A senior engineer is called at two in the morning. A failed job is blocking a customer process. She grants herself the access she needs, fixes the problem in twenty minutes, and removes the access again before going back to sleep. This is good work.

Now imagine the same events with no record: a powerful role granted at 2 a.m. and removed by 2:30 a.m., by the person who received it, with no explanation anywhere.

Written down, it is a professional response to an incident. Not written down, it is the exact pattern an investigator would flag as suspicious. The actions were identical. Only the record was different.

This is why review status matters as much as detection

A good review process lets a reviewer mark an event as acceptable. It is not a “false positive”, because it truly happened. It is not a “violation”, because it was justified. It happened, it was checked, and here is the reason. That written reason is the evidence.

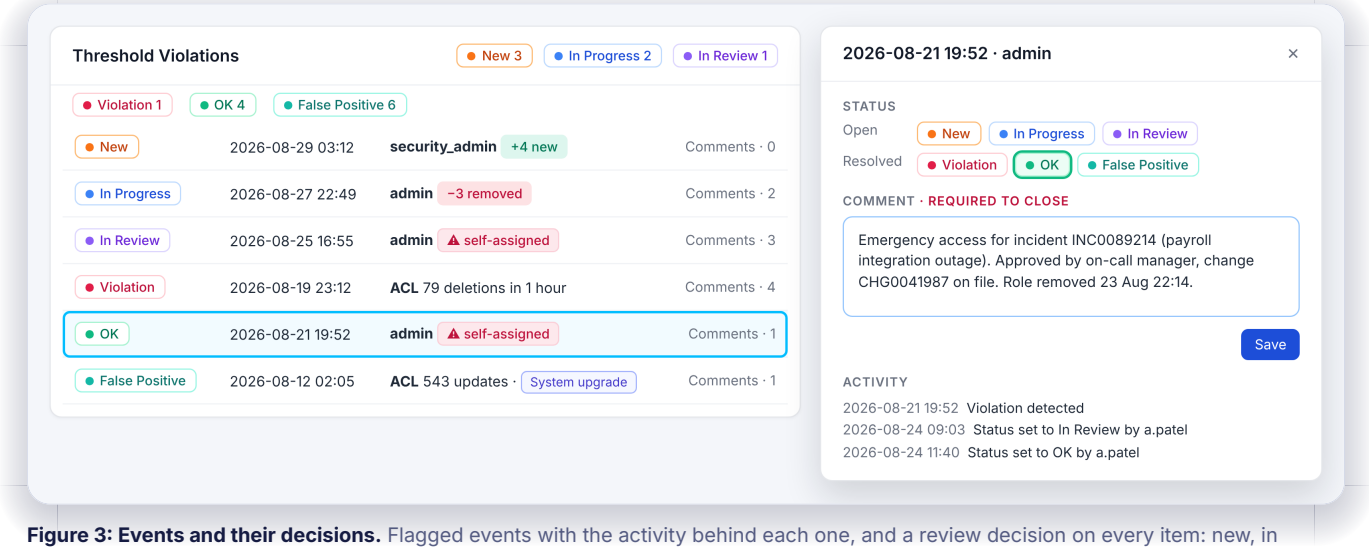


Figure 3: Events and their decisions. Flagged events with the activity behind each one, and a review decision on every item: new, in progress, in review, violation, false positive or OK (acceptable), each requiring a written comment before it is closed. *Illustrative demo data.*

> **The goal is not zero flagged events. The goal is zero unexplained ones.**

05 CHAPTER 5

What good looks like, and what the rules now require

A well-governed ServiceNow stack does not depend on people remembering things. It has six visible habits.

1. **Powerful roles are named and watched.** You have a list of which roles count as high access, based on what each role can do and not on its name, and it is reviewed on a schedule.
2. **Access is reviewed regularly, not only before an audit.** Unused accounts are removed. Every remaining one has a reason.
3. **You can show how someone received a role,** not only that they have it.
4. **Builder and deployer are separated, and you can prove it** with the real person behind each deployment, across every instance.
5. **Emergency access is allowed, recorded and reviewed afterwards,** with the reason written down.
6. **Evidence can be produced in minutes,** in a file that explains itself: what period it covers, when it was created and who created it.

These are no longer only good practice

Regulators have moved toward the platforms that run critical work, with new or increasingly strict requirements. Three key regulatory frameworks now apply to most of our readers, with enforcement already under way.

