# A world without FCAPS

Dominic Rathbone

May 10, 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 caused their web server to crash.

## Company 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 direct debit card or cheque to over 250,000 registered users. Their product 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 traditional computer. Similar to the web application, users can 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 or request money from another user. In order to support this application, the development team built a new API 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 corners were cut 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 with 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 a 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 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;">
<!ENTITY xmlbomb2 "&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;&
      xmlbomb1;&xmlbomb1;&xmlbomb1;&xmlbomb1;">
<!ENTITY xmlbomb3 "&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;&
      xmlbomb2;&xmlbomb2;&xmlbomb2;&xmlbomb2;">
<!ENTITY xmlbomb4 "&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;&
      xmlbomb3;&xmlbomb3;&xmlbomb3;&xmlbomb3;">
<!ENTITY xmlbomb5 "&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;&
      xmlbomb4;&xmlbomb4;&xmlbomb4;&xmlbomb4;">
<!ENTITY xmlbomb6 "&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;&
      xmlbomb5;&xmlbomb5;&xmlbomb5;&xmlbomb5;">
<!ENTITY xmlbomb7 "&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;&
      xmlbomb6;&xmlbomb6;&xmlbomb6;&xmlbomb6;">
<!ENTITY xmlbomb8 "&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 using 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 also 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, the entire payment gateway and web application were ran from a singular server and when the new API was created, 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 was 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 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 a specific group of IP addresses and user accounts that were sending these requests as the logging from the API was mis-configured to send data to the web application log files as opposed to it's own. 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 by the API as well as the user accounts used in the attack. This allowed for E-Corp to provide services to their web application users but also meant that new mobile users were still unable to use them. Once it 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 application store whilst it was being modified and tested lasting for a further 2 weeks, dissatisfying customers. In reaction to this, the company also had to release a public apology and explanation to their consumers and the businesses using their payment gateway. On top of this, E-Corp 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 10% decrease in the account registration rate over the next month.

# Chapter 2

# Reflective Commentary

The causes and consequences of the network outage will be explained through the context of the International Organization for Standardization's FCAPS model. This model supplies a framework that separates management of a network into particular sections in order to understand and improve on it.

Fault Management how well a major faults in a network are detected, isolated and fixed. This includes preventing future bugs from occurring and recovering from them if they do

Configuration Management how a network controls configuration of the network and changes to it this such as firmware updates or hardware upgrades.

Accounting Management how a network gathers 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 well a access to a network's resources is controlled.

## Fault Management

One of the problems that E-Corp faced was that management rushed the development team to release the product out without being tested and

## Configuration Management

## Accounting Management

## Security Management