National College of Ireland

Project Submission Sheet – 2021/2022

Student Name:	Simon Lowry		
Student ID:	x21168938		
Programme:	MSCCYBE_JANOL_O	**Year:**	2 of 2
Module:	Secure Programming for Application Development		
Lecturer:	Liam McCabe		
Submission Due Date:	12/03/2022		
Project Title:	SPAD - CA 2023 Jan Msc Cyber Security		
Word Count:	8318 words		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Simon Lowry

Date: 28/02/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Conents **Page Number**

Section One: Impact of application security vulnerabilities

45 Marks Total

Research and find an example of a commercially deployed application, written in any language, that has been documented as being breached. (10 marks).

An integral part of application development is the maintenance phase. This important part of the software lifecycle can help to add new features, perform bug fixes and also address identified security vulnerabilities through updates. This process is vital to the continued evolution of software over time. Without it, there wouldn't be a way to continually improve the application in response to customer needs. On top of that, applications that have security vulnerabilities could be endlessly exploited by attackers. This could render those applications too much of a risk for organisations and end users alike if it wasn't present. However, it can have grave consequences if those updates sent out to the users are not secured properly themselves. The application that best illustrates how devastating it can be to not be properly secured when issuing updates is M.E. Doc.

M.E, Doc is an accounting piece of commercial software that's predominantly used throughout Ukraine. It was created by a company called Linkos Group and primarily used in relation to Ukrainian tax systems. At the time M.E.Doc had been used by more than 80% of companies throughout Ukraine. It had around 400,000 clients. [31] This application and its software updates are also responsible for one of the most devastating pieces of malware ever, NotPetya.

This piece of malware was designed to cause as much destruction as possible with little concern about collateral damage and no means to stop it spreading in the malware itself. It's been miscategorized as ransomware but it's actual behaviour is closer to being a wiper. M.E.Doc's poor application security practices and security practices in general led to their own breach and the propagation of this malware.

NotPetya contained similar aspects to its considered predecessor Petya, but this is effectively just camouflage. Normally, with ransomware, there would be some kind of ransom and the user would be expected to pay that ransom within a given timeframe and if they did they would gain a key for unlocking their wares and regain control of their systems and data. With wannacry which was a very potent and devastating piece of ransomware the attackers had a kill switch that was eventually exploited by security researchers to render it neutralized. Effectively, the researchers purchased a specific domain that wannacry reached out to and the malware would shut down if it was live. With NotPetya however, there was no ransom, there was no killswitch. To claim that this is a piece of ransomware is a misnomer since there was no opportunity to actually pay the ransom as the malware causes the systems to abruptly restart and it was actually the attackers applying misdirection. Instead, this piece of malware was intended to cause maximum damage and the creators had no care about any collateral damage produced.

NotPetya propagated at an unprecedented speed using a number of different mechanisms to do so. It capitalized on the same exploit that Wannacry used, the NSA zero day that was released online known as Eternal Blue and also Mimikatz. Mimikatz which was originally a proof of concept to prompt Microsoft into fixing a security issue that allows it to obtain passwords out of RAM. NotPetya was thus able to spread either by exploiting vulnerable systems which had not yet been patched for Eternal Blue or otherwise making use of the passwords it obtained with Mimikatz. [22] Other tools NotPetya tries to use also include other non-exploit windows operations such as PsExec and Windows Management Instrumentation to further their propagation efforts as well. [23]

Identify and critically assess the aspect of the application that lead to the breach. Identify a solution that should have been implemented prior to the breach as a preventative measure. (10 Marks)

