

UWTSD

Ethical Hacking Report

Research on a recent cyber security attack

Change healthcare

Introduction

In today's world, technology is essential in healthcare. Companies like Change Healthcare manage millions of patient records, medical payments, and communication between hospitals, pharmacies, and insurance providers. This makes them very important but also very vulnerable to attacks. In February 2024, Change Healthcare was the victim of one of the biggest cyberattacks ever seen in the healthcare sector. A ransomware group hacked into their systems, blocked access to important services, and stole massive amounts of private data.

This attack had serious consequences. Hospitals couldn't process payments or approve treatments, pharmacies couldn't verify prescriptions, and many patients had to wait longer for care. It also raised big questions about data protection and cybersecurity in healthcare. If a large company like Change Healthcare can be attacked, then the whole system might be at risk.

In this report, I will explain what is the attack used, how the attack happened, who was responsible, and what the impact was. I will also describe how the company and the U.S. government responded, and what can be done to avoid similar attacks in the future.

The attack

What is a ransomware?

A ransomware attack is a type of cyberattack where hackers use a special kind of malicious software (called "ransomware") to lock or encrypt files on a victim's computer or network. Once the files are locked, the victim cannot access them unless they pay a ransom usually in cryptocurrency. A famous case of ransomware that happened and is known by many people is WannaCry, one of the biggest ransomwares that ever happened in the world.

Ransomware works by secretly entering a system (often through a weak password, a phishing email, or a software flaw), then spreading across the network. It can block hospitals, companies, or even governments from using their systems. Sometimes, the attackers also steal sensitive information and threaten to release it if the ransom is not paid.

In the case of Change Healthcare, the hackers not only encrypted their systems but also stole a large amount of private medical and financial data, which made the situation even more dangerous.

Why change healthcare?

As we saw earlier, Change Healthcare plays a key role in the U.S. healthcare system. It is involved in over 15 billion healthcare transactions every year, including things like insurance claims, pharmacy billing, and patient eligibility checks. Because of this, if Change Healthcare stops working, the effects are felt almost everywhere hospitals, pharmacies, clinics, and insurance companies all get disrupted.

This made it a perfect target for a ransomware group. The attackers knew that shutting down Change Healthcare would cause nationwide chaos and put pressure on the company to pay the ransom quickly. In cybersecurity, this is called a “high-value target” a company that handles critical operations and cannot afford to be offline.

Also, the attackers found a technical weakness a remote server without multi-factor authentication. That gave them a way into the system, and once inside, they could move around easily and prepare the attack.

Many cases of ransomware happened to hospital, a hospital or a vital service cannot stop to work, it could be dangerous to people, and hackers know that types of medical structure could not wait to find how to solve the problem, and will pay even if they aren't sure their data can be recovered.

What happened to Change Healthcare

The ransomware attack on Change Healthcare began at the end of January 2024, although it wasn't discovered until February 21, 2024. The hackers used a remote access method to enter the system. They found a server without multi-factor authentication (MFA), which made it easier for them to log in and move inside the company's network.

The cybercriminals belonged to a ransomware group called ALPHV, also known as BlackCat. This group is known for targeting large organizations and demanding huge ransoms. Once they entered the system, they moved quickly to:

- Access sensitive systems,
- Steal around 6 terabytes of data,
- Encrypt important files and stop the company's services from working.

On February 21, 2024, Change Healthcare noticed unusual activity and began shutting down parts of its network to limit the damage. But by then, the ransomware was already deployed and the company's payment processing, pharmacy verification, and other services were frozen.

In early March 2024, UnitedHealth Group (the parent company of Change Healthcare) decided to pay the hackers \$22 million in Bitcoin to recover the systems. However, another hacker group called RansomHub later claimed it still had a copy of the stolen data and tried to blackmail the company again.

Even after the payment, it took weeks before Change Healthcare could fully restore its systems. During that time, hospitals and pharmacies across the United States experienced delays and serious problems delivering care to patients.

