Bespoke Software

Volunteer Management
Data Management and Security

March, 2012

William Cornett
Lead Developer
Bespoke Software, Inc.

Introduction

Volunteer management means managing a lot of information. Part of that management is a concern about the privacy, security, accessibility and control of that information. This document describes some of the considerations and best practices I've found while working with volunteer organizations around the world, especially in the United States and Canada. Specifically I'll be focusing on four important aspects of data and the systems you use to access and manage your information: Privacy, Security, Reliability/Availability and Ownership/Control.

1. Privacy

In both the United States and Canada there are laws regarding the management and disclosure of personal information. In the US it's primarily HIPAA, since there's usually medical data about your volunteers (e.g. TB tests); in Canada a big concern is the storage and transport of that data through the United States since there it's subject to USA Patriot Act disclosures.

2. Security

Information must be protected from unauthorized access both from within (your staff) and from without (vendors, hackers and undesirables). When it's your staff, who have a legitimate right to be in the system, you have the further considerations of restricting who and what they have access to, as well as tracking what they do (audit logs).

3. Reliability and Availability

You need your data and the systems around it to be reliable – will they be there when you need them? Will it be accurate?

4. Ownership and Control

Who really owns your data? You do, but how much is that worth if you're unable to exercise day-to-day control over it? What happens if something goes bad at your vendor, or between you and your vendor, if you don't have that control?

Privacy

Every day we disclose pieces of our personal information to others and we hope that they respect our trust by keeping that data secure and not disclosing it to others. We are in the position of being the ones trusted and it's our duty to protect the private information of others. As part of privacy I'll be talking about two sub-topics: software tools and vendors/hosting.

Software tools

To protect that information we use a series of software tools as part of our overall volunteer management systems.

Internally

For staff, and other trusted people who are given authorized access to our data systems, access rights define what those individuals are allowed to see and do. Access rights, like good fences, keep people away from what they should not see or use. These rights are usually defined in two ways: role-based and individually; in some cases you may use both of these mechanisms together.

By defining generic roles, e.g. "Administrator", "Staff", "Data entry temp", you can avoid having to define rights on a person-by-person basis and this means that your settings are more likely to be correct. This also means that when you edit the rights of a role, the users in that role are automatically updated. Use role-based permissions where you can.

Does your system support only a built-in set of fixed roles? Or is it flexible enough to let you define roles of your own? For a small organization you can probably get by with just a few simple roles; for a large system you may need to define a whole series of them giving permissions to data on a site-by-site basis.

Where your system supports it, use fine-grained settings on security. If a user just does data entry of new applications, there's not much reason for him to be exporting data or running security reports.

External threats

When you're exposing your system over the Internet as a web application, you are putting a sign on your door which says "Hey world, I've got good data in here." Make sure that you have good locks.

For a vendor-hosted system, what do they use to protect your data from those threats? Ask some questions:

- Are all communications done via encrypted tunnels (SSL, IPSEC VPN, etc.)?
- Is the data for different organizations mixed in together in one database? If so, what controls exist to ensure that no other organization sees the data for another in case of a programming or database error?
- Are systems firewalled? Can IP address restrictions be put in to limit who can access the system?
- Are backups stored off-site, and are they encrypted?
- What systems are in place to detect intrusions and to limit the damage if one occurs? Are there guarantees of customer notification if a system is compromised?
- Are vendor employees bonded, and have they signed binding non-disclosure agreements? Better yet: are there controls in place which make it impossible for them to access customer data?

For a self-hosted system,

- Do all communications via encrypted tunnels (SSL, IPSEC VPN, etc.)
- Use proper firewalls. Restrict access by IP address where possible.
- Keep the operating system and its components (IIS, Apache, OpenSSL, etc.) up-to-date and patched.

Vendors and Hosting: Self-hosted vs. Cloud

If you host your own system, you have the necessary controls over data access by vendors, their staff and their suppliers. It means you're responsible for your own firewalls and access systems, but you can fully control them as fits your needs and know that no outside entity can access your data without permission.

For a cloud-based system the opposite is true: you outsource the responsibility and controls with the expectation that your vendor will handle these for you appropriately.

United States

The primary concern in the US isn't where your data is hosted geographically but who can access it where it is. HIPAA regulations control everything about any data which contains medical history data, even TB tests tracked for your volunteers. Solutions for US companies:

- Self-host your data, or
- Choose a vendor whose systems can be audited and monitored. Check that they use modern (strong) encryption so that their staff can't read your information and that backups, if compromised, would be useless to their recipients due to the data being encrypted.

Canada

For Canadian organizations, cloud-based solutions can be a problem. Data stored in the US is subject to disclosure under provisions of the USA Patriot Act, and your volunteers need to know that. This isn't something that can be covered under a non-disclosure agreement (NDA): the vendor can be compelled under law to provide any requested data to the US government and can be specifically prohibited from telling you that this has happened!

Solutions for Canadian companies:

- Self-host your data, or
- Choose a Canadian vendor who can guarantee that the information does not get hosted, transferred or backed up onto systems in the United States.

