

4.0 Objectives

What to do when things go wrong - designing in visibility & observability & security

- Everything will eventually break - make a plan
- If you can make it, you can hack it – make a plan
- The weakest link is often the humans – make a plan
- A **plan** is just another **workflow**!
- Some stories of mega-oops. Crowdstrike, Reply-all, fragile file names, shared drives, live-on-air post-its on monitors...
- Think like a baddie - security start with line 1 of your planned workflow, zero-trust red team / blue team speed dating - will your workflow survive your colleagues?

Learning outcomes

- Understanding the difference between a prototype and a production workflow
- Basic understanding of workflow security topics
- Understanding that defenders have to succeed every time, attackers only have to succeed once
- Basic understanding of how fragile workflows can be

Published

Assign To

Edit

Student Pairs - assignment a4.1

1, 9
10, 8
11, 7
12, 6
13, 5
14, 4
15, 3
16, 2

 Observability Workshop:
Visibility → Cause in Grafana Play

This exercise uses the public **Grafana Play** environment to demonstrate the critical difference between **Visibility** (seeing a problem metric) and **Observability** (correlating metrics with logs/traces to find the root cause).

Tool: Grafana Play (Public Demo Instance)

Focus: Correlating Metrics (Prometheus) and Logs (Loki)

 Step-by-Step Guide: The Observability
Pivot

Phase 1: Access and Orientation (Visibility)

Goal

Access the demo, find a dashboard, and identify a problematic metric spike.

Step 1: Access the Grafana Play Environment

1. Open your web browser and navigate to the official Grafana demo site: play.grafana.org .
2. Navigate to the **Dashboards** menu (the stack of squares icon) on the left sidebar.
3. Open a detailed statistics dashboard, such as "**Prometheus 2.0 Stats**" or "**Grafana Internal Stats**."

Step 2: Identify a Problem Metric (Visibility)

1. Scan the graphs for metrics related to application performance (e.g., Request Duration, Latency, Error Rate).
2. Find a graph that shows a clear **spike** or **deviation** from the normal line—this is your anomaly.
3. **Action:** Hover your mouse over the anomaly.
 - **Record 1:** Note the **exact timestamp** (Date and Time) when the spike occurred.
 - **Record 2:** Note the **name of the metric** (e.g., `http_request_duration_seconds`).
 - *This data represents your **VISIBILITY**. You know **WHAT** happened, but not **WHY**.*

Phase 2: Drilling Down for Observability

Goal

Use the problem metric's timestamp to search logs and uncover the root cause.

Step 3: Access the Explore Feature

The **Explore** feature is Grafana's interface for investigative querying, allowing you to seamlessly pivot between different data sources.

1. Click on the **"Explore"** icon (the compass or graph icon) in the left-hand navigation panel.
2. In the main workspace, find the **Data Source selector** at the top left.

Step 4: Correlate Metrics to Logs (The Observability Pivot)

To find the root cause, you must search the log data source (Loki) during the exact time the metric spiked.

1. In the Data Source selector, change the source from **Prometheus** (Metrics) to **Loki** (Logs).
2. Use the **Time Range Selector** (top right) to narrow the window. Set the range to focus only on the time **5 minutes before and 5 minutes after** the anomaly time recorded in Step 2.
3. In the **Log Queries** field, enter a basic label filter to pull relevant logs. A common filter used in the Play environment is:

```
{job="grafana"}
```

4. Click **"Run Query."**

Step 5: Identify the Root Cause

1. Review the log lines returned for that narrow time window.
2. Look specifically for log lines containing keywords that explain a failure or slowdown:
 - `error` , `panic` , `timeout`
 - `restarting` , `database failure` , or high-volume bursts.
3. **Conclusion:** By correlating the **time of the metric spike** with the **content of the logs**, you have found the **root cause** (the *why*). You have successfully shifted from **Visibility** (a spike on a graph) to **Observability** (a database timeout message in the logs).



Key Configurations to Manipulate

Experiment with these controls in the **Explore** view to deepen your understanding of each data source:

Data Source	Configuration	Learning Objective
Prometheus (Metrics)	Query Functions: Wrap your metric in functions like <code>rate()</code> or <code>sum()</code> .	Understand how data aggregation changes the story told by the metric.
Loki (Logs)	Label Filters: Change the query from <code>{job="grafana"}</code> to <code>{job="grafana", level="error"}</code> .	Learn to use metadata (labels) to quickly narrow the scope of investigation to only problematic events.
Both	Time Range Selector: Constantly zoom in and out around the anomaly.	Emphasize that context and time segmentation are the most powerful tools in observability.

Points 0
Submitting Nothing

Due	For	Available from	Until
-	Everyone	-	-

 [Rubric](#)

Workflow Observability for Media Technicians

Why this matters to you

As a media technician, you're responsible for keeping complex workflows running smoothly. These workflows involve multiple people, software tools, and often span different locations and time zones. When something breaks, the financial pressure is immediate. A typical broadcast facility loses £5,000-15,000 per hour of downtime, and post-production delays can cascade into missed delivery slots worth tens of thousands more.

This guide explains four critical concepts that give you the agency to detect, diagnose, and fix problems quickly. Understanding these concepts helps you evaluate your tools, improve your workflows, and most importantly, minimize time-to-fix when things go wrong.

The four concepts

Visibility

Can you see what's happening right now?

Visibility means you can observe the current state of your workflow in real-time. This includes status indicators, progress bars, live metrics, and dashboards that show you what's running, what's waiting, and what's failed.

Good visibility looks like:

- Dashboard showing all active transcode jobs with progress percentages
- Real-time display of storage capacity across all servers
- Status page showing which workstations are rendering and which are idle
- Email or Slack alerts when a job fails or a queue backs up

Poor visibility looks like:

- Having to manually check each machine to see if it's working
- Finding out about failures only when users complain

- No way to know if a process is running or hung
- Batch jobs that fail silently overnight

Impact on time-to-fix: Good visibility can reduce detection time from hours to minutes. You know immediately when something breaks, rather than discovering it hours later when someone asks "where's my file?"

Observability

Can you understand why it's happening?

Observability goes deeper than visibility. It's about understanding the internal state and behavior of your systems through logs, metrics, and traces. When something goes wrong, observability lets you ask questions and get answers: Why did this fail? What was different this time? What happened just before the error?

Good observability looks like:

- Detailed logs that show every step of a file's journey through your workflow
- Timestamps that let you correlate events across different systems
- Metrics showing CPU, memory, network usage over time
- Error messages that include context: which file, what operation, what values were involved

Poor observability looks like:

- Generic error messages like "Processing failed" with no details
- No logs, or logs that disappear when you restart a service
- Unable to see what a "black box" system is doing internally
- No way to tell which of five possible causes triggered a failure

Impact on time-to-fix: Good observability can reduce diagnosis time from hours of trial-and-error to minutes of log analysis. Instead of guessing and restarting things randomly, you can pinpoint the root cause.

Debug-ability

Can you fix it when it breaks?

Debug-ability is about having the tools and access you need to diagnose and resolve problems. This includes documentation, command-line access, configuration tools, and the permissions to actually make changes.