The aftermath

One of the most serious consequences of the ransomware attack was the financial impact, not only for Change Healthcare but for the entire U.S. healthcare system. After the attack, UnitedHealth Group, which owns Change Healthcare, said it expected total losses of more than 2.5 billion dollars. This included the ransom itself, the costs of restoring systems, legal actions, and compensation for clients and patients affected by the service disruptions.

The real problem, however, was the effect on hospitals, clinics, and pharmacies across the country. Many of these healthcare providers depend on Change Healthcare to process insurance claims and receive payments. When the system stopped working, these organizations couldn't get paid. Some clinics couldn't even afford to pay their own employees and had to use their savings just to stay open. A few had to reduce their opening hours or stop taking new patients altogether.

The situation became so serious that the U.S. government even offered temporary financial assistance to help healthcare providers survive. Insurance companies also tried to find workarounds, but nothing could fully replace the role of Change Healthcare.

A second thing that this case has shown is the importance of having more than one company in the same part of the market. Change Healthcare is so big in the U.S. that this problem raised serious questions about giving one company control over the data of almost all U.S. citizens. If a data leak happens in a company that has competitors, it will only affect a part of the population. But in this case, it's almost everyone in the U.S.

Who is to blame?

Even though Change Healthcare plays a huge role in the U.S. healthcare system and some people blame the company for being "too big," it's important to not forget where the real danger comes from. The real problem is the attackers themselves. Cybercriminals today are smart, experienced, and organized. They know exactly where to look, what to target, and how to put pressure on companies to pay. They don't just guess they study their victims, find weaknesses, and attack when they know it will hurt the most. In this case, they found a technical vulnerability and took advantage of it at the perfect moment.

The truth is, no system is ever completely safe. Even with good security, there will always be some risk. That's just how the digital world works. Technology changes fast, and attackers always find new ways to break in. So while it's true that Change Healthcare is a very big company with a lot of control over healthcare data, the fact that it was targeted doesn't automatically mean it was careless. Being big makes it more visible but no company, big or small, is completely protected.

This is why the fight against ransomware can't just be about improving security in one company. It has to include better laws, more international cooperation, and stronger actions against cybercriminal groups. We should continue asking questions about how data is managed and shared, but we also need to stay focused on the real source of the damage: the people who launch these attacks in the first place.

Conclusion

The ransomware attack on Change Healthcare was not just a technical incident it was a national crisis. It showed how one cyberattack can affect millions of people when one company has such a central role in the healthcare system. Hospitals, pharmacies, and patients all suffered because of the loss of access to basic digital services.

This case also reminded us that no system is fully protected, and that data will always be a target. Even a large and powerful company like Change Healthcare, with big resources, was not safe. But while it's important to improve cybersecurity and rethink the structure of the healthcare industry, we must not forget who the real enemy is: the organized and professional cybercriminal groups who build and spread ransomware.

In the future, better collaboration between companies, governments, and international agencies will be necessary to protect critical services like healthcare. More transparency, better preparation, and stronger action against attackers are the keys to reducing the impact of future attacks.

Sources

- <https://csrc.nist.gov/pubs/ir/8374/ipd>
- <https://thehackernews.com/2023/06/improved-blackcat-ransomware-strikes.html>
- <https://www.zdnet.fr/actualites/alphvblackcat-ils-raflent-une-rancon-de-22-millions-de-dollars-et-ferment-subitement-boutique-39964786.htm>
- <https://www.aha.org/change-healthcare-cyberattack-underscores-urgent-need-strengthen-cyber-preparedness-individual-health-care-organizations-and>

Penetration testing for virtualize simulated company website

Website presentation and idea of research

In this penetration testing project, the goal is to evaluate the security of a company website that would typically be hosted on a virtualized Linux server. The website would include common features such as a public home page, a contact form, a login page for clients, a password reset function, and an admin interface. The technologies used in this case are Apache for the web server, PHP for the backend logic, and MySQL for the database.



