A world without FCAPS

Dominic Rathbone

May 13, 2016

Chapter 1

Case Study - E-Corp Network Outage

Introduction

This case study aims to look at a fictional example of a malicious attack on a company called E-Corp that was able to cause a network outage.

Background

E-Corp are a digital payment processing start-up founded in 2009 that supply an alternative means of payment to traditional electronic and paper methods(such as debit card or cheque) to over 250,000 registered users. Their products takes the form of a consumer web application and an external payment gateway API that e-commerce companies can develop against and include on their site. The web application is the user-facing platform for the gateway and allows consumers to sign up and register their debit and credit cards within a digital wallet. To receive money, E-Corp users register their bank account which they can forward money to. Once a consumer has registered a card, they can either send money directly to someone through the web application or login to their account through the gateway on external sites and buy services or products. E-Corp charge a 3.5% fees to sellers using the service but this gets lower as transactions increase.

2012 Network Outage

In 2011, E-Corp started development on an android application that enabled buyers and sellers to sign in to their account away from a desktop computer. Similar to the web application, the mobile application allowed users to see a list of transactions as well as registered debit cards, credit cards and bank accounts whilst also being able to add and remove from these accounts and send money to or request money from another user. In order to support th mobile application, the development team built a new API against the web application using XML as it's data transfer format. As the release deadline for the mobile application got closer, management started to put pressure on the development team in order to get it released on time. Due to this, shortcuts were taken and although it was released on time, it was done so without being fully tested by the developers and the quality analysis team.

A few weeks after the application was released in the first quarter of 2012, the E-Corp web application crashed and the mobile application was getting server errors from the API. Eventually, it was discovered that this was maliciously caused by an anonymous hacker who was able to run a Distributed Denial of Service (DDoS) attack against the E-Corp network through the new API built for the mobile application. This is an attack where a server is flooded with requests until it is either not able to serve legitimate users anymore or crash. It was revealed that this attack was made possible as the endpoints into the new API used a vunerable XML parser enabling the attacker to generate requests containing an ”XML Bomb”.