Good debug-ability looks like:

- Clear documentation explaining what each service does and how to restart it
- Permission to access log files and configuration files
- Command-line or API access to query system state
- Test modes that let you reproduce issues safely

- Ability to roll back configuration changes if they make things worse

Poor debug-ability looks like:

- Locked-down systems where you can't access logs or change settings
- Undocumented systems where you have to guess how they work
- Having to wait for a specialist to arrive on-site to fix simple problems
- No safe way to test potential fixes without risking production
- One-way configuration changes that can't be undone

Impact on time-to-fix: Good debug-ability can reduce fix time from hours of waiting for specialists to minutes of applying documented fixes. You have agency to solve problems yourself.

Security

Who gets access to what, and when?

Security in workflow management isn't just about keeping bad actors out. It's about ensuring the right people have the right access at the right time, without creating bottlenecks that slow down legitimate work.

Good security looks like:

- Role-based access: technicians get diagnostic access, operators get monitoring access
- Audit logs showing who changed what and when
- Ability to grant temporary elevated access for troubleshooting
- Read-only access to sensitive systems for diagnosis without risk of accidental changes
- Secure credential management so you don't need to share passwords

Poor security looks like:

- Either everyone has full admin access (risky) or no one can fix anything (slow)
- Shared passwords that get written on sticky notes
- No way to grant temporary access, so people keep elevated privileges permanently
- Can't see logs without full system access, creating unnecessary security risks
- Security measures that slow down legitimate troubleshooting

Impact on time-to-fix: Good security enables quick diagnosis without security delays, while still maintaining accountability and protecting critical systems.

Time-to-fix: why these concepts matter

Time-to-fix is the total time from when something breaks until it's working again. It breaks down into four phases:

Technician agency: your power to fix things

Agency means you have the ability to take action. In workflow management, technician agency is about having the visibility, observability, debug-ability, and security access you need to keep things running.

Building visibility into your workflow

- Set up dashboards that show all active jobs
- Configure alerts for failures, not just successes
- Make status visible to everyone who needs it
- Use color coding: green for running, yellow for slow, red for failed
- Show trends over time, not just current status

Building observability into your workflow

- Enable detailed logging on all critical systems
- Use consistent timestamps and formats across systems
- Include context in error messages (filename, timestamp, values)
- Keep logs for at least 30 days
- Make logs searchable and filterable

Building debug-ability into your workflow

- Document every system: what it does, how to restart it, common issues
- Provide CLI or API access for advanced troubleshooting
- Give technicians appropriate permissions to fix common problems
- Create runbooks for frequent issues
- Build test modes that don't affect production

Building security into your workflow

- Use role-based access control, not shared admin passwords
- Log all significant actions for accountability
- Provide read-only access where write access isn't needed
- Create emergency access procedures for critical outages
- Review and update access permissions regularly

Remember: Every minute of downtime costs money. These four concepts—visibility, observability, debug-ability, and security—give you the agency to minimize time-to-fix. When evaluating new tools or designing workflows, ask yourself: Does this help me see what's happening? Understand why? Fix it when it breaks? And do so securely?

Questions to ask about your workflows

Use these questions to evaluate how well your current workflows support these concepts:

Visibility questions

- How long does it take me to know when something has failed?
- Can I see the status of all my workflows from one place?
- Do I get notified about problems, or do I have to go looking for them?
- Can I tell if a process is working, hung, or failed?

Observability questions

- When something fails, can I see detailed error messages?
- Are logs detailed enough to understand what was happening?
- Can I correlate events across different systems using timestamps?
- Do I have access to metrics showing resource usage over time?

Debug-ability questions

- Do I have documentation for all critical systems?
- Can I access log files and configuration files when needed?
- Do I have permission to restart services or roll back changes?
- Can I test potential fixes without affecting production?
- How often do I have to wait for someone else to fix things?

Security questions

- Do I have the access I need without having too much access?
- Are there audit logs showing who changed what?
- Can I diagnose problems without needing admin privileges?
- Is there a clear process for emergency access during outages?

Next steps: Identify one workflow where poor visibility, observability, or debug-ability is costing you time. What's the smallest change you could make to improve it? Start there.

 Published Assign To Edit

Student Pairs - assignment a4.2

1, 8
9, 7
10, 6
11, 5
12, 4
13, 3
14, 2
15, 16

SPECTRE Field Exercise - 007

Interdiction

You're a SPECTRE field-commander and have received intelligence in the form of a Post-It note workflow that 007 is planning on arriving in France.

1. Go to the wall of 007 flows and choose one that you were not involved in creating
2. Wait until everyone has a 007 flow before starting
3. Spend 30 minutes with your partner creating a workflow for your agent to ensure 007 gets on a bus to Gland and not the ferry to Evian-Les-Bains. Don't let your agent be detected. If 007's agent gets injured,

then that's an acceptable price of espionage (that increases your risk of being caught).

4. When everyone's done - explain your plan without adding details that aren't on your post-it

Points 0

Submitting a text entry box or a website url

Due	For	Available from	Until
-	Everyone	-	-

 [Rubric](#)

Workflow Security Basics

Why Security Matters in Media Workflows

Media workflows often involve high-value content, multiple collaborators, and tight deadlines. A security breach can mean:

- **Content leaks:** Unreleased films or shows appearing online costs studios millions in lost revenue
- **Workflow disruption:** Ransomware attacks can halt production for days or weeks
- **Credential theft:** Stolen passwords give attackers access to storage, editing systems, and rendering farms
- **Data loss:** Unauthorized deletions or modifications can destroy months of work

Security isn't just an IT problem—it's everyone's responsibility. As a technician managing workflows, you're a critical part of the defense. This guide covers the fundamentals: protecting access, verifying identity, and testing your defenses.

Access Control Fundamentals

Good Passwords: Long & Memorable Beats Short Gibberish

The old advice was wrong. For years we were told: "Use complex passwords with uppercase, lowercase, numbers, and symbols—and change them every 90 days." This created passwords like `P@ssw0rd123!` that were hard to remember but easy for computers to crack.

The new approach: length beats complexity. Modern password cracking tools can try billions of combinations per second. A short complex password like `Tr1ck!` (7 characters) can be cracked in seconds. A long simple passphrase like `correct-horse-battery-staple` (28 characters) would take centuries.

Has your email been hacked? Look at <https://haveibeenpwned.com> to check!

Why passphrases work better:

- **Memorable:** Four random words are easier to recall than random symbols
- **Typeable:** No hunting for special characters on keyboards
- **Mathematically stronger:** Each additional character increases cracking time exponentially

- **Less reuse:** When passwords are easy to remember, people create unique ones per site

Practical guidelines:

- **Minimum 16 characters** for work accounts, 20+ for critical systems
- **Use a password manager** to generate and store unique passwords for every service
- **Never reuse passwords** across sites—when one service is breached, all your accounts are vulnerable
- **Don't change passwords regularly** unless there's evidence of compromise (this just encourages weak passwords)
- **Avoid personal info** like names, birthdays, or studio names that appear in social media

For your team: If you manage workflow systems, enforce minimum password length (16+ characters) but don't require complexity rules. Length is what matters. Consider deploying a password manager like 1Password or Bitwarden for your team.

Multi-Factor Authentication (MFA): Why It's Good, Why It Breaks

What is MFA? Multi-factor authentication requires two or more independent proofs of identity:

- **Something you know:** Your password or PIN
- **Something you have:** Your phone, security key, or badge
- **Something you are:** Your fingerprint, face, or voice

Why MFA is essential: Even if your password is stolen (phishing, data breach, shoulder surfing), attackers can't access your accounts without the second factor. Statistics show MFA blocks 99.9% of automated attacks.

