

CONFIDENTIAL

Security Configuration Report for Cluster Deployment
Company

SAXGOR Security

Table of Contents

Table of Contents	1
Executive summary	2
Technical Overview	3-4
Network Architecture Overview	5
Server Configuration Documentation	6-9
Short Term Security Recommendations	10-13
Long Term Security Recommendations	13-15
Conclusion	16

Executive Summary

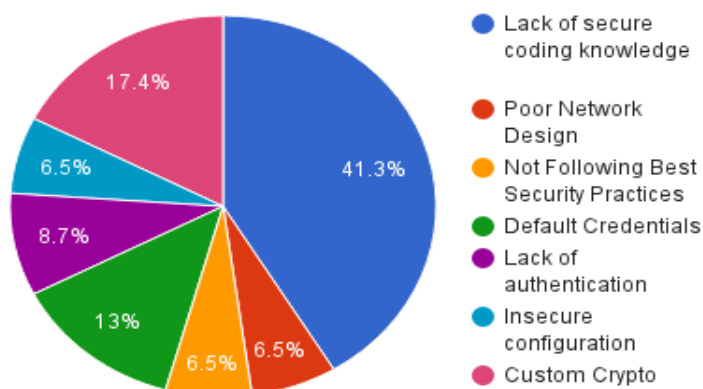
SAXGOR Security was contracted by the CIO of your company, the Cluster Deployment Company (CDC), to provide Information Security related consulting services. The previous company which was hired to setup and secure your systems had done a poor job configuring the systems and intrusions were occurring regularly. We were contracted to identify any possible security vulnerabilities on the network and to redesign the network in order to work to make it more secure.

We must admit that this task was not easy. There were a number of time constraints which we had to put up with and a number of requirements which we had to satisfy which increased the complexity of deploying the design of the network which took additional time which we were unable to spend doing other things to secure your network.

We have done our best to secure your network based on the constraints which you have put on us and have attempted to prioritize the security issues we fixed in order to fix the most serious vulnerabilities first. However, depending on your company's risk tolerance level and other factors it may be necessary to put more robust security controls in place. More information on this is included in the technical portion of our report in order to assist individuals responsible for managing your network.

While auditing your network we identified a number of inherent issues which we believe need to be addressed. The main one is that we have identified a large number of issues in your network which reflect a lack of understanding of security by your developers. Because of this we feel that additional training is needed in order to ensure the continued security of your network. If you examine the chart below you will see that the majority of the vulnerabilities identified or fixed on your network were the result of a lack of knowledge of secure coding and security best practices.

Overall Vulnerability Summary



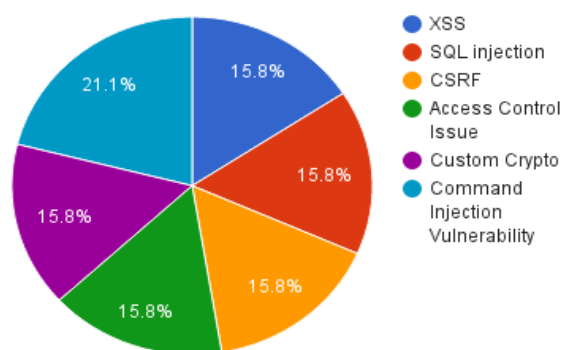
Technical Overview

While re-designing your network we took the approach of building security in from the beginning as opposed to building on something that is known to be broken. We did this because we believed it would be faster to re-build from a known secure configuration. This also stopped us from having to play “privilege escalation vuln whack-a-mole” because we were able to build from what we can assume is a more or less a known secure configuration. We setup a centralized authentication server in order to handle all logins. This server uses OpenLDAP in order to provide directory services. All of our systems are configured to authenticate against this system. It also stores users public and private keys. We have another server running a web service which implements key escrow. This service interacts with our OpenLDAP server in order to allow users to change their keys, generate new ones, and also allows the website to get users keys and generate new ones also.