An XML bomb is an attack utilising a HTTP request containing a small XML document that expands dramatically when parsed by a vulnerable XML parser. This XML document contains an entity ”xmlbombs” at it’s root that refers to another entity ”xmlbomb9” however this also refers to 10 instances of ”xmlbomb8” and so forth until it reaches ”xmlbomb0”. The XML parser reads the initial root entity, sees that it refers to another and expands on it, doing so continuously until it reaches the end. By this time, the XML document has expanded to take up a huge amount of the server’s temporary memory, slowing it down to the point where it can’t process other user’s requests or even crashing the server.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xmlbombz [
<!ENTITY xmlbomb "xmlbomb">
<!ENTITY xmlbomb1 "&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&xmlbomb;&
      xmlbomb;">
<!ENTITY xmlbomb2 "&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&
      xmlbomb1;&xmlbomb1;">
<!ENTITY xmlbomb3 "&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&
      xmlbomb2;&xmlbomb2;">
<!ENTITY xmlbomb4 "&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&
      xmlbomb3;&xmlbomb3;">
<!ENTITY xmlbomb5 "&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&
      xmlbomb4;&xmlbomb4;">
<!ENTITY xmlbomb6 "&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&
      xmlbomb5;&xmlbomb5;">
<!ENTITY xmlbomb7 "&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&
      xmlbomb6;&xmlbomb6;">
<!ENTITY xmlbomb8 "&xmlbomb7;&xmlbomb7;&xmlbomb7;&xmlbomb7;&xmlbomb7;&xmlbomb7;&xmlbomb7;&xmlbomb7;&
      xmlbomb7;&xmlbomb7;">
<!ENTITY xmlbomb9 "&xmlbomb8;&xmlbomb8;&xmlbomb8;&xmlbomb8;&xmlbomb8;&xmlbomb8;&xmlbomb8;&xmlbomb8;&
      xmlbomb8;&xmlbomb8;">
]>
<xmlbombz>&xmlbomb9;</xmlbombz>
```

As the account creation process for the E-Corp service didn’t use any anti-spam protection, the hacker was able to automate the registration of fake E-Corp accounts and then use these in combination with a bot-net they had previously constructed to send hundreds of requests containing XML bombs through different machines from all over the world and as the requests were authenticated, the API did not stop them. By using a bot-net, the identity and location of the hacker was undetectable as none of the requests were being sent from their machine. As well as this, it took longer to stop the attack as it was harder to isolate the malicious client as the IP address of the requests constantly changed.

As E-Corp scaled their service vertically, their entire range of services including the payment gateway and web application were ran from a singular server and when the new API was released, it was also put on this server. This meant every single malicious request sent to the vulnerable API endpoints also went to the server the web application and payment gateway were hosted on. Initially, this resulted in requests from legitimate users using the mobile application, web application and payment gateway being slowed down but the attack quickly managed to completely crash the server. This meant not only the API crashed but the outage also spread to the other services E-Corp supplied and thus to every single consumer and every e-commerce platform that used their payment gateway. This limited the payment methods these platforms offered to users and in some cases, completely hindered the platforms that were dependent solely on E-Corp as a payment method.

Due to the lack of monitoring put in place, the system administration team were severely impacted in how fast they were able to react to the problem as they were only notified to the outage when user’s started to send in complaints. Once they were alerted to the problem, they struggled to isolate the attack to the specific group of IP addresses and user accounts that were sending these requests as the logging for all the services was configured to send data to one set of monolithic log files as opposed to being configured to use their own log files. In turn, this increased how long it took for the particular vulnerability to be found. Once the system administration team had scraped a list of malicious IP addresses and user accounts from the logs, they were able to stop the attack by banning them from the network. They then had to run a manual roll-back of the API to an older, fault-free version and create a database roll-back script that removed the new columns added when the API was released as well as the user accounts used in the attack. This eventually allowed for E-Corp to provide limited services to their web users through the means of the web application but also meant that new mobile users were still unable to use them. Once the fault was detected and isolated, every effected endpoint in the API had to be re-factored by the development team in order to use a different XML parser and implement data validation on incoming requests. Once the development team had implemented these changes, the quality analysis team had to run extensive API tests against the re-factored endpoints to ensure they were working as expected. The system administration team then had to shut down the server for maintenance to re-release the fixed API version. As the mobile application used the API, it also had to be re-factored to reflect the changes made to the API and as this was where the vulnerability was discovered, it also had to be thoroughly retested before it could be re-released to the public. Overall, the attack had a significant effect on E-Corp's software release cycle as it interrupted the development process of the company, delaying other project's deadlines by several weeks.

The pressure from E-Corp's higher management to release the product on time caused the application to be released untested which meant that they were not aware of the vulnerability happening before the hacker was able to exploit it. Due to this, the event was unexpected by E-Corp and they had to temporarily disable their services for a 3 day long period until the system administration team were able to isolate the fault and roll-back the API, losing the company over 142,500 in user transactions per day and 5,000 in revenue per day during this period. Further to this, they also had to remove the mobile application from the android store whilst it was being modified and tested lasting for a further 2 weeks, dissatisfying customers. In reaction to this event, the company also had to release a public apology and explanation to their consumers and the businesses using their payment gateway. E-Corp also had to offer reparations to the platforms that lost money due to the outage in the form of free transactions in order to keep their business. In the end, their reputation was significantly harmed as reflected by an increase in user complaints by 25% over the next week from when the outage occurred. This was also shown by a 15% decrease in the account registration rate over the next month.

Chapter 2

Reflective Commentary

2.1 Introduction

The handling of the network outage by E-Corp will be evaluated through the context of the International Organization for Standardization's (ISO) FCAPS model. This model supplies a framework that separates the management of a network into 5 particular sections.

- Fault Management: How well a major faults in a network are detected, isolated and fixed. This includes the recovery from and prevention of future bugs.
- Configuration Management: How a network controls configuration of the network and manages changes such as firmware updates or hardware upgrades.
- Accounting Management: How a network gathers usage statistics on a network such as resource utilization across peers. Traditionally, this is used to appropriately bill the peers of a network according to how they use it.
- Performance Management: How well the performance of the network is maintained and scaled.
- Security Management: How access to a network's resources is controlled and restricted.

Fault Management

One of the problems that E-Corp had was the lack of fault detection resulting in them not being alerted to the problem until the user's encountered it. This issue could be fixed by using a network infrastructure monitoring tool such as Nagios. Nagios can be set up to send out email, instant message or even SMS (in combination with external tools such as an SMS gateway) alerts to groups of users such as system administrators or tech support. These alerts are sent out when specific parameters go out of bound and can be separated into levels such as "Critical", "Warning" or "Recovered". For example, if the amount of traffic to a web server increases from the average volume, a "warning" alert could be sent to the system administration team but if a server crashed, a "Critical" alert could be sent to management, system administration and technical support. On top of this, the alert could have also supplied the system administration team with a report detailing the problem that resulted in the alert. In this case, a fault detection system would have ensured that E-Corp knew about the attack as soon as possible, allowing them to isolate the problem faster as well as saving them from significant reputational damage of their user base finding out before them.