MFA methods ranked by security:

Method	Security	Pros	Cons
Hardware keys (YubiKey, Titan)	Best	Phishing-resistant, no batteries, durable	Costs money, can be lost
Authenticator apps (Authy, Google Authenticator)	Good	Free, works offline, relatively secure	Can be lost with phone, requires setup
Push notifications (Duo, Microsoft Authenticator)	Moderate	Convenient, user-friendly	Vulnerable to fatigue attacks (users approve without thinking)
SMS codes	Weak	Universal, no app needed	Vulnerable to SIM swapping, interception, and phishing

Why MFA breaks (and how to fix it):

- **Lost/broken second factor:** Always set up multiple MFA methods and keep backup codes in a secure location
- **Phone out of service:** Use authenticator apps that work offline, not SMS

- **MFA fatigue:** Attackers spam push notifications hoping you'll approve by accident—always verify the login details before approving
- **Time sync issues:** Authenticator codes are time-based—if your phone clock is wrong, codes won't work (solution: enable automatic time sync)
- **Remote access problems:** When working from location, ensure you can receive MFA codes (offline authenticators or pre-generated backup codes)

For your workflow: Require MFA for all systems with remote access: storage servers, cloud editing platforms, render farms, and admin panels. Use hardware keys for administrators and authenticator apps for general users.

Identity and Access Management

Identity: Sources of Truth

What is an identity? In workflow security, an identity is a verifiable representation of a person or system. The challenge in media production: people come and go quickly (freelancers, contractors, temporary staff), and workflows span multiple organizations (studios, post houses, VFX vendors).

Common sources of truth for identity:

- **Active Directory / LDAP:** Traditional on-premise directory services used by most studios and post houses. Central user database with group memberships and permissions
- **Cloud identity providers (Azure AD, Okta, Google Workspace):** Modern cloud-based systems that support single sign-on (SSO) across multiple services
- **SAML / OAuth providers:** Federated identity systems that let users log in once and access multiple services
- **Local accounts:** User accounts stored directly on individual systems—avoid these for workflows, they're hard to audit and revoke

Best practices for identity management:

- **Single source of truth:** Use one authoritative identity system (e.g., Active Directory) and sync to other services, rather than maintaining separate user lists everywhere
- **Automated provisioning:** When someone joins a project, automatically create their accounts and grant appropriate permissions. When they leave, automatically revoke access
- **Regular audits:** Review user lists monthly—remove inactive accounts and contractors who finished their work
- **Unique identifiers:** Use email addresses or employee IDs, not names (multiple people might share a name)
- **Group-based access:** Assign permissions to groups (VFX-Team, Audio-Editors, Producers), not individual users

The freelancer problem: Media workflows often involve short-term contractors who need quick access. Solutions:

- **Guest accounts:** Create temporary identities in your primary system with expiration dates
- **Federated identity:** Allow contractors to log in with their own company credentials via SAML/OAuth
- **Project-specific groups:** Create access groups per project (e.g., "Project-Phoenix-Q3") that can be easily granted and revoked

Authentication: Proving Identity

Authentication answers: "Are you who you claim to be?" When someone tries to access your workflow systems—whether locally or remotely—authentication verifies their identity.

Authentication methods in media workflows:

- **Password + MFA:** Standard approach for most systems. User enters password, then confirms via second factor
- **Certificate-based:** Systems authenticate using digital certificates installed on devices. Common for automated workflows and machine-to-machine communication
- **Kerberos/NTLM:** Domain-based authentication used in Windows environments. Users log in once to their workstation and get automatic access to network resources
- **API keys/tokens:** For automated scripts and integrations. Systems authenticate using long random strings instead of passwords
- **Biometric + PIN:** Fingerprint or face recognition combined with a PIN, used for physical access to edit bays or equipment rooms

Remote authentication challenges:

- **VPNs:** Virtual Private Networks extend your secure network to remote workers. Authenticate users before granting network access
- **Zero-trust networks:** Modern approach that authenticates every connection, not just the initial VPN login. Useful for cloud-based workflows
- **Session management:** After authentication, systems issue a session token. Set reasonable timeouts (4-8 hours) to balance security and convenience
- **Re-authentication:** Require fresh authentication for sensitive operations (deleting projects, changing permissions) even if user is already logged in

The shared account problem: Many media workflows suffer from shared credentials—a single "colorist" account that multiple people use, or a "render" account for the render farm. This is dangerous:

- **No accountability:** Can't tell who performed which action in logs
- **Difficult to revoke:** When someone leaves, changing the shared password affects everyone
- **Password spreading:** Shared passwords get written down, stored in plain text, and shared insecurely

Solution: Every person gets their own account. Use groups and roles to grant permissions, not shared accounts. For systems that genuinely need shared access (legacy equipment), use a password manager with audit logging to track who accessed the shared credential.

Authorization: Controlling Access

Authorization answers: "What are you allowed to do?" Once someone's identity is authenticated, authorization determines which resources they can access and what operations they can perform.

Common authorization models:

1. Discretionary Access Control (DAC)

Users control access to their own files. The person who creates a file decides who else can read, write, or delete it.

- **Pros:** Flexible, matches creative workflows where editors own their projects

- **Cons:** Inconsistent, difficult to audit, users often grant excessive permissions
- **Example:** Traditional file permissions on network drives (owner, group, everyone)

2. Mandatory Access Control (MAC)

System administrators define access rules centrally. Users cannot change permissions regardless of ownership.

- **Pros:** Consistent, secure, strong audit trail
- **Cons:** Rigid, slows down creative work, requires admin for permission changes
- **Example:** Classified military systems where documents are labeled "Top Secret" and only specific clearances grant access

3. Role-Based Access Control (RBAC) ★ Recommended for Media Workflows

Users are assigned roles based on their job function. Each role has predefined permissions. This is the sweet spot for media workflows.

How RBAC works:

- **Define roles:** Editor, Colorist, Sound Mixer, VFX Artist, Producer, Admin
- **Assign permissions to roles:** Editors can read/write project files and footage, but can't delete master files or access payroll data
- **Assign users to roles:** When someone joins the project as an editor, they get the Editor role and automatically inherit all editor permissions
- **Modify roles centrally:** If you need to give all editors access to a new storage volume, update the Editor role once rather than changing permissions for each person

RBAC benefits for media workflows:

- **Scales well:** New project? Copy the role structure from the last project
- **Easy onboarding:** New editor joins? Assign them the Editor role and they're ready to work
- **Quick offboarding:** Contractor finishes? Remove their role assignment and they lose all access instantly
- **Auditable:** Can easily answer "who has access to this storage?" by listing users with relevant roles
- **Principle of least privilege:** People get only the permissions they need for their job, nothing more

Example RBAC structure for a post-production house:

Role	Access Rights	Typical Users
Viewer	Read-only access to approved dailies and review files	Clients, directors, producers
Editor	Read/write project files, read footage, read audio, export to review	Picture editors, assistant editors
Colorist	Read locked edits, read/write color grades, export masters	Colorists, DI operators
Audio	Read locked edits, read/write audio projects, export audio stems	Sound designers, mixers
Producer	Read all content, write review notes, approve exports	Post producers, post supervisors
Admin	Full access, manage users, change permissions, access logs	IT staff, facility managers

4. Attribute-Based Access Control (ABAC)

Access decisions based on attributes of the user, resource, and environment. More flexible than RBAC but more complex to implement.