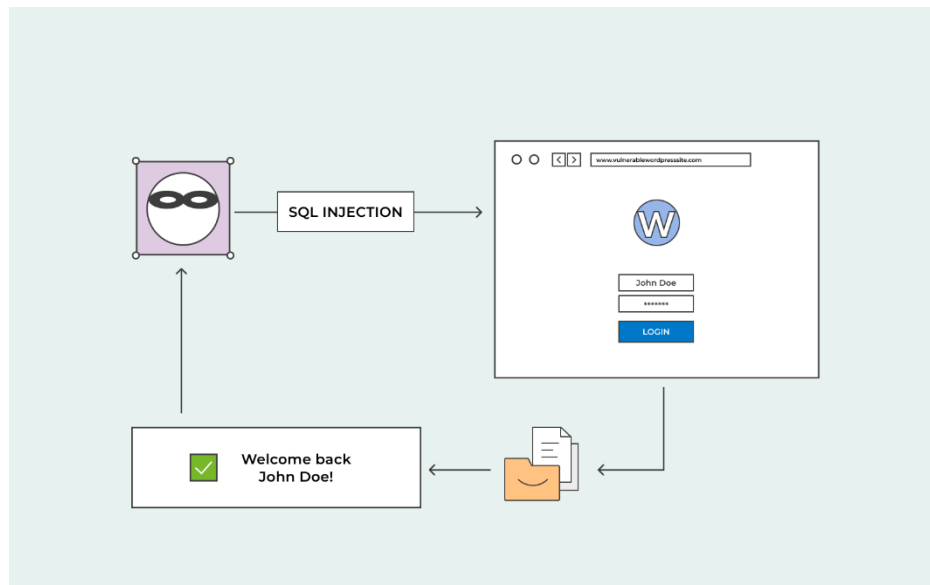
To carry out the testing, several tools are used. Nmap is used first to scan for open ports and identify the services running on the server. Then, Nikto is launched to detect outdated components, default files, and misconfigurations on the web server. Finally, Burp Suite is used for intercepting HTTP requests, modifying form submissions, and checking for vulnerabilities such as injection flaws or weak session handling. In case any user input appears to interact with a database, SQLmap is used to test for SQL injection vulnerabilities.

Before using the tools, a manual exploration of the website is done to identify points of interest, and actions that can be made as a human, for example SQL injection can be done using login forms, any point of user interaction can be a weakness on a website if things are not done properly. This includes browsing each page, observing how forms work, checking the structure of the URLs, and looking at cookies and HTTP headers. The aim of this phase is to get an idea of the surface exposed to users, and to define what parts of the application should be tested in more detail.

After this information gathering phase, active testing will be performed. This will include automated scans as well as manual testing. Each action taken will be focused on identifying weak points in the site's security that could allow us to have non authorized access, or data extraction, or functionality abuse.

Penetration testing

SQL injection



After collecting enough information about the website and identifying potential entry points, I began active testing using the tools previously mentioned. One of the first tests focused on the login page. By intercepting the HTTP request using Burp Suite and modifying the username field, I tested how the server handled input that included SQL syntax. By inserting a classic payload like ' OR '1'='1 into the username field, the server responded as if the login was valid, even though the password was incorrect. This confirmed the presence of an SQL injection vulnerability.

This kind of flaw suggests that the server takes user input directly and places it into an SQL query without sanitizing it. The server likely builds a query such as: `SELECT * FROM users WHERE username = '$username' AND password = '$password';` With the payload, the condition becomes always true, which allows bypassing authentication. In a real environment, this could allow an attacker to access private accounts without knowing the correct credentials.

Nikto (PHP version verification)



During the testing phase, I used Nikto to scan the web server and identified that the application was running on PHP version 5.6.40. This version is outdated and no longer receives official security updates. Even though the server was using Apache and not Nginx, the presence of such an old version of PHP raised several concerns. I was able to confirm that the server responded differently when interacting with certain inputs, especially when using file upload or script execution features, which suggested that the backend might be vulnerable to unsafe function calls or poor input handling.