Performance Management

The second largest issue that E-Corp faced was that they did not have a network architecture built for redundancy or any measures put in place to prevent service outages in periods of high request volume. An example of this was all of their services being ran from one server, providing a single point of failure. Due to this, their network was easily overwhelmed when the attack started and they were not able to recover from it until the IP addresses sending the requests were banned.

To prevent this, the architecture of the network could be remodelled in order to scale horizontally. Whereas vertical scaling is the ability to grow with the usage of a service by increasing the specification of the hardware it is running on, horizontal scaling is the ability to grow via the increase in amount of servers providing the service. One of the advantages of this is that the network is more resilient to attacks as there is no longer a single point of failure. It also has a higher ceiling to how large it can scale and it costs less on average compared to vertical scaling which eventually would require more and more expensive components to maintain. Scaling horizontally also means that services can scale dynamically as new server instances can be spun up without having to take down the original service. This could be implemented by using a 3rd party service such as Amazon Web Services (AWS) and their Auto-Scaling feature which allows for the automatic creation of new instances in response to increased service usage. These servers are then able to be pre-configured and provisioned with a specific stack of software by a script created with AWS CloudFormation. By moving to a third party cloud server environment such as AWS, E-corp would also be able to unload some of the burden of network management onto a service provider that most likely has a better knowledge of handling issues such as this.

Further to this, to prevent an attack from propagating from one service to another like it did in this case, the service could be modelled in a micro-services architecture in which different smaller services are developed and deployed separately to form one larger distributed service. By separating the concerns of each server, they are less likely to feel the effects of an attack such as this on different parts of the system as it would be isolated to one server. Lastly if an architecture like this was adopted, to diminish the effect of this attack, the volume of requests that were sent to the API could be routed through a load balancer. This is a server that sits in front of the application servers and distributes requests between each server instance to stop one single server from becoming overloaded. However, this does suffer the risk of crashing itself.

By changing the network's architecture in these ways, it would have been significantly more prepared for such a severe increase in traffic and would have been able to provide legitimate users with a better, more reliable service even whilst the attack occurring.

Configuration Management

In order to respond rapidly to faults, a network has to have well managed configuration to allow for services to be downgraded or upgraded with ease. In this case, the lack of pre-prepared scripts to roll the API and database back to an older version meant that it took longer for the service to recover once the fault had been detected and isolated. To stop this from happening, these scripts should always exist before a service is released in the case that something goes wrong. On top of this, the database roll-backs could be automated using a tool such as Flyway or Liquibase which add version control to a database system, allowing for changes to be added or reverted through a command line interface (or within a script).

If E-Corp was to adapt their network architecture to suit a micro-services approach, they could also adopt a tool such as Apache ZooKeeper in order to aggregate configuration for each individual server through one centralized service. By centralizing configuration files this way, it makes the process of re-configuration a lot simpler and thus faster to do as it means administrators do not have to browse every instance of a server or application to make configuration changes. In effect, this also makes it easier for the network to be managed as it scales as configuration would become less of a concern.

Accounting Management

Another problem E-Corp faced was that every service on their server was logging usage statistics to the same file. This made the management of the server's allocation significantly less organised and resulted in it being harder to isolate which IP addresses were being allocated a certain amount of resources. To simplify accounting management at E-Corps, the logging of these statistics should be separated by service and they could then implement a logging aggregation stack such as ELK (ElasticSearch, Logstash, Kibana). This would allow them to collect the logs from each logging destination together in one service whilst still keeping the destinations themselves separate. Using this stack, the network administrators or tech support could also run log analysis and visualise the log data through Kibana's user interface. By doing this, the services could be easily reviewed in order to audit specific users and see how much of the network's resources they are taking up. In turn, this would make it faster to find users running a malicious attack such as as in the case study and for the vulnerability to be isolated.

Security Management

The main entry point into the E-Corp network for a user is the web application where they can register an E-Corp account. However, as E-Corp lacked any anti-spam protection for this account creation process, they were not able to stop the attacker from automating the mass production of E-Corp accounts to authenticate the malicious requests with. In order to avoid this, the account creation process should incorporate a CAPTCHA process that would require the user to enter a unique code in order to register an account, meaning account creation could not be automated.

However, if the user was to circumvent this and manages to start sending the malicious requests, this measure would not be enough. Another measure would be to use anti-DDoS protection hardware such as provided by CloudFare. This works by placing a network of external servers that your network which it connects to the external internet through. By doing this, CloudFare bares the brunt of a DDoS attack and has specific migitation techniques to diminish them, avoiding the E-Corp network from having to endure the attack.

Reflection

One of the problems I faced with providing solutions to the attack that E-Corp faced was that it was hard to divide the analysis of the case study into the 5 individual areas of management as described by the FCAPS model as the model lacks the particular details of what makes up each one. I feel like there is a lot of overlap between configuration management and accounting management with the rest of the areas especially as the configuration is nearly ubiquitous in the domains it has an effect over. For example, the solution to improve performance by scale horizontally using AWS would require a significant amount of server configuration in order to achieve. As a supplement or even an alternative to FCAPS, the ITIL (Information Technology Infrastrcture Library) model could be reviewed. As opposed to acting just as a model to base the management of a network, ITIL offers implementation details such as methodologies to achieve efficient network management.

2.2 Conclusion

Overall, I think the solutions put forward would make the network more resilient to attacks like this in the future as make it easier for network administrators to manage the E-Corp network and detect and isolate faults if they did occur. As a consequence of this, the users of the service would receive a better, more reliable service and the company would suffer less reputational damage and financial losses by avoiding events like this.