Our primary focus was on addressing “low hanging fruit” vulnerabilities first and due to the time constraints put on us we did not try to implement more advanced access controls. This is simply done because there is really no reason to bother with implementing more advanced security controls when we are under limited time constraints. We want to provide you with the best services we can in the time-frame you have allotted so we have focused on fixing issues which will give us the greatest return on investment.

The website PHP code was very broken and contained a large number of security issues. We ended up ripping out most of that code and re-writing it in a secure manner. The primary issues were access control related, injection related issues (command injection and SQL injection), along with a large number of cross site scripting and CSRF vulnerabilities. It also implemented a custom session management mechanism with a number of vulnerabilities and used a custom version of “SHA256”. If you examine the chart below you will be able to see what kind of vulnerabilities were discovered.

Web Application Vulnerabilities Summary



When re-designing the network we kept the original systems in place and then just made new VMs to replace the old ones. Afterwards we simply looked at the old systems and either threw out the bad stuff (key escrow and most of the website code) and replaced it. With key escrow we noticed the use of custom encryption and it was written in a relatively obscure programming language. We instead replaced it with standardized protocols and software (OpenLDAP and SSL/TLS) which are used instead. This allows us to avoid the problems associated with trying to build a custom piece of software with custom protocols and custom crypto and instead use something which is proven to work and have been well audited and documented.

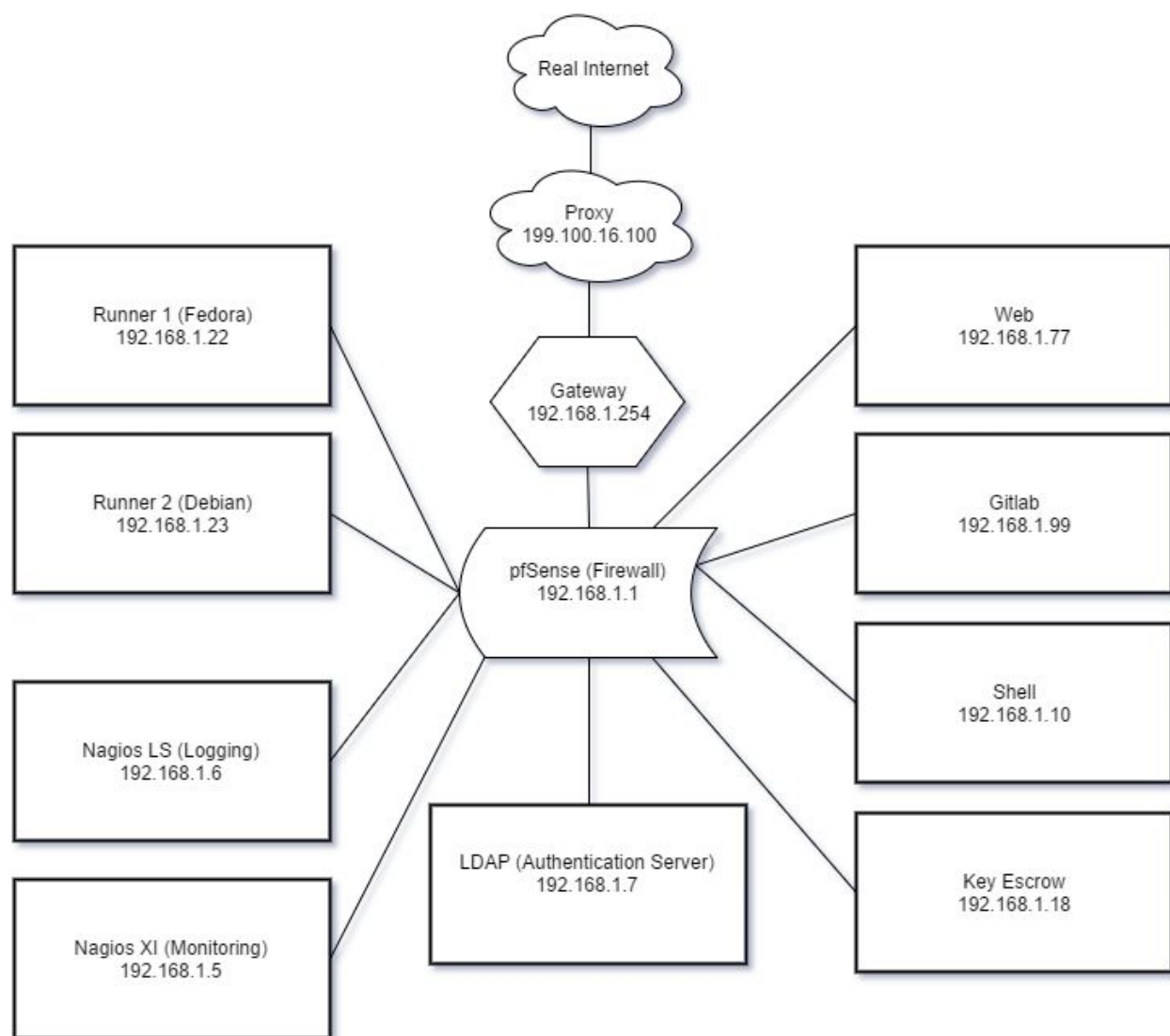
In addition to this we also attempted to deploy extensive logging on your systems, so that if any of your systems are compromised it is much easier to trace back the source of the intrusion and clean up afterwards. We are using Nagios Log Server as our centralized log server. The advantage of using a remote log server is that once a system is compromised the integrity of the log files on that system can no longer be trusted. When logs are sent to a remote log server this raises the bar for an attacker because they must compromise the log server in order to delete any suspicious log file entries. We also deployed another solution called Nagios XI which allows for log file monitoring and for alerts to be set for when certain suspicious events occur on your network.