By analyzing the way the application handled parameters and user submitted data, I found that it did not properly validate certain inputs. For example, when submitting long strings or special characters into form fields, the server occasionally returned error messages or unusual behavior. This indicated that the version of PHP installed could be exposing internal logic or error handling mechanisms that should normally be hidden.

The presence of this outdated PHP version also allowed me to explore potential weaknesses related to how the server handles memory and error reporting. Even though I did not achieve direct code execution, the version in use made it possible to collect detailed information about the application's structure, response patterns, and internal components. This information could be useful in further testing or for chaining with other vulnerabilities.

In combination with the SQL injection previously found in the login form, I was able to access protected parts of the application without valid credentials. Once inside, I tested how the application managed user permissions, session handling, and data visibility. In some areas, sensitive information such as user names, emails, and internal notes were accessible. If the server-side code had additional weaknesses tied to the outdated PHP version, it's likely that further escalation would be possible.

These findings confirm that the combination of weak input handling and unmaintained software creates a real risk. By using only the tools and data available during this test, I was able to gather useful information, bypass login protection, and access internal areas of the site all without having special access or credentials.

How to secure this website

To secure and patch the website, several actions should be taken. The most urgent fix concerns the SQL injection in the login form. This issue exists because user inputs are directly included in the SQL query without any filtering. The correct way to prevent this is by using parameterized queries or prepared statements. These methods separate SQL commands from user input, making it impossible to inject arbitrary code. For example, using PHP with PDO or MySQLi in prepared mode would completely block the injection I used to bypass the login.

In addition to changing the way queries are built, it is important to disable detailed error messages in production. During my test, some invalid inputs triggered server responses that revealed parts of the SQL query structure. These messages should be logged internally but never shown to users, as they can give away too much information about how the application works.

The second major issue is related to the outdated PHP version. Running unsupported versions of software like PHP introduces multiple risks, especially when those versions are known to contain exploitable bugs. The server should be upgraded to a current, supported version of PHP (such as 8.1 or above at the time of writing). This update process must be done carefully to make sure the website code remains compatible, but it is essential for long-term security.

Other good practices include limiting file upload permissions, restricting access to administration pages, and applying the principle of least privilege for user accounts. During the test, some admin related features were accessible too easily after bypassing login. These areas should be protected with stronger authentication, and their URLs should not be easily guessable or public.

Regular updates and vulnerability scans are also part of a proper defense strategy. Tools like Nikto, Nmap, and Burp Suite can be used internally to check the system before attackers do. This kind of internal auditing should be part of a routine maintenance process, not something done only once.

Conclusion

I choose to focus on these two issues SQL injection and outdated PHP because I've personally faced them in small websites I developed for myself. At the time, everything seemed to work fine, but I later realized, with the help of people more experienced than me, that serious security problems were hidden behind the surface. They showed me where the mistakes were and how easily they could be exploited if the site had been public.

This made me understand that even simple projects need to be properly protected and maintained. A website is never "finished" it needs to be updated regularly, monitored, and reviewed for weaknesses. Anyone, even without bad intentions, can end up in a vulnerable situation if they don't pay attention to these things. That's why it's important to be informed, to learn from others, and to take security seriously from the beginning.

Even if someone is not targeting a specific website, many sites can still be hacked using general automated tools like the ones I used during this test. There are bots constantly scanning the internet, visiting as many sites as possible every day, looking for common vulnerabilities and weak configurations. If one of them finds a known issue, it can be exploited in seconds without any human decision involved. That's why taking care of security is not optional. It's part of the job when putting any website online.

Sources

- https://owasp.org/www-community/attacks/SQL_Injection
- <https://nvd.nist.gov/vuln/detail/cve-2019-11043> (example of CVE for using an old php version)