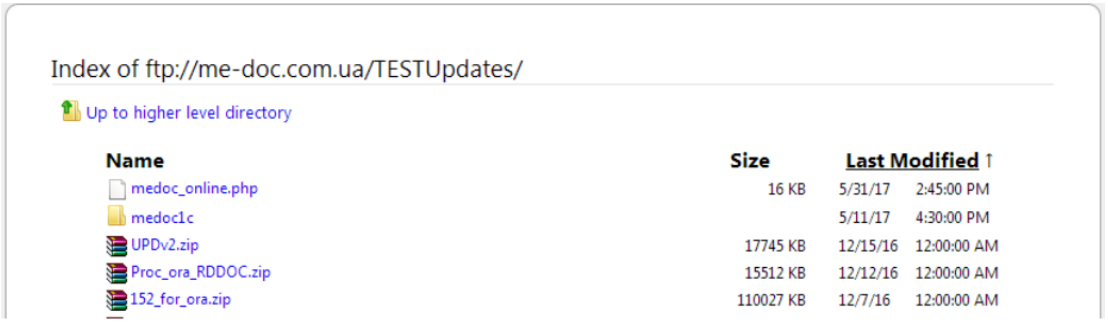
```
Starting Nmap 6.40 ( http://nmap.org ) at 2017-06-29 06:46 EDT
Nmap scan report for upd.me-doc.com.ua (92.60.184.55)
Host is up (0.12s latency).
rDNS record for 92.60.184.55: reduk-55.colo0.kv.wnet.ua
Not shown: 994 filtered ports
PORT    STATE  SERVICE
20/tcp closed ftp-data
21/tcp open   ftp
22/tcp open   ssh
25/tcp closed smtp
53/tcp closed domain
80/tcp open   http

Nmap done: 1 IP address (1 host up) scanned in 11.04 seconds
root@vps94775:~# ftp 92.60.184.55
Connected to 92.60.184.55.
220 ProFTPD 1.3.4c Server (WWW ftp server) [92.60.184.55]
```

Above is an nmap scan of the server that M.E. Doc had for their updates at the time of compromise. It's ip is at 92[.]60[.]184[.]55 and it's domain resolves to reduk-55[.]colo0.kv[.]wnet[.]ua, via reverse lookup. Customers that were using M.E.Doc would receive from this update server that is diabolically insecure.

Cisco's Talos team was brought in by Linkos Group to perform retrospective analysis and help remediate the situation. It was discovered that attackers had compromised the update server of Linkos Group (previously known as Intellect Services). [24] Linkos Group was subsequently been shown to be direly short on security practices. Their servers had not received any updates at all since 2013. That's 4 whole years without a single update. They had ports left open for ssh and ftp as mentioned. They were using out of date versions of ssh, ftp and it's web server software Nginx. FTP was using PoFTPD 1.3.4.c which had a number of remote code execution exploits whereby attackers would be able to obtain root privileges on the system. These were also very basic exploits with step by step instructions available online for them. [25] They also had OpenSSH v.5.4 which has a host of other known exploits as well. So the compromise of such a web server would be comically easy for even very

basic attackers and Linkos group was attacked by an APT which would have had a field day getting into these systems.

There was some logging evident on their servers which showed the attackers were sitting in their servers for 3 weeks before they completed their tasks. Evidence of the attackers performing privilege escalation was evident in the logs as well.



PHP webshell on M.E.Doc update server

[27] Here we can see the php webshell the attackers had placed on their servers after compromising it. On top of this, there were not one but *three* releases of the software shown to be containing the backdoored module of NotPetya.

Software release versions containing the module & their release date:

- 10.01.175-10.01.176, released on April 14th 2017
- 10.01.180-10.01.181, released on May 15th 2017
- 10.01.188-10.01.189, released on June 22nd 2017

[26]