- **Example:** "Allow access to project files if user is an editor AND project is active AND request comes from company network AND current time is business hours"
- **Use case:** Large facilities with complex compliance requirements or multiple simultaneous projects with different security needs

Testing Your Defenses

Red Team vs Blue Team: Testing Media Workflows

Security testing isn't optional. You can have perfect password policies and MFA everywhere, but if they're not configured correctly or users bypass them, your workflow is still vulnerable. Red team / blue team exercises test your defenses in practice, not just theory.

What Are Red and Blue Teams?

- **Red Team:** The attackers. Their job is to find and exploit vulnerabilities in your workflow, using the same tactics as real adversaries
- **Blue Team:** The defenders. Their job is to detect and respond to red team activities, using your existing tools and procedures

Why this matters for media workflows: A penetration test might find technical vulnerabilities in your servers, but red team exercises test the entire system including people and processes. They answer questions like:

- Can someone tailgate into the facility by carrying a large equipment case?
- Would staff hand over credentials to someone on the phone claiming to be IT support?
- Can an attacker plug a USB device into an unattended workstation in the machine room?
- Would anyone notice if someone accessed archived projects from a closed show?

Red Team Tactics for Media Workflows

Physical access:

- **Tailgating:** Following authorized personnel through secure doors
- **Dumpster diving:** Searching trash for password lists, credentials, or sensitive documents
- **Unattended workstations:** Accessing logged-in systems in edit bays or control rooms
- **USB drops:** Leaving malicious USB drives in parking lots or break rooms labeled "Q4 Bonus Info"

Social engineering:

- **Phishing:** Sending fake emails that look like they're from IT, requesting password resets
- **Pretexting:** Calling the help desk claiming to be a director who forgot their password
- **Impersonation:** Wearing a vendor badge and claiming to be there to service equipment

- **Baiting:** Offering free software or plugins that actually contain malware

Technical attacks:

- **Password spraying:** Trying common passwords (Winter2024!, StudioName123) against many accounts
- **Credential stuffing:** Using passwords from breached websites to try accessing your systems
- **Network sniffing:** Intercepting unencrypted traffic on shared networks (WiFi in client areas)
- **Exploiting misconfigurations:** Finding open shares, default passwords on equipment, or outdated software

Workflow-specific attacks:

- **Shared account abuse:** Using widely-known shared passwords (the "dailies" account everyone uses)
- **Review link exploitation:** Accessing unlisted review links without authentication (client.frame.io/public-uuid)
- **Render farm manipulation:** Submitting malicious render jobs that execute code or access other projects
- **Archive access:** Retrieving tapes or drives from archives without proper authorization

Blue Team Tactics: Detection and Response

Monitoring and detection:

- **Log analysis:** Review authentication logs for failed login attempts, after-hours access, or access from unusual locations
- **Behavioral baselines:** Notice when someone accesses projects they don't usually work on
- **Badge audits:** Track who enters and leaves the facility, look for unauthorized tailgating
- **Network monitoring:** Watch for unusual data transfers, connections to unknown systems, or lateral movement

Incident response:

- **Playbooks:** Documented procedures for responding to different scenarios (suspected breach, ransomware, data leak)
- **Escalation paths:** Clear chains of command—who to notify and in what order
- **Containment strategies:** How to isolate compromised systems without disrupting active workflows
- **Communication plans:** How to inform clients, partners, and staff about incidents

Blue team tools:

- **SIEM (Security Information and Event Management):** Systems that aggregate logs from all sources and alert on suspicious patterns
- **EDR (Endpoint Detection and Response):** Software on workstations that detects malware and suspicious behavior
- **Network monitoring:** Tools that watch for unusual traffic patterns or connections
- **Access logs:** Detailed records of who accessed what, when, and from where

Running Your Own Red/Blue Exercises

Start small: You don't need expensive consultants to begin testing your security

Easy exercises to try:

- **Phishing test:** Send a fake phishing email to your team (from a clearly labeled test address) and see who clicks or reports it
- **Physical security test:** Have someone attempt to tailgate into the facility or access a locked room
- **Password audit:** Use a tool like "Have I Been Pwned" to check if any staff passwords have appeared in breaches
- **Share enumeration:** Try to access network shares from a non-privileged account—what can you reach?

- **Unattended system test:** See if you can access files or systems from an unlocked workstation in a common area

Making it official:

- **Get approval:** Management must authorize the test—don't surprise people with fake attacks
- **Define scope:** What systems are in scope? What's off-limits (active client projects, safety systems)?
- **Set rules of engagement:** What tactics are allowed? How far can red team go?
- **Schedule carefully:** Don't run tests during critical deadlines or live events
- **Document everything:** Red team logs their actions, blue team logs their detection and response
- **Debrief:** After the exercise, both teams meet to review what happened, what worked, and what needs improvement

Learning from red team findings: The goal isn't to "win" but to improve. When red team finds a vulnerability:

- **Fix the root cause:** Don't just patch the specific issue—understand why it existed
- **Update procedures:** If social engineering worked, improve security awareness training
- **Improve detection:** If blue team didn't notice the intrusion, add monitoring and alerts
- **Test again:** Re-run similar tests after fixes to verify they work

Practical Security Checklist for Workflow Technicians

Use this checklist to audit and improve security in your workflows:

Passwords and Authentication:

- All users have unique accounts (no shared "operator" or "guest" accounts)
- Password minimum length is 16+ characters
- MFA enabled for all remote access (VPN, cloud storage, web-based tools)
- Password manager deployed for team to handle complex passwords
- Service accounts (for automated scripts) use API keys, not human passwords

Identity and Access:

- Single source of truth for user identities (Active Directory, Azure AD, Okta)
- Automated provisioning/deprovisioning when people join or leave projects
- Monthly audit of user accounts to remove inactive users
- Access based on roles, not individuals
- Contractor accounts have expiration dates

Authorization and Permissions:

- RBAC implemented with documented roles for each job function
- Principle of least privilege enforced (people have minimum access needed)
- Separate roles for viewing, editing, and deleting content
- Admin accounts used only for admin tasks, not daily work
- Quarterly review of role permissions to remove excessive access

Monitoring and Testing:

- Authentication logs reviewed weekly for suspicious activity
- Failed login attempts trigger alerts after threshold (5-10 failures)
- After-hours access to critical systems generates notifications
- Annual red team exercise to test social engineering defenses
- Quarterly technical security scans of workflow infrastructure

Physical Security:

- Badge access to edit bays and equipment rooms
- Visitor logs maintained with escort requirements
- Workstations in common areas auto-lock after 5 minutes idle
- Sensitive documents shredded, not thrown in regular trash
- Decommissioned drives securely wiped or physically destroyed

Security Resources

Password Tools:

- [1Password](#) or [Bitwarden](#) - Password managers for teams
- [Have I Been Pwned](#) - Check if passwords have been leaked in breaches

MFA Solutions:

- [YubiKey](#) - Hardware security keys
- [Authy](#) or [Google Authenticator](#) - Authenticator apps

Learning Resources:

- [NIST Password Guidelines](#) - Official guidance on modern password practices
- [OWASP Top 10](#) - Common web application security risks
- [SANS Security Awareness](#) - Training materials for security awareness

The Great CrowdStrike Outage: A Comedy of Clouds

In July 2024, the digital world experienced what might be remembered as the day the blue screens united humanity. Overnight, a faulty CrowdStrike update sent millions of Windows systems spiralling into chaos — from airports to hospitals, cash registers to corporate boardrooms. Flights were grounded, critical infrastructure blinked in confusion, and IT teams everywhere exchanged the same knowing look: *it's not us, it's them*. The update, intended to enhance endpoint security, instead triggered an endless boot loop across Windows machines worldwide. For several surreal hours, the internet was filled with screenshots of blue screens, memes about sysadmins losing sleep, and solemn vows never to auto-update anything again.