Best practices (Privacy)

Keep detailed audit logs. Systems which track the actions of users, from "what records they viewed", "whose data they edited", "when they ran reports" and "did they forget to log out" let you see what your staff members are doing and can help you answer questions even six months down the line. Set up reports beforehand to extract these logs and they'll be ready when your compliance department comes asking.

Avoid shared accounts. Your audit logs are worthless if they can't identify a single individual associated with specific actions because three people share the same account. What happens when one of those people leaves and still knows someone else's password?

Use Active Directory integration for authentication. If the user account in your volunteer system is linked to their AD account, then if a person leaves their AD account is disabled or deleted and they can no longer access the volunteer data even if you forget to remove them from that system.

Enable account lockouts. If a staff member gets her password wrong three times in a row, there may be an innocent explanation – and they can tell you what it is because their account was locked out after they did it.

For touch screen/kiosk systems, put passwords on volunteer logins if there is private information available in volunteer accounts. Better yet, use a card swipe, barcode or biometrics login to make the sign-in process fast, reliable and more secure.

Test. After you've set up your users and their access rights, log in under their role and check: do they see anything you don't want them to see? If they perform an unauthorized action, do you know where to find a list of what they've done and when?

Report. Regularly report a list of all people and their access rights, especially those with elevated/superuser system rights; should they still have these rights?

Audit your staff. Twice per year look at what each individual has done in the past week. Look for any suspicious behavior (logging in at odd hours or from unusual locations). Check to see if they're opening records or running reports that they may have access to but probably shouldn't be using.

Monitor accessibility. For a cloud-based system, or if you host a web-based system on your own hardware, be aware of how that data can be accessed from outside your facility. Can (should?) your staff be able to use that system from home? Remember that touch screens and kiosks can expose information as well: ensure that they're only available on authorized systems within your facility. Just because the URL to your web-based kiosk system isn't public, don't assume that it is *secret*.

Security

Information must be protected from unauthorized access both from within (your staff) and from without (vendors, hackers and other undesirables). When it's your staff, who have a legitimate right to be in the system, you have the further considerations of restricting what access they have as well as tracking what they do (audit logs).

There's a considerable amount of overlap between Privacy and Security: you can't implement privacy without security!

Encryption

> *It is insufficient to protect ourselves with laws; we need to protect ourselves with mathematics.*
> - Bruce Schneier, noted security researcher and consultant

In cryptography, encryption is the process of transforming information using mathematical tools to make it unreadable to anyone except those possessing the right key. It's your last, best line of defense against intrusions from within or without. Proper use of encryption protects your data. Even if someone walks off with your laptop, if the data on it is encrypted, you have to buy a new laptop – but you don't have to notify 4,000 volunteers that their personal information may have been compromised.

There are many different ways of implementing encryption, and it can be done at multiple levels. For example, you can encrypt your laptop's hard drive, and having done that none of the data on that machine can be accessed if someone steals it. That's good. But your volunteer data is probably stored in a database somewhere, and not on your individual machine.

If you host your own system, you can have the database encrypt data for you. For example, SQL Server can encrypt its files so stolen backups or lost equipment doesn't necessarily mean that the data is compromised. Anyone with legitimate access to the database, though, isn't affected because the database is decrypting the data for them. At the application level you may want another layer of encryption. For example you'll likely want to encrypt any details about a volunteer's background check results, and any private information about medical tests like TB tests. All of your users can access the database, but when they try to open up a background check or a TB test they'll be stopped in their tracks. Even better: breaking into the SQL Server won't help them, since SQL Server doesn't know how to decrypt that data.

Finally, you need to look at how the data is delivered. If you're on your local network you probably don't need the overhead of over-the-wire encryption. Across the Internet, however, all of your communications with the database should be encrypted! (Your browser will use SSL, any applications should be tunneled through a proper IPsec VPN connection.)

Things to look at when checking out encryption in a volunteer system:

- Is it encrypted at the database level, or can it be?
- Can it be encrypted at the application level as well?
    - If it's encrypted at the application level, what kind of encryption is used? Asymmetric encryption lets anyone with the proper security rights write certain pieces of information like background check requests and results, but they can't get the data back again later. This is great for letting your data entry people enter or scan forms but then have no way to get back to see them later.
    - With application encryption, can you revoke individual user's keys? This is like smartcard systems: even if a staff member walks off with their key after leaving, the security system won't respect that key.
    - Is encryption separate from security? If so then even users who find themselves with rights they shouldn't have still can't get to the most sensitive data.
- How are electronic communications handled? Can you configure this yourself?

Verifiability

"Security by obscurity" is probably the most common error in trying to secure information. If they (attackers) don't know how our data is stored, or how our encryption works, they can't access it. Right? Right???

All of the best security systems, especially encryption tools, are open. This means that you or your consultant can look at what the vendor has done and see for yourself if it's a real solution or snake oil. It means that security professionals can audit that system and spot the problems. The CIA, NSA and other acronym agencies – the most paranoid people in the world – only use public algorithms like AES and RSA which have been vetted by the best minds. Even the CIA and NSA can't break AES or RSA, and that's why they trust it.