RULE SET	WHAT IT EXPECTS (IN SIMPLE TERMS)	WHICH HABIT IT MAPS TO
SOX (ITGC) US listed companies	Control who can change systems and data. Show that changes were authorized and reviewed. Keep the evidence long enough for the audit.	Access review · builder–deployer separation · evidence on demand
DORA EU financial entities	Manage and document ICT changes. Limit access rights and review them. Be able to trace what changed and who changed it.	All six: DORA is explicit about both access and change
NYDFS Part 500 NY financial services	Limit privileged access and review it at least once a year. Require multi-factor authentication. Keep audit trails, including records that allow material events to be reconstructed.	Named roles · access review · local-login accounts · retained evidence

ONE HONEST NOTE ABOUT EVIDENCE

A record begins on the day you start keeping it. No tool can recover history that the platform already deleted. This is not a weakness. It is the reason to start now rather than later. A record with a clear start date is credible. A claim to know everything is not.

WHERE TO BEGIN THIS WEEK

Choose your production instance

Answer three questions:

1. **Who holds powerful roles, and how did they get them?**
2. **Which changes in the last month were built and deployed by the same person?**
3. **Which emergency access events happened, with a written reason for each?**

If you cannot answer in an afternoon, the gap is not your team. It is the record.

→ **Download the one-page checklist: The Privileged Access & Change Governance Review.**
14 questions an auditor is likely to ask, and where to find each answer.

06 CHAPTER 6

Seeing both gaps in one place: xtype Compliance

The six habits in Chapter 5 are easy to agree with and hard to prove by hand.

xtype Compliance is a set of reports inside xtype for ServiceNow that answers the two findings from this guide directly. Compliance and audit teams can read the reports without ServiceNow administrator rights, so the people who review the platform no longer need to ask the platform team for the evidence.

Gap one: who really has power

The Users with Elevated Roles report (Figure 1) shows, for one instance, every account that holds a role you have marked as powerful. It answers the Chapter 2 questions on one screen. In the example, 72 accounts hold powerful roles; 55.6% are inactive and 23 have never signed in.

- **Unused power is visible at once.** Accounts are grouped as active, recent, dormant or never signed in. The dormant and never groups are your shortest cleanup list.
- **People and service accounts are separated,** so each can be reviewed with the right rules.
- **Self-assigned roles are counted for you.** Here, 8 people gave a powerful role to themselves.
- **Each role shows its own risk.** One role can be well managed while another is not. The role charts show where to look first.

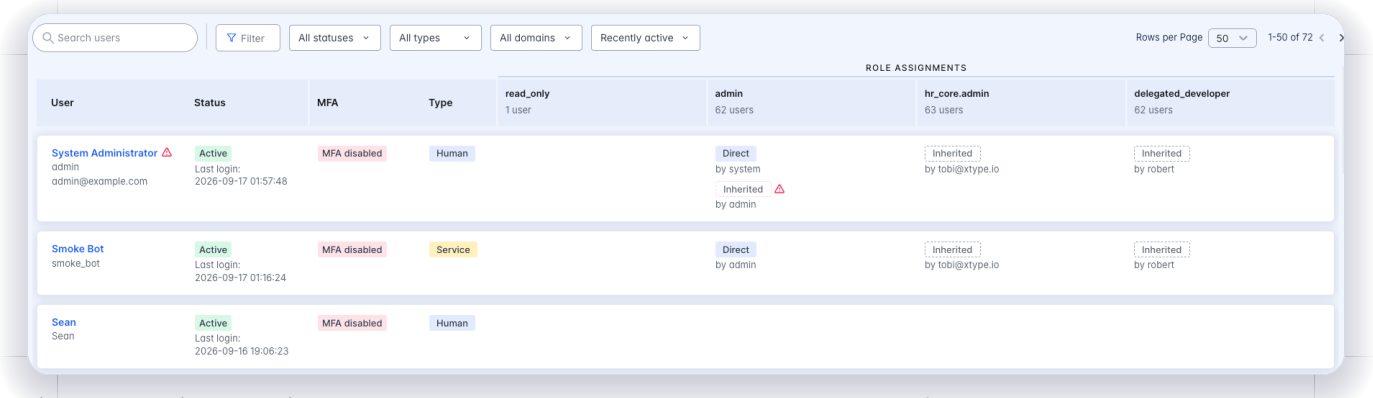


Figure 4: How did they get it?. The same report, table view. For every person and every powerful role: Direct or Inherited, and by whom. The answer to "how did they get it?", not only "who has it?" *Illustrative demo data.*

This report is a picture of today, read live from the instance. For history (who received or lost a watched role, when, and from whom), xtype's Role Assignments report keeps the record from the day monitoring starts, with simple rules that flag unusual activity for review.