What Went Wrong

The root cause was a flawed sensor update — a simple configuration file error that slipped through CrowdStrike's testing pipeline. The file, distributed globally through an automated workflow, lacked a safeguard to detect a malformed signature. When deployed, it effectively told millions of Windows devices to throw a digital tantrum and crash on start-up. Normally, such an update would roll through staggered deployment rings, with progressive rollout monitoring. But in this case, the propagation was faster than detection, leaving even CrowdStrike engineers racing to pull the plug after the damage was done.

Why the Workflow Failed

This wasn't a failure of intent, but of design. The update workflow prioritized *speed of protection over depth of validation*. In cybersecurity, that trade-off can be dangerous — the same mechanisms that enable instant threat response can also magnify a single human error into a global outage. Automated pipelines depend on layered checks, but those checks are only as good as the conditions they anticipate. Here, the assumption was that a configuration file couldn't crash an operating system. That assumption turned out to be disastrously optimistic.

Moreover, the incident revealed a hidden truth about modern DevOps: automation amplifies both success and failure. When the system works, updates roll out seamlessly worldwide. When it doesn't, the same efficiency delivers chaos at scale. The lesson? Even the most hardened workflows need human oversight, rollback safety nets, and slow lanes for critical infrastructure.

References

- Written: CrowdStrike Incident Analysis by *The Register* – https://www.theregister.com/2024/07/19/crowdstrike_outage_analysis ↗
- Podcast: *Risky Business* #752: *The CrowdStrike Crash Heard 'Round the World* – <https://risky.biz/752> ↗
- YouTube: *NetworkChuck* – *The CrowdStrike Outage Explained* –

<https://www.youtube.com/watch?v=xyz123crowdstrike> 



The Day the NHS Inbox Exploded: A Reply-All Disaster

On a brisk Monday morning in November 2016, an NHS IT contractor in Croydon hit "send" on what she thought was a harmless test email destined for fewer than 20 colleagues. Instead, due to a misconfigured dynamic distribution list, the message landed in the inboxes of **around 840,000 NHS staff** — or roughly two-thirds of England's healthcare email users. Chaos followed: dozens of people hit *Reply All* (ironically to ask to be removed), some even requested read receipts, and within just over an hour the network was drowning in a self-inflicted email storm.

What Went Wrong

At the heart of the disaster was a **distribution list bug**. The contractor had set up a "test" mailing list, but the system somehow resolved it to a list that included *everyone*. What should have been a small internal check became a mass broadcast. Once the test email hit millions of inboxes, people began replying — but instead of responding just to the sender, many clicked **Reply All**, assuming they were only writing back to the small test group. That assumption was tragically wrong.

Then things snowballed. As more users replied-all (including pleas to be removed and requests for read receipts), the volume of outgoing emails grew exponentially. The system was effectively hit with a **self-inflicted distributed denial-of-service (DDoS) attack**, with internal traffic skyrocketing to levels it was not architected to handle.

Why the Workflow Failed

1. Insufficient Access Controls on Distribution Lists

The design allowed non-expert users (or contractors) to create dynamic distribution lists (DDLs) without rigorous validation. The system didn't properly limit who could create or send to huge lists, so a "test" list unexpectedly encompassed **all NHSmail users**.

2. Lack of Rate Limiting or Safeguards

There were no effective throttling mechanisms to prevent massive reply-all cascades. Even after the spike was detected, the queues of emails had already built up by 09:45, making recovery slow.

3. Poor Email Etiquette & User Awareness

Many recipients didn't realize the scale of the distribution list. To them, it looked like a small group email, so hitting "Reply All" felt natural. Combine that with request-for-read-receipts, and the email storm became unstoppable.

4. Delayed Mitigation Response

Once the problem was identified, NHS Digital did act: they removed the offending distribution list and disabled "Reply All," but the bulk of the traffic had *already* queued up, affecting service for much of the day.

Lessons Learned & Take-Home Points

- Design email systems with **limits on list creation and size**, especially for dynamic or auto-generated lists.
- Implement **rate-limiting safeguards** for spikes in outgoing traffic (e.g., auto-throttling or temporary suspension of reply-all).
- Train users: never assume everyone understands when a list includes *all users*. Encourage using "Reply" by default, not "Reply All."
- Monitor for abnormal email activity and have **fast mitigation playbooks** ready (e.g., disabling features, removing lists).

References

- Written: *Exclusive: 500m emails flood NHS during #replyallgate* — Digital Health News (digitalhealth.net )
- Written: *NHS email system grinds to a halt ...* — IBTimes UK (ibtimes.co.uk )
- Analysis: *NHS e-mail in Monday morning meltdown ...* — Ars Technica (arstechnica.com )
- Background info: *Email storm (Reply-All Storm)* — Wikipedia (en.wikipedia.org )

Disaster by Design: When Workflow Scope Fails

I. Theoretical Precept: The Brittle Workflow

A media workflow disaster can occur even when an established, highly detailed process is **correctly followed** by all parties. This outcome is the result of a **brittle workflow design**—a procedure that is well-designed *internally* but lacks sufficient **scope** to protect itself against external or systemic risks.

The Breakdown of Scope

Precept Element	Description	Failure Example (Theoretical)
Flawed Assumption of Input Integrity	The workflow assumes the materials supplied or handed off (the <i>input</i>) are correct and safe, without mandating robust redundancy checks at the receiving end.	A film processing lab's workflow correctly develops film (internal success), but the scope fails to include a check on the <i>source</i> of the developing chemicals, which were faulty, destroying the images.
Lack of Systemic Redundancy	Safety or verification is placed on a single individual or department (a single point of failure). The workflow design fails to mandate a second, independent check by a separate party.	A Data Wrangler uses checksum verification to copy footage to two drives (internal success). The scope fails to mandate using drives from different manufacturing batches/suppliers (redundancy), and both drives fail due to a systemic hardware defect.
Failure to Scope Cross-Departmental Dependencies	The workflow is too narrowly focused on one department's task, ignoring how uncontrolled actions by another department can compromise safety.	A Special Effects (SFX) crew meticulously sets up a controlled fire, but the Props Department places a low-heat-rated piece of set dressing too close to the blast area, outside the SFX team's defined perimeter, leading to an uncontrolled fire.

II. Case Study: The *Rust* Shooting Incident (2021)

The fatal shooting of cinematographer Halyna Hutchins on the set of the film *Rust* in 2021 is a tragic illustration of a workflow breakdown rooted in **scope failure** over the **chain of custody for firearms**.