Below is a table which indicates which servers logging has been configured on and also lists their external and internal IP addresses and servers associated with them.

Server Name	Network IP Address	Internet IP Address	Logging Status
<i>Runner Debian</i>	192.168.1.22	79.5.64.22	Fully logged
<i>Runner Fedora</i>	192.168.1.23	79.5.64.23	Fully logged
<i>Shell</i>	192.168.1.10	79.5.64.10	Fully logged
<i>Key Escrow Debian</i>	192.168.1.18	79.5.64.18	Fully logged
<i>GitLab</i>	192.168.1.99	79.5.64.99	Fully logged
<i>LDAP (Authentication)</i>	192.168.1.8	79.5.64.8	Fully logged
<i>Web</i>	192.168.1.77	79.5.64.77	Fully logged
<i>Nagios LS (Logging)</i>	192.168.1.6	79.5.64.6	Will not log
<i>Nagios XI (Monitoring)</i>	192.168.1.5	79.5.64.5	Will not log
<i>pfSense (Firewall)</i>	192.168.1.1	79.5.64.1	Will not log

Network Architecture Overview

We did not change the network architecture at all from your original design, however we did change the IP address of certain servers. We have created a network diagram below which displays the network layout and the IP addresses which have been assigned to each server.



Server Configuration Documentation

This section covers what was done on each one of your servers. There are a few servers which we added like the Nagios Log Server, Authentication Servers, and Nagios XI Server. Outside of rebuilding systems from a known secure configuration we simply reconfigured things trying to follow best security practices and again just focusing on the low hanging fruit issues.

Authentication Server

The authentication server is a new server which we have deployed which runs OpenLDAP this server is used to store user authentication information/user information and public and private keys for users. This server runs Ubuntu Server and its domain name is “ldap.team6.isucdc.com”

The following things were done to configure the Authentication Server

- Configured OpenLDAP directory/database
- Configured HTTPS
- Runs Ubuntu Server
- IP Address 192.168.1.8 -> 79.5.64.8
- Configured PHPLDAPADMIN control panel
- Configured logging on server

Runner Debian

The Runner Server running Debian Linux primary purpose is to run user programs and display the results of these programs. Because these programs are untrusted it is important that a high degree of security is implemented on these systems.