Beyond named roles

In Users with Elevated Roles, you choose which roles count as powerful, including roles such as delegated_developer, as in the example above. But power does not only come from the roles on your list. Two more xtype reports look at the other paths:

- **Direct Role Holders** finds every role given directly to a person instead of through a group. Before it raises a finding, it checks groups inside groups and roles inside roles, so a role already covered by a group is not counted as a problem. It suggests existing groups that would cover each person's direct roles, and it lists roles that no group grants at all.
- **ACL Audit** lists every access rule on the instance and the roles each rule allows. It marks rules that let administrators skip the check, and rules where a script decides the result. You can search for the rules that protect scripts and business rules and see exactly which roles they allow. From the day monitoring starts, it also records every change to the rules (including a role being added to a rule, which widens power without changing anyone's roles) with the old and new value of each changed field, where ServiceNow audits that field.

Gap two: changes that reach production without independent review

The Segregation of Duties report follows each update set across all your instances (development, test and production) on one line, as in Figure 2. It shows who built it, who deployed it, where and when.

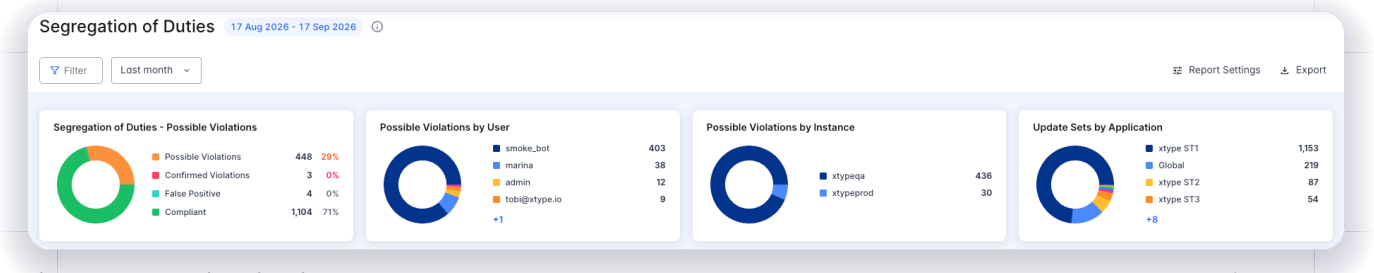


Figure 5: Segregation of Duties, one month. Headline view: 448 possible violations among the update sets in the period, 3 confirmed and 4 marked false positive. The by-user chart is led by smoke_bot, an automation account, which points to a process to fix rather than a person to blame. *Illustrative demo data.*

- **One line per change, across every instance.** The Chapter 3 problem, each instance only knowing its own part, is solved by putting the parts side by side.
- **The real person, not only the service account.** xtype can use its own deployment data to find who started each deployment. This is the fix for "the change that looked clean."
- **Changes made directly in production are visible.** They appear as their own category, so a real emergency can be explained and an unplanned change can be found.
- **A chart shows who appears most often.** When one name leads the list (here, an automation account), it usually points to a process gap to fix, not a person to blame.
- **You choose what is protected.** Findings are raised only on the instances your change policy protects.
- **What the report covers.** The Segregation of Duties report covers update sets, whether they are moved by xtype (packages, auto deploy, click deploy), moved with ServiceNow's own update set tools, or created directly in a protected instance. Applications published through the Application Repository, deployments from source control or pipeline tools outside xtype, direct data changes and fix scripts are outside its scope today; list those roads separately in your review.

Closing the loop: decisions and evidence

In every report, each flagged item can be given a status, including "acceptable" for a documented emergency, and a written comment is required. Every report can be exported as a spreadsheet with a built-in information sheet: a unique export number, when the file was made, what period it covers, which instance, which filters, and who exported it.

The file explains itself.

SCOPE TODAY

The reports show who built and who deployed each change; approval evidence stays in your change process.

Rules are checked when a report is opened; automatic alerts are on the roadmap.

History begins on the day monitoring is switched on for an instance.

Segregation of Duties covers update sets; other roads to production carry their own evidence.

CHECKLIST

The Privileged Access & Change Governance Review

Fourteen questions an auditor is likely to ask, in three groups.