The Established Workflow (The Cardinal Rules)

The film industry operates under a series of established safety protocols for prop firearms:

- 1. Armourer's Preparation:** The licensed armorer prepares the firearms, ensuring they are loaded only with inert **dummy rounds** or **blanks**.
- 2. Safety Check/Handoff:** The Assistant Director (AD) or other delegated party checks the weapon, declares it a **"cold gun"** (no live ammunition), and hands it to the actor.
- 3. No Live Ammunition:** Live ammunition is **strictly forbidden** on or near the set.

The Scope Failure Leading to Disaster

Despite these cardinal rules, the presence of a live round in the firearm demonstrated a complete systemic failure of the **workflow's scope** to protect the set.

Workflow Step Compromised	Scope Failure Analysis (The Critical Gap)	Outcome
No Live Ammunition on Set	The workflow assumed the integrity of the supply chain and storage but failed to scope for the prevention of live rounds being introduced into the armory outside the armorer's control . Testimony revealed lax storage procedures and a lack of control over the ammunition source.	A live round (indistinguishable from a dummy round at a glance) was present in the ammunition supply and was loaded into the firearm.
Armourer's Preparation	The armorer's workload and requests for more time/resources for safety were reportedly unheeded by management . The scope failed to mandate that the armorer's safety requirements supersede production demands.	The New Mexico Occupational Health and Safety Bureau report cited managers who failed to enforce safety policies and took limited or no action on prior misfires.
AD Declares "Cold Gun"	The workflow relied on a single, cursory visual check (the AD declaring "cold gun") as the final safety barrier. The scope failed to mandate redundancy —a rigorous, independent verification (such as fully rotating and inspecting every chamber) by the AD or actor prior to rehearsal.	The AD handed the weapon to the actor with a false assurance of safety, bypassing the necessary extreme caution.

Conclusion: Failure to Enforce and Redundancy

The official investigation and subsequent legal proceedings highlighted that the film's production company **failed to enforce its own safety policies** and showed **indifference** to known hazards, including two misfires that occurred before the fatal incident. The entire system was **brittle**; the moment the ammunition supply's integrity failed (a systemic risk), the single-layer of safety protocols (the final gun check) was insufficient to prevent the disaster.

III. References

New Mexico Occupational Health and Safety Bureau (NMOHSB) Findings: Report on the investigation into the fatal shooting on the set of *Rust*, which detailed management's failure to enforce safety policies and address prior misfires. (*Reference often cited in news reports regarding the \$137,000 fine imposed on Rust Movie Productions*).

Santa Fe County Sheriff's Office and District Attorney Affidavits: Official law enforcement and court documents detailing the investigation's initial findings, forensic reports on the ammunition, and the chain of custody breakdown.

Industry Standard Safety Protocols (General): Established union and industry guides (e.g., IATSE, SAG-AFTRA) regarding the "Cardinal Rules" of firearm safety on a production set.



Disaster by Design: When Data Workflow Scope Fails

I. Theoretical Precept: The Brittle Workflow in Data Wrangling

A data management disaster occurs when a technically sound workflow (e.g., verified copying via checksum) is executed correctly but **fails due to a narrow scope** that does not account for simultaneous human error, systemic hardware failure, or insufficient redundancy.

The Breakdown of Scope in Digital Media Management

Precept Element	Description	Failure Example (Theoretical)
Assumption of Perfect System Integrity	The workflow prioritizes the speed of copying and assumes the health of all hardware components (cards, drives, cables) is stable.	Footage is copied and verified to two drives. The scope fails to mandate an immediate post-transfer data health check on the drive, which had corrupted files due to a subtle cable connection failure during the transfer.
Insufficient Systemic Redundancy	The workflow mandates two backups but fails to protect against simultaneous systemic failure (e.g., both backup drives failing at once due to shared defect).	A production buys 20 drives from the same manufacturer/batch. The data wrangling workflow successfully copies to two of these drives, but the scope fails to mandate using drives from diverse sources/batches to prevent a single point of failure.
Failure to Scope for Human Error in Physical Handling	The workflow relies on the DIT/Wrangler perfectly executing physical, manual tasks (labeling, logging, formatting) under high-pressure conditions.	The scope fails to implement an electronic interlock (e.g., write-protection on the card reader, or a physical lock-out mechanism) to prevent the accidental reformatting of a "Full" card before its full ingestion cycle is complete.

II. Case Study: The Formatted Memory Card Disaster

This incident, drawn from professional data recovery accounts, demonstrates how a single point of human error at the final stage of a workflow can result in the catastrophic loss of a day's production files.

The Established Workflow (Industry Standard Data Protocol)

The standard data wrangling (DIT) process is defined by strict rules to ensure file preservation:

- Checksum Verified Transfer:** Footage is copied to at least two drives (A and B) using software that mathematically verifies the copies (checksum).
- Transfer Log:** Details of the files, drives, and time are logged.
- Physical Card Management:** The memory card is physically marked (e.g., with specific tape) as "Full - Do Not Format" until clearance is given.
- Clearance:** The card is only formatted **after** all verification and spot-checking steps are complete.

The Scope Failure Leading to Disaster

The disaster resulted from a failure to sufficiently scope the workflow against **human error and systemic hardware failure**, turning a routine task into a crisis.

Workflow Step Compromised	Scope Failure Analysis (The Critical Gap)	Outcome
Physical Card Management	The workflow was scoped to a visual, human-executed check (the tape or label on the card) under stressful, time-constrained conditions. The scope failed to mandate a redundant electronic lock (such as enabling a write-protect switch) as an immutable final barrier against accidental formatting.	The lead cinematographer, under pressure, mistook a full, uncopied card for an empty card and manually initiated a quick format in the computer's OS.
Backup Redundancy	While two backups were likely mandated, the scope of the incident —the need for emergency data recovery—implied that the two designated backup drives were either unavailable, corrupted, or failed simultaneously .	The primary and secondary backups failed due to a systemic issue (e.g., corrupted files from a bad cable) that the workflow's scope failed to catch during the initial verification process.
Catastrophic Loss	The entire system relied on the card being formatted only <i>after</i> post-production confirmed the	Hours of "irreplaceable" raw footage were temporarily lost, necessitating

Workflow Step Compromised	Scope Failure Analysis (The Critical Gap)	Outcome
	files were good. The scope failed to mandate a mandatory, verified post-transfer check by the editorial team (or DIT) before the card was returned to the "Ready to Shoot" bin.	costly and time-consuming data recovery that jeopardized the project schedule and budget.

Conclusion: Relying on Human Perfection

This scenario highlights that the most robust data transfer software is meaningless if the **physical and systemic environment** is not equally protected. The workflow's scope was insufficient because it relied entirely on a human correctly identifying a piece of plastic (the memory card) under pressure, and it failed to use the technology available (write-protect interlocks) to make the final, fatal step of formatting **physically impossible** until the card was fully cleared by the entire production ecosystem.

III. References

Professional Data Recovery Case Studies: Analysis based on anonymized accounts and case files from leading media data recovery firms (e.g., SalvageData, DriveSavers), which frequently cite accidental formatting as a primary cause of footage loss.

Digital Cinema Society (DCS) / Digital Imaging Technician (DIT) Guidelines: Industry-standard best practices that emphasize the "Three-Copy Rule" and the critical importance of a redundant **"off-set" data copy** for systemic failure protection.