The following things were done to configure and secure the Runner Server that was running Debian

- Re-imaged the system in order to start from a known secure configuration
- Configured OpenSSH authentication to go through LDAP
- Configured OpenSSH “AuthorizedKeysCommand” in order to run a script which retrieves a user's public keys from the LDAP server
- Configured a static IP address for this system (192.168.1.22 -> 79.5.64.22)

Runner Fedora

The Runner Server running Fedora Linux primary purpose is to run user programs and display the results of these programs. Because these programs are untrusted it is important that a high degree of security is implemented on these systems.

The following things were done to configure and secure the Runner Server that was running Fedora

- Re-imaged the system in order to start from a known secure configuration
- Configured OpenSSH authentication to go through LDAP
- Configured OpenSSH “AuthorizedKeysCommand” in order to run a script which retrieves a user's public keys from the LDAP server
- Configured a static IP address for this system (192.168.1.23 -> 79.5.64.23)

Gitlab

The Gitlab server exists to act similar to github and to allow developers to upload their source code to git and run it on the runners.

The following things were done to configure and secure the Runner Server that was running Ubuntu

- Re-imaged the system in order to start from a known secure configuration
- Configured OpenSSH authentication to go through LDAP
- Configured a static IP for this system (192.168.1.9 -> 79.5.64.9)
- Installed gitlab
- Wrote scripts to use gitlab API from the website
- Configured HTTPS

Web Server

The web server is used to host the www.team6.isucdc.com website.

The following things were done to configure and secure the Web Server

- Re-imaged server and rebuilt from scratch
- Configured LDAP authentication
- Only allowed logins from administrators not customers
- Configured HTTPS
- Installed CRCONSOLE application
- Audited website code and fixed vulnerabilities

On the website the following types of vulnerabilities were fixed in your application

- Access control related vulnerabilities (users could view other users private keys and credit card numbers)
- Insecure Password hashing mechanism
- Custom session management mechanism - weak session token generation and sessions did not expire ever
- SQL Injection Vulnerabilities
- Command Injection Vulnerabilities
- CSRF Vulnerabilities
- Cross Site Scripting Vulnerabilities

These vulnerabilities were fixed by using a combination of stripping bad input from user input and also escaping that input. This is a fairly paranoid approach but we want to avoid command injection vulnerabilities at all costs. We do have some concerns that the data being passed to the crconsole application is going to lead to some command injection issues on the website so we are heavily stripping user input that is being passed to that command to prevent command injection vulnerabilities later in that program's execution. We re-built the session management mechanism in order to use PHP's default session management mechanism which we believe should be sufficiently secure and works better than trying to implement our own. The website was also modified to use LDAP for user login and registration.

Shell Server

We ran into some “minor” issues while configuring the OpenSuse Shell Server. The primary issues we ran into were that there is a lack of documentation as to how to configure things. The stock OpenSuse install does not seem to have a man command at all by default and it has to be installed as an add on post install (which seems a little strange to use). We ended up re-imaging the system and looking at the old shell server in order to figure out how to do things with it.

The following things were done to configure the OpenSuse Shell Server

- Configured static ip address and proxy
- Configured LDAP authentication and key based authentication through LDAP
- Installed development tools on the system
- Installed CRCONSOLE
- Deployed Key Escrow Client Utility

Key Escrow Server

The key escrow server is used by the key escrow client in order to generate new keys and fetch keys. We re-wrote the key escrow software to use industry standard protocols and crypto instead of trying to use a custom solution which is much more risky and likely to cause problems.

The following things were done to configure the Key Escrow Server

- Configure static ip address and proxy
- Configure LDAP authentication
- Setup HTTPS and Apache in order to install key escrow web service

Nagios Log Server

This system was very quick to deploy because we were provided with a ready-made image which we could import into vcenter. We did not do much with this, but we did configure all servers to log to this log server.