Group 1: Who has power

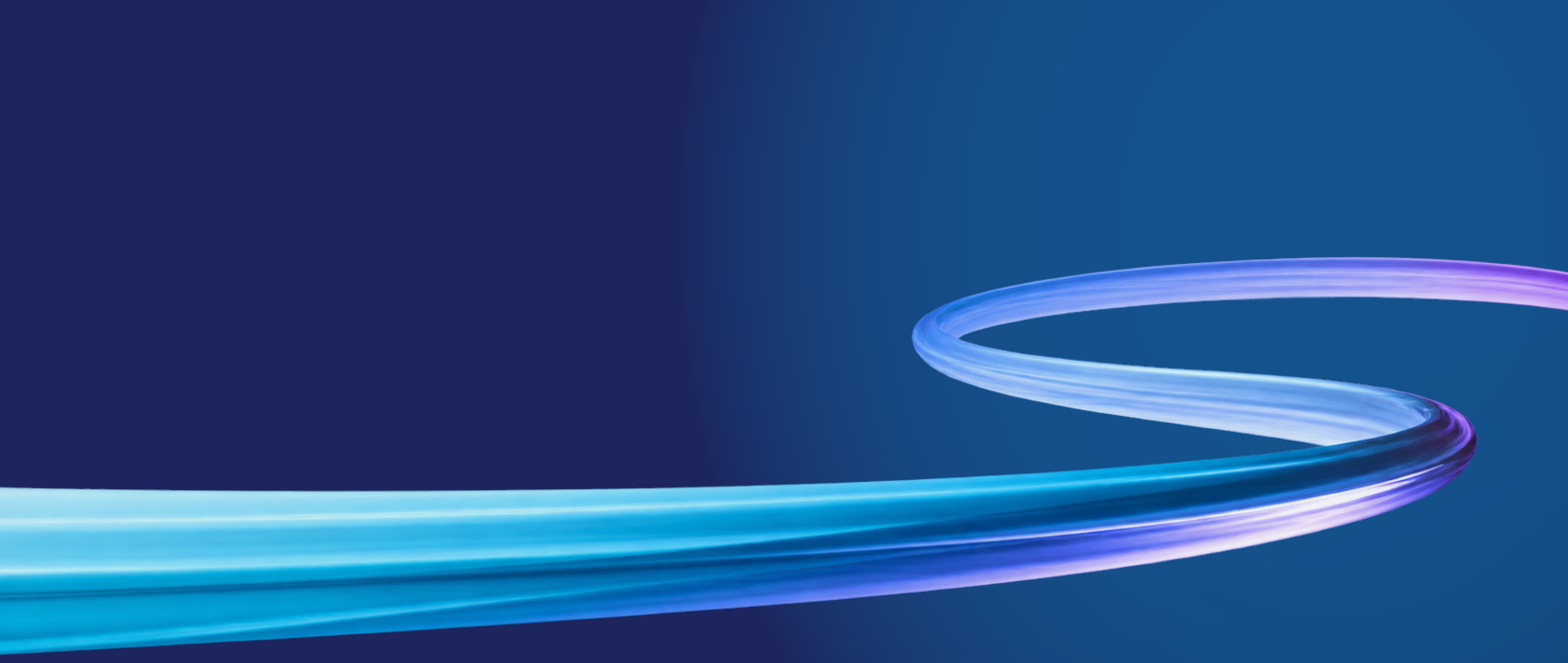
1. Which roles do we treat as high access? When did we last agree that list?
2. Is that list based on what each role can do (scripts, access rules, sensitive data, roles inside roles), and not only on its name?
3. How many accounts hold them, and how many have not signed in for 90 days?
4. Which privileged accounts use a local password instead of the company login?
5. Which are service accounts, and who owns each one?
6. Who has given a powerful role to themselves in the last 90 days?

Group 2: How changes reach production

7. Which instances do we treat as protected?
8. In the last month, which changes were built and deployed by the same person?
9. Do our deployment records show the real person, or a service account?
10. Which changes were created directly in production, and were they intended?
11. Which roads can a change take to production (update sets, application publishing, pipelines, data changes, fix scripts), and does our evidence cover each one?
12. Can we show this for every instance, not only production?

Group 3: Proving it

13. For each flagged event, is there a decision and a written reason?
14. Can we produce the evidence for a past period, and does the file say what period it covers and when it was made?



AVAILABLE NOW

The best test is your own data

The xtype Compliance reports are available now. Ask for a demonstration session on one of your instances.

Book a demonstration →

About xtype

xtype is a governance platform built for ServiceNow. It gives enterprises a single, real-time view of every change, access decision and configuration, across every instance they run, so that governance is provable, not reconstructed after the fact.

xtype Compliance, the product behind the reports shown in this e-book, turns that record into audit-ready evidence for privileged access, segregation of duties and change governance, mapped to frameworks such as SOX, SOC 2, ISO 27001, HIPAA and GDPR.



Visit xtype.io ↗ | Book a demo

LinkedIn YouTube

Product screens show xtype Compliance with illustrative demo data. This guide is general information, not legal advice; confirm how each regulation applies to your organization with your compliance and audit teams. ServiceNow, the ServiceNow logo, Now, Now Platform and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc. in the United States and/or other countries.