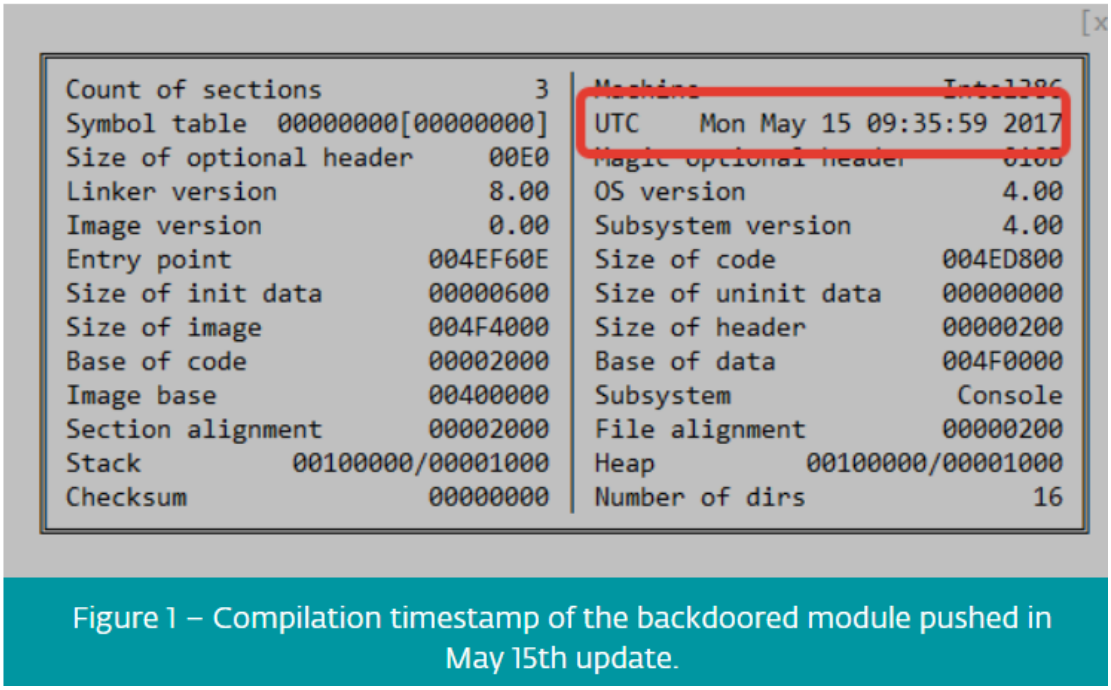


Figure 1 – Compilation timestamp of the backdoored module pushed in May 15th update.

[26]