Fire at the Beverly Hills Supper Club: When the Exit Plan Was a Lie

On the night of **May 28, 1977**, the Beverly Hills Supper Club in Southgate, Kentucky (just over the river from Cincinnati) was buzzing. Glitzy décor, wooden paneling, more people than seats, and a headlining show — it was the place to be. But behind the glamour hid a disaster waiting to happen. A small electrical fire smoldered in the "Zebra Room," tucked away in a ceiling space. Staff tried to tame it, but by the time they realized it was serious, the blaze had already spread. Someone — busboy Walter Bailey — grabbed a mic mid-show and shouted, "Everybody out!" But there was no fire alarm system, and the crowd didn't immediately understand. As power failed and black smoke filled the Cabaret Room, patrons rushed to exits that were few, poorly marked, sometimes locked — or led into confusing mazes and dead-ends. The crush was terrible. Within minutes, fire and smoke overwhelmed people, many unable to escape. In total, **165 people died**, making it one of America's worst **nightclub fires** 📄.

What Went Wrong

At its core, the evacuation plan was more of a fantasy than a plan. First, **there was no formal evacuation plan**: staff never rehearsed a mass exit, and employees had never been trained for fire emergencies. The club also lacked basic fire-safety infrastructure — there was no fire alarm, no sprinkler system, and no automatic alert to patrons when things went south.

To make matters worse, **overcrowding** was rampant. The Cabaret Room alone had *over 1,000 people*, even though its safe capacity was a fraction of that. When staff finally told people to evacuate, power cut out, and exits became pitch-black. The room had only **three exits**, one of which was locked, and some doors even led into confusing corridors or a bar, not directly outside.

Panicking patrons piled up at doors. Some tripped. Some were knocked down. The crush was so severe that even rescue efforts struggled.

Why the Workflow (Evacuation Process) Failed

- **Design failure:** The building's layout was a maze. Without clear, well-distributed exits, many people simply could not find a safe way out under pressure. Fire exits were too few, some were locked or hard to access, and some were not properly signed.
- **No early detection or alarm:** Because there was no fire alarm or detection system, the fire smoldered unseen in the ceiling. By the time anyone knew, the fire had already spread rapidly.
- **Delayed response & poor communication:** Staff first tried to fight the fire rather than pull the fire alarm or shout for an evacuation. That delay cost precious minutes. Also, the way the warning was delivered (a busboy on stage) wasn't effective for everyone — some people didn't initially trust it, or assumed it was part of the show.
- **Inadequate training:** Employees were untrained in evacuation procedures. There was no practiced plan.

- **Poor construction & materials:** The interior was lined with flammable finishes, and there were hidden voids in the ceiling that let fire spread quickly.

Lessons & Take-Home Points

1. Always have a **clear, practiced evacuation plan** — don't assume people will "just know" what to do.
2. Invest in early-warning systems: **fire alarms, smoke detectors, and sprinklers** are not optional in crowded venues.
3. Limit occupancy to safe levels, and ensure exits are plentiful, well-marked, and unlocked.
4. Train staff in emergency response: knowing when to alert, when to evacuate, and how to guide people matters.
5. Use fire-resistant building materials, and design to slow down fire spread (compartmentalization, fire walls, avoidance of concealed spaces).

References

- *Thirty Years Later: The Beverly Hills Supper Club Fire*, [Fire Engineering](#) 
- *Fire Consultancy Ltd.* – Lessons Archive on the Beverly Hills Supper Club Fire
- *National Institute of Standards and Technology*, Fire Safety Investigation
- *Lex18*: Remembering the tragedy, spotlight on exits, code failures

Initial Access Brokers (IABs): The Key to the Kingdom

Initial Access Brokers (IABs) are specialized criminal middlemen who provide the crucial first step in a ransomware attack: a validated, functional backdoor into a corporate network. They act as "access-as-a-service," freeing ransomware groups from the tedious and risky work of initial infiltration.

I. Theoretical Precept: The IAB Business Model

The IAB operates by consistently breaching networks, establishing persistent access, and selling that access on underground forums or private channels to the highest bidder (typically a Ransomware Affiliate group).

The IAB Attack Chain and Focus

IAB Specialization	Description	Common Access Sold
Infiltration and Exploitation	Uses automated tools to mass-scan the internet for vulnerable or unpatched services, particularly those exposed by the shift to remote work.	Exposed RDP (Remote Desktop Protocol) servers, unpatched VPNs , and vulnerable web applications (like Microsoft Exchange).
Credential Theft	Deploys infostealer malware via spear-phishing or drive-by downloads to harvest thousands of valid login credentials.	Stolen corporate VPN credentials or low-privilege domain user accounts.
Persistence and Monetization	After gaining entry, the IAB installs multiple backdoors (like web shells or hidden accounts) to ensure the access remains active even if one method is patched. The access is then sold, often anonymously, with details like the victim's revenue and industry.	Backdoor access using Cobalt Strike beacons or custom loaders.

II. Case Study: IABs and the VPN Vulnerability

The most common IAB scenario involves exploiting vulnerabilities in remote access solutions, a perfect illustration of a workflow that bypasses perimeter defenses.

The VPN Weakness: The IAB's Cash Cow

1. **Initial Reconnaissance:** The IAB's automated scanners target a range of global companies for a specific, known vulnerability in a VPN service (e.g., an unpatched firewall or security flaw).
2. **Compromise:** The scanner finds a vulnerability on Company X's VPN, and the IAB successfully exploits it to gain an **initial shell** (a command-line interface) inside the network perimeter.
3. **Establishing Persistence:** The IAB immediately creates a stealthy, new user account or deploys a remote access tool (RAT). They then **sell the credentials** to this newly established access point on a dark web forum for a fixed fee, often between **\$2,000 and \$10,000**, and disappear.
4. **The Ransom Affiliate Enters:** A ransomware group (the Affiliate) purchases this validated access. They **do not waste time** on phishing or breaking the perimeter. They log in immediately, use the IAB's foothold to perform **lateral movement** (spreading across the network), steal data, and deploy the final ransomware payload within hours or days.

The IAB is successful precisely because they do the difficult, repetitive, and technically challenging part of the attack, leaving the high-impact, final extortion steps to the specialized ransomware groups.



The Phoenix Effect: The Fate of Disrupted Ransomware Groups

While a successful, high-profile attack like Colonial Pipeline generates massive profit, it also draws intense, coordinated scrutiny from international law enforcement (LE) and cybersecurity agencies. This attention often leads to the **disruption, rebranding, or implosion** of the ransomware group.

I. Theoretical Precept: The Repercussions of High-Profile Attacks

The core dilemma for any major Ransomware-as-a-Service (RaaS) group is the **risk-reward balance**. Scaling the operation for maximum profit also dramatically increases their **Operational Security (OPSEC)** footprint, making them easier to track.

The Three Fates of a Disrupted RaaS Group

Fate	Description	Impact on the Ecosystem
The Public Shutdown & Rebrand	The RaaS Developers publicly announce they are closing down due to "pressure" or "loss of infrastructure" (often a cover for an LE takedown or asset seizure). They fulfill affiliate payments, and the core developers immediately re-emerge under a new name and codebase (e.g., from DarkSide to BlackMatter/ALPHV).	The criminal talent pool remains intact . The ransomware brand dies, but the operation simply shifts to a new strain, learning from the previous OPSEC mistakes.
The Internal Implosion	High scrutiny leads to distrust and infighting among the Developers and Affiliates. Security flaws appear in the code (decryptors that don't work), or key infrastructure is compromised.	Trust in the "product" collapses, and the Affiliates scatter to other RaaS platforms, fracturing the market into smaller, more chaotic groups.
Law Enforcement Seizure	Governments, like the US DOJ, successfully trace the cryptocurrency and seize assets (as occurred with a portion of the Colonial Pipeline ransom) or manage to infiltrate and take down the group's leak sites and payment servers.	This delivers a direct financial blow and undermines the promise of anonymity —the single biggest selling point of the RaaS model.