Nagios XI Server

This system was very easy to install also as we also had a pre-made image for this we just imported the virtual machine into vcenter. This is used to monitor log files and set off alerts when suspicious activity occurs.

Short Term Security Recommendations

There are a number of recommendations we have for things you can do in the short-term if you wish to improve the security and efficiency of your network. We have listed these things below.

Standard linux distribution

Your environment required that we use different Linux Distributions on a large number of different servers. This increased the amount of work we had to do which then caused us to have to spend more time configuring our systems and decreased the amount of time we were able to spend securing your systems. By using a standardized distribution, we recommend Ubuntu Server because of its ease of deployment and the plethora of documentation which is available to quickly configure things, this will decrease the amount of work that is required to configure systems on your network and save you time and money. However, since you required us to do this we have done it even though we think that it does increase complexity and makes life harder.

Security Testing

It is great that you are bringing in people to test the security of your network. This is especially good because we have been under such strict time constraints that we have been unable to do much in the way of security testing other than just basic security baseline testing. However, one thing we would like to note is that the penetration testers you have hired to assess your network are limited in their scope of engagement. Limiting penetration testers in the scope of their engagement is bad because it can artificially decrease the attack surface of your network. They are also operating on a limited time frame and have a large number of systems they need to access. This can cause their testing to feel much more like a penetration test and much less like an attack being performed by a dedicated, targeted attacker who is able to methodically and patiently move through your network in an organized and logical fashion.

Redesigning PAM_LDAP Module for more secure automated user registration

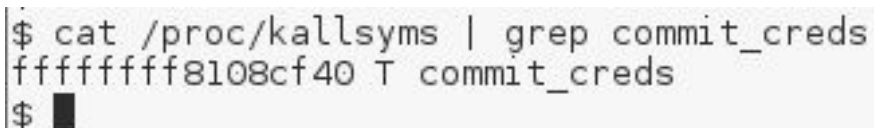
One issue we were not able to address which may affect your users is that we had to disable automatic registration because there are a number of issues with the pam_ldap module we have installed which make registration risky. The first is that you could register a user named “root” and even though they did not have uid=0 set they still had root privileges when they logged in over LDAP. The main issue with this is that the data stored in LDAP is untrusted user-input for the most part and the pam_ldap module is not designed to treat it in this way. We have several ideas on how you could rework authentication in order to implement fully-automatic registration

in a secure way. However, we are not able to do this due to time constraints we chose to make a trade-off between security and usability as the risk of deploying automatic registration in a production environment just seemed too risky at this time.

Deploying GRSecurity on Runners and Shell Server

There is a high degree of probability that in the long term an attacker will be able to get root on one of your runners or the shell server as these systems essentially exist to run untrusted code. Right now your systems should be fully patched and well configured, but in the long term there is a chance these systems may become outdated or misconfigured (especially if staff are not properly trained in security). These two things together can lead to a dangerous outcome for your company. One way to harden your systems is to use GRSecurity both to protect against kernel exploitation which is beneficial because even if it is several months from now there is inevitably going to be another linux kernel privilege escalation vulnerability (like the keyctl system call use after free vulnerability which was disclosed not too long ago) and an attacker can take the proof of concept and rush to write a working exploit (in the case of your systems the proof of concept would of worked unmodified, but on some systems it requires modification to bypass a minimal number of exploit mitigations). GRSecurity significantly raises the bar to kernel exploitation and makes it much more difficult to go from a proof of concept to a working exploit. So even if you do a good job of patching your system if it takes you twelve hours to update your systems that is still plenty of a time who has remote access to rush to exploit the vulnerability. However, if you have GRSecurity in place it makes it much harder to go from a basic proof of concept to a full fledged exploit quickly and even then if the author or GRSecurity finds out which technique you are using he will typically come up with a new mitigation to kill off that technique.