When evaluating a system, check into customizability on the database and application: if you can see the database schema, if you can define your own database access rights by user, workstation and physical location, you can enhance your security and monitor the systems to whatever extent you decide that you want to.

Permissions

Does your system support only a built-in set of fixed roles or is it flexible and lets you define roles of your own? For a small organization you can probably get by with just a few simple roles; for a large system you may need to define a whole series of them giving permissions to data on a site-by-site basis.

Where your system supports it, use fine-grained settings on the security. If a user just does data entry of new applications, there's not much reason for him to be exporting data or running security reports.

Best practices (Security)

Use encryption. Even if your physical systems are compromised, or your outside vendor goes rogue, if your most sensitive data is encrypted and they don't have the key, that data is still safe.

Trust but verify. Insist on seeing the specifications of your vendor's systems. Will they let you see the database schema? Look at their encryption (under non-disclosure, of course)? If not, there may be a reason.

Availability

"Please don't
bring the system down for maintenance *or* update the software *or* try something new
just before
spring registration *or* the big event on Saturday *or* when I'm on vacation and can't be there to fix it"

You need your data and the systems around it – will they be there when you need them? Will it be accurate? If your volunteer database is down, or it's wrong, you have:

1. Unhappy volunteers who can't sign in and out.
2. Unhappy staff who can't do scheduling, run reports, and have to deal with unhappy volunteers.
3. Unhappy administrators who are upset about #1 and #2 above *and* who just got word that Joint Commission is doing an audit today.

How much availability is enough? If you're managing 300 volunteers and they mostly take care of themselves, and if your database is down for maintenance or the vendor's 'net connection is offline for a few hours, their signing in and out on paper may be enough until you can fix the problem. With 2,000 volunteers on six campuses and no time to "fake it", your system needs a bit more planning to be sure it's up when you need it. And there's that matter of surprise audits, of course.

Self-hosted

When you manage your own systems, you have the control. And responsibility too, of course. Based on your needs you'll want to consider:

- Failover hardware (if one server goes down, another takes its place).
- Database clusters (multiple database servers all kept up-to-date together).
- For a system that's visible externally, redundant Internet connections.

If someone else is hosting your database

Ask questions. Whether these matter and to what degree all depend on how much availability you need.

- Do they use major professional co-location facilities like Rackspace or Amazon's EC2/EWS?
- Do they have failover systems? Are they on a multi-homed network? Backups?
- Do they have reliable power systems? Alternate hosting facilities?

Best practices (Availability)

Decide early. How much reliability and availability do you need? Choose your software and hardware accordingly.

Test the numbers. Your hours reports and volunteer counts for the year aren't due until January, but run the reports in April, July, October and December. If there's a problem – whether from data entry problems, an import gone awry or a corrupted database due to a disk malfunction – it's a lot easier to fix the issue when you've got time.

Check your backups. Murphy's Law says that your backups will complete every night without error, and that when your server crashes the tapes will be lost or the files corrupted. Get your IT department to do a test restore of your data once every few months.

Have a fallback plan. If you're using an outside vendor, have exports (or even printouts) of your most critical data.

Talk to your vendor. If they manage your data, do they have good backups? Do *they* test those backups? Even better: will they share them with you?

Ownership and Control

How long have you had your current system? Have you outlived the vendor and their support for that product? If so, you're not alone. And you're probably glad that you still have access to your information and it didn't go missing when they left the scene.

If you're not hosting your own system now, or if you're considering a move to a vendor-hosted solution, ask yourself and the vendor(s):

- Do you own your own data? Assuming that you do, do you exercise operational control over it or does some outside organization?
- What happens if the vendor stops supporting the system, or if your relationship with that vendor goes sour, what can you do?
    - Can you export your data and import it into a new system yourself?
    - Is all of the data available?
- What are your options for integration with other systems?

Having operational control over your systems is your last fallback if your vendor goes missing or the relationship goes south. It also broadly widens your options if/when you want to integrate your volunteer system with other applications: most cloud-based solutions have limited general import/export capabilities.

Best practices (Ownership and Control)

Understand where your data is. Is it in your hands or a vendor's?

Have a fallback plan. Source code escrow, triggered if the vendor goes into Chapter 7 bankruptcy or stops supporting the system; protects you by retaining your options and value in your investment.

Get backups/exports. Even if you're not concerned today, having backups of your data – even limited ones – means that you can access your information if something goes wrong, even if that "something" is someone on your end accidentally damaging it.

About the author

Mr. Cornett has worked with and for large non-profit organizations since 1989 building information systems for managing athletes, coaches, volunteers and other participants. He's the lead developer of VSys One and other applications for Bespoke Software, Inc.; he's also the company president. In addition to VSys One and its companion tools, he designed and built GMS, used worldwide to manage Special Olympics competitions of all sizes.