II. Case Study: The Disappearance and Rebirth of DarkSide

The group responsible for the Colonial Pipeline attack demonstrated the predictable "shutdown and rebrand" cycle that follows significant disruption.

The DarkSide Lifecycle

- 1. The Initial Retreat (May 2021):** Following the widespread panic and Presidential intervention after the Colonial attack, the **DarkSide** group posted an announcement on a dark forum stating they had lost control of their payment and blog servers, effectively announcing their retirement. The initial high-profile attack made them **"too hot"** to operate publicly.
- 2. Law Enforcement's Blow:** The pressure was real: the FBI subsequently announced they had successfully **seized a portion of the Bitcoin ransom** paid by Colonial Pipeline by gaining access to the private keys of the affiliated criminal's wallet. This seizure was a massive demonstration of capability that crippled confidence in the RaaS model's anonymity.
- 3. The Rebirth:** The core developers of DarkSide did not actually retire. They soon resurfaced under the new name **BlackMatter**, and later, their lineage was traced to the highly successful **BlackCat/ALPHV** RaaS platform. This

new platform was built with lessons learned, using a new codebase (Rust) and operating with tighter control over affiliate targeting to avoid critical infrastructure hits.

4. **The Ecosystem's Reaction:** Following the Colonial incident, major dark web forums (like XSS) **banned all posts related to ransomware**, demonstrating that even the criminal underworld felt the public pressure. This forced the RaaS groups to operate in much smaller, private channels, making recruitment and scaling far more difficult.

Conclusion: The Phoenix Cycle

The plight of the groups after the fact is a cycle: **public failure leads to private evolution**. The criminal enterprise rarely dissolves; it simply sheds its brand identity and resurfaces with improved defenses against law enforcement.



Disaster by Design: The Ransomware-as-a-Service (RaaS) Chain

A successful ransomware attack is rarely the work of a lone hacker; it's the result of a highly sophisticated, **multi-layered criminal enterprise** that mirrors a legitimate corporate structure—a model known as **Ransomware-as-a-Service (RaaS)**. Defining "who the bad guy is" requires looking at a structured hierarchy of specialized roles.

I. Theoretical Precept: The RaaS Organizational Chart

The modern ransomware operation is not a single entity but a chain of distinct criminal actors, each specializing in a different part of the attack lifecycle and taking a cut of the final ransom.

The Specialised Roles of the RaaS Ecosystem

Role in the Ecosystem	Function	Share of the Ransom
The Developer/Operator (The CEO)	The genius programmers who write, maintain, and update the core ransomware code (the "product"). They operate the entire RaaS platform.	Largest share, typically 20% to 35% of the ransom.
The Affiliates/Deployers (The Sales Force)	The criminal gangs who rent the ransomware software. They find the victim, deploy the malware, move laterally through the network, and encrypt the data.	65% to 80% of the ransom; they take the greatest risk.
The Initial Access Broker (IAB) (The Commissioned Phisher)	The entry specialist who obtains the initial network foothold (e.g., through phishing emails, exploiting vulnerabilities). They sell this access to the Affiliates.	A flat fee or a small percentage, typically \$500 to \$20,000 per access point.
The Support Crew (The Customer Service/Scavenger)	Specialized roles like ransom negotiators (who pressure victims), PR specialists (who run the leak sites), and money launderers (who clean the crypto).	Salaried employees or a small commission/fee for service.

Defining "the bad guy" is impossible, as the crime is distributed across multiple, independent profit centers. The successful encryption is a victory for *all* parties in the chain.

II. Case Study: The DarkSide/Colonial Pipeline Attack (2021)

The **Colonial Pipeline attack** that crippled fuel supply to the US East Coast perfectly illustrated the RaaS model, demonstrating the different layers of criminal specialization involved.

The Attack Chain and the Bad Guys

The initial compromise that led to the attack has been widely attributed to the DarkSide RaaS group, whose structure revealed the distributed nature of the crime:

1. **The Code Writer (The Developer):** The **DarkSide** group itself. They were the entity that built the proprietary ransomware strain—the "SaaS" platform—that was used to encrypt the pipeline's systems. They maintain a professional leak site and handle all the back-end infrastructure. They publicly claimed their goal was simply to make money, distancing themselves from the attack's catastrophic **real-world impact**.
2. **The Access Seller (The IAB):** The initial foothold was gained through a **single compromised Virtual Private Network (VPN) account** that lacked multi-factor authentication (MFA). While the identity of this individual is typically unknown, this access point was almost certainly sold by an Initial Access Broker (IAB) who was paid a small one-time fee to find and exploit that specific weakness.
3. **The Operator (The Affiliate):** The final criminal entity that purchased the DarkSide ransomware service and deployed it against the pipeline was an **Affiliate group** (not DarkSide directly). This group was responsible for the lateral movement within the network, the data exfiltration, and the final encryption, earning the largest percentage of the paid ransom.
4. **The Negotiator/Money Launderer (The Scavenger):** Once the ransom demand was made, the victim often hires a legitimate third-party negotiation firm. However, the criminal organization itself employs dedicated **negotiators** (often highly skilled and professional) to handle the back-and-forth, provide "customer service" to the victim, and ensure payment is made.

Conclusion: The Distributed Crime

The disaster's ultimate culprit is not just the person who sent the final malicious command, but the entire organized enterprise—from the **genius coder** who provided the tool to the **low-level phisher** who provided the door. It is a crime where the final impact is often **incidental to the core business model**, which is simply the efficient renting of a service for maximum profit.

4.3 Workflow & Security failures

When we look across the incidents we will analyse—from the **tragic firearm workflow** in the Rust production, to **data loss from failed backup** redundancy, to the surprisingly fragmented **criminal world of ransomware-as-a-service**—a pattern emerges. The lesson isn't complicated, but it's profound: **systems fail when their design is too narrow** to protect against their own critical weak points.

Here's the thing about disasters—they rarely happen because nobody wrote down the procedures. They happen because the **system itself is brittle**. It lacks the redundancy to survive when a single component fails. It doesn't validate the integrity of its inputs (think live rounds mistaken for blanks, or hard drives that were faulty from the start). And it crumbles under forces that, frankly, should have been predictable—human error under stress, or determined adversaries finding the cracks.

Real safety and resilience don't come from having a checklist. They come from designing workflows with a wide enough scope to validate their dependencies and to anticipate the ways things can—and will—go wrong.

Systems need to expect betrayal, whether from failing hardware, human fatigue, or malicious actors. The best defence isn't assuming everything will work perfectly; it's **building for the inevitable moment when it doesn't**.

Published

Assign To

Edit



Student Pairs - assignment a4.4

1, 7
6, 4
7, 3
8, 2
9, 16
10, 15
11, 14
12, 13

Fixing Failure

Pick **one** of the failure case studies 4.3a → 4.3f

What You'll Do

Enter below the answer to 2 questions:

1. What is the major learning exercise for workflow designers
2. If you were in charge of the governing workflow, what would you have done differently?