This is an excellent example of a lack of hardening on behalf of the Linux Kernel developers and distribution vendors. The standard linux kernel also by default exports a lot of sensitive information to userland programs which makes kernel exploitation much easier such as the addresses of internal kernel functions in memory.



```
$ cat /proc/kallsyms | grep commit_creds
ffffffff8108cf40 T commit_creds
$
```

This is useful to an attacker when they are running their Ring 0 shellcode to elevate their privileges and modify their processes credential structure in order to give themselves root privileges.

A large number of systems now have another mitigation called SMEP enabled which is designed to stop attackers from jumping directly from executing userland memory in the kernel, but it is trivially disabled with a simple ROP chain and can therefore be bypassed quickly fairly quickly (an attacker just has to modify a bit in the CR4 register in order to disable the SMEP mitigation). However, most mitigations in stock Linux Kernels from most distributions is insufficient to pose any sort of challenge to a halfway competent attacker.

GRSecurity also allows for strong access controls similar to that provided by SELinux or Apparmor, but arguably easier to use and maintain in a practical manner. It is also much stronger because apparmor and SELinux do not make any attempt to harden the kernel from exploitation and if an attacker is able to execute privileged ring 0 code he or she can disable any kernel level mitigations. This is not just something that you have to worry about with nation state backed attackers (and if nation states are in your threat model you are overthinking things for an organization such as yours) if your systems have been operational for a long period of time there is a chance that some of them may be missing critical patches if you do not have an organized team to manage your systems or an attacker can rush to exploit a brand new vulnerability before you can manage to find the time to patch it (even if it is a very limited time frame) and bypass said mitigations. The strong access controls provided by GRSecurity RBAC are beneficial because they allow you to go so far as to restrict the capabilities of the root user also so if an attacker is able to gain root privileges through a misconfiguration that may not lead to a complete compromise of your systems if configured with strong access controls and they are unable to execute privileged code in the context of the kernel.

Of course this does not address the underlying issue of password reuse and lack of strong access controls which are prevalent in your network it is one step of many which would allow you to better secure your systems in the long run and GRSecurity is arguably essential in environments like the ones the runners and shell server are deployed in where attackers can run their own unprivileged code on the system whenever they want and it is a requirement which is necessary in the design of the system.

Storing user private keys is not necessary in your current situation

One of our other comments is that you are storing users private keys which creates a large number of liabilities and negates many of the benefits of using key based authentication. You should consider re-designing your network so that only user's public keys are stored and private keys are only known by the user. There is really no reason not to be doing this and key escrow is almost completely unnecessary. If your service was encrypting files and you wanted to store private keys for law enforcement or nation states that is more understandable (even if some of us may have ethical objections to this), but these keys are just being used to authenticate the

customer not to encrypt information which you may want to be able to later decrypt so it does not really make much sense at all.

We do understand that you would like to be able to store keys in order to comply with law enforcement requests. But, again, these keys are used for authentication and not for encrypting things so storing them is just a liability.

Long Term Security Recommendations

There are a number of recommendations we have for improving the security of your network over the long term. Our primary concern is that there are many underlying issues in your network which needs to be addressed if you want to maintain any degree of security. Security is not stagnant and even if we have done a good job of securing your systems right now that is not any indication of what your systems will look like in 6 months of a year from now. We have listed these recommendations below.

Staff Training - Addressing underlying issues

Another comment we must make is that while we are fixing issues on your network we cannot help to address the underlying issues. Your web developers are not at all trained in secure coding practices for programming. We recommend that your developers attend training courses in order to become familiar with secure programming practices. This is important because while your network may be secure now in the future it will not be because new vulnerabilities will be introduced into your network because the programmers do not know how to write secure code. This is something we know from first hand experience where a vulnerability has been reported to a vendor or developer and then several months later they make the same mistakes again. This makes it impossible to actually secure the software because it is inevitable that developers will keep making the same mistakes if they do not actually learn from them. We cannot comment on the skills of your network administrators since we re-designed the network from scratch we did not audit those servers.

As our goal is to actually make your organization more secure in the long term we feel that this is something that is important to point out. New vulnerabilities are going to be introduced into your systems if staff are not properly trained.

Implementing more robust authentication mechanisms (Kerberos, 2FA, Key Based Auth)

One issue with the design of your network which leads to a large number of single points of failure is the fact that we are using password based authentication and are re-using passwords. This leads to multiple single points of failure because of the fact that if one server is compromised credentials can be stolen and re-used other places. One might think that the use of password hashing on servers is sufficient in order to prevent this especially if passwords are sufficiently complex and the hashing algorithms used make it difficult for attackers to “decrypt” hashed passwords. But, credentials can always be stolen when they are entered into the system before they are hashed. One example of this is an attacker can write a program which injects into the OpenSSH process on the system and steals credentials from

```
root@localhost's password:
Last login: Sat Mar 19 16:15:10 2016 from localhost

Hello,
Backdoor has been activated..
enjoy your root access and credentials :)
Username Mushnik Password
[0.7]INCORRECT Validity 0
Username dlimanow Password cdc Validity 1
Username coryscheer Password cdc Validity 1
Username lcosman Password cdc Validity 1
Username acrosser Password Uc#7n9fIGu")DJh$?S Validity 1
Username chein Password cdc Validity 1
Username bbartels Password cdc Validity 1
Username pchihak Password cdc Validity 1
root@webserver:~#
```

memory after they are decrypted after being sent over the network, but before they are hashed and used to authenticate the user on the system. An example of this is shown above.

OpenSSH is just one example of where passwords can be stolen other examples include: sudo, su, backdooring web forms hosted on the server, etc. Because of this an attacker who is able to get root on one server is able to easily spread laterally across our network and compromise every other system on the network. There are a number of ways you can combat this by using key based authentication, using password managers in order to prevent password reuse, using kerberos so that servers are not being sent credentials only authorization tickets,, and using two factor authentication.

In order to combat credential theft we could have elected to use Kerberos, key based authentication, or multi-factor authentication (or some combination of these things). However, we elected to not do this as we did not want to break compatibility with the ISCORE service

checker and were also worried about time constraints and wanted to focus on fixing other “low hanging fruit” security issues within our network. This is a problem which needs to be addressed in the long term.

Implementing Stronger Access Controls

Another way to combat account compromise is to restrict what certain users have access to and for how long. Depending on your level of risk tolerance and implications of a compromise you can restrict down what even administrators are able to do to a very limited level and scope and only temporarily expand that scope as they need access to things. This way even if a single administrative account is compromised the implications of such a compromise are somewhat limited.

Conclusion

There are a number of other factors which we could go over for long term security and there are a number we are completely overlooking due to the fact that we are very limited in the time that we have to deploy and secure your systems.

There is a lot that needs to be done to make sure that our network design is stable and that individual servers getting completely compromised does not lead to the compromise of our entire network. This could be an issue and requires stronger access controls to be put into place which we are currently unable to do. This is a company that many companies face and is appropriately deemed “M&M Security” meaning that while your network is secure from attackers on the perimeter, but once they are able to actually get in and compromise a few servers certain design issues can make it easy for them to spread laterally and compromise the entire network. This is not something that is affected by smaller companies like you're from reading about how organizations like NASA were compromised (assuming the source is credible it seems within the bounds of reason but NASA has not admitted to being compromised) if true it is apparent that this is an issue that impacts even larger organizations.

Because of this we believe that in the long term more robust security controls should be put in place. This of course all depends on who and what you are trying to defend against and what the ramifications of a compromise are for your company. One could imagine that with a company like yours where developers are trusting you with their source code (and its confidentiality and integrity), financial information (credit card numbers) and the computations which they are doing on your servers may also be confidential. It is likely that security is an important factor in your business especially since it is a largely online organization.

These are of course a number of other things which we are not even able to mention (or we would spend all day writing documentation and not securing your systems) .So if you are at all interested in trying to really improve security beyond the currently level you have implemented please consider extending your contract with us. We have very much enjoyed working with the staff at your organization and on behalf of our entire team here at SAXGOR Security we would like to thank you for choosing us to assist you in securing your systems.