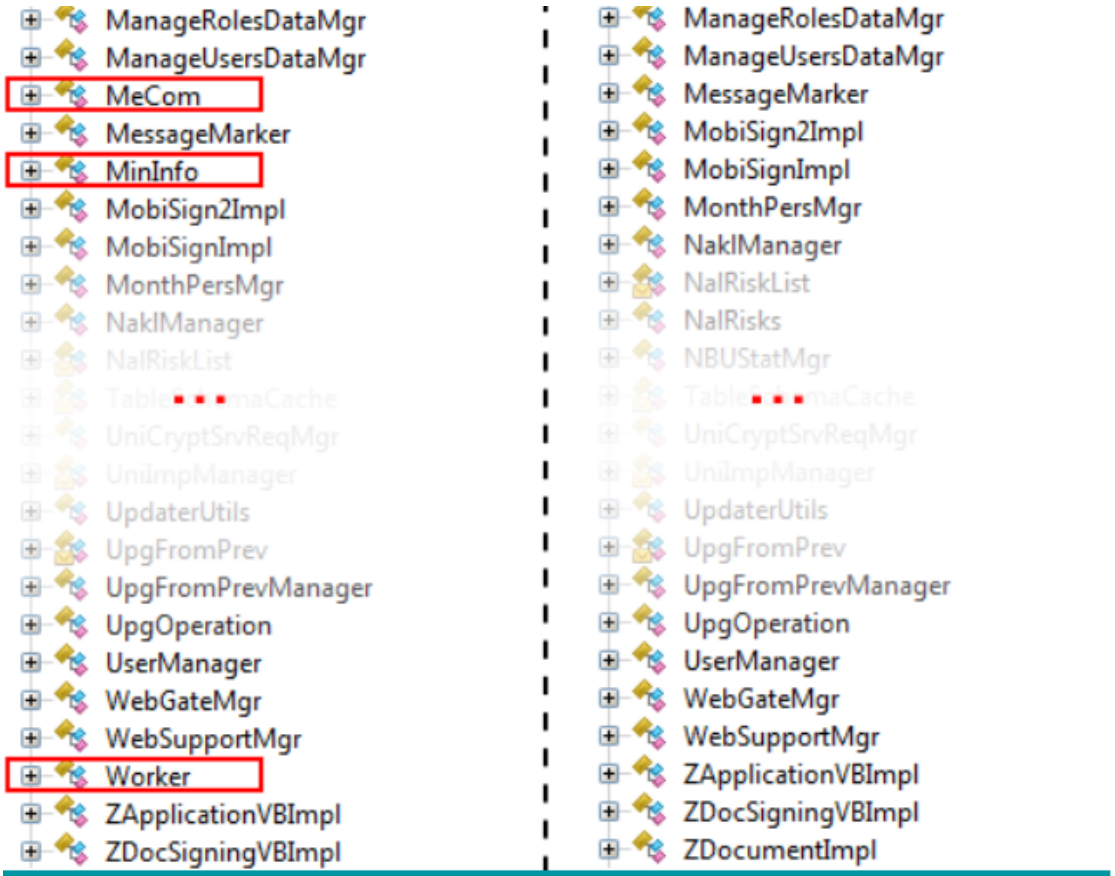


Figure 2 – List of classes in backdoored module (at left) and non-backdoored (at right).

[26] Not a single person in their development appears to have recognised in three releases, that were spread over *two months*, that they contained these additional malicious modules. The modules were MeCom, MinInfo & Worker as shown above. These modules would end up creating backdoors into thousands of systems that had the accounting software M.E.Doc installed with an update sent out to their systems. This would be the first phase that would be followed by downloading a debilitating piece of malware, NotPetya, onto all of these systems.

A report done by Accenture showed that 43% of cyber attacks occur on SMEs! [28] The fallacy of "it will never happen to us" that many SMEs can have is an accident waiting to happen as shown here. Threat actors tend to opt for low hanging fruit where possible since it involves less effort for them to obtain their goals. They are potently aware of the lack of security controls in place in these companies. Organisations like these who have next to no security practices in place are ripe for the exploit. This lack of security also lowers the barrier of entry for less sophisticated attackers to try their hand at attacking them. Without performing the necessary due diligence to ensure their systems are secure they left the door open for attackers to be able to perform these devastating Software Supply Chain Attacks.

Software supply chain attacks have been massively on the rise in recent years growing by a reported 300% according to Argon Security. [29] Supply chain attacks are able to infiltrate what could be otherwise secure networks and systems by leveraging the established trust of applications used by those organisations or individuals. In this case, the updates from Linkos Group were digitally signed by them and provided a

great way for attackers to bypass defences on their intended target organisations. From there they were able to perform their malicious activity. They can provide a means of causing damage, disruption and otherwise exploiting the users and organisations these compromised applications are used on. The attackers had found their way into damaging targets through the use of these dependencies.

Preventative Measures for Linkos group:

- **Security training, security training & more security training:** It's clear that both the developers and those managing the servers have a dearth of any knowledge of any kind of security. This is likely in their application as well as demonstrated by their inability to even recognise three new modules in their applications for over two months. The team and organisation requires some hiring of security engineers and application security engineers to help get the teams of developers to a much better standard of security practices.
- **Limit updates releases:** Limiting updates to specific times and days of the week could help to reduce the capacity of attackers to push an update.
- **Diligently assess & Monitor releases:** Prior to performing a release, the contents of releases should be inspected and any unknown files or packages included should result in the release not happening until further inspection occurs.
- **Limit file types that can be released in updates:** Apply an allowlist of file types for releases. This reduces the attack surface for which attackers can include file types used to exploit systems.
- **Apply Least privilege:** Apply least privilege to authorisation of employees as well as to servers and the application M.E. Doc users. Servers should only have services, ports accessible that are necessary to perform duties and no more. Ports such as port 21 ssh and ftp ought to be closed unless there is business justification for keeping them open.
- **Apply separation of duties:** If the update process required two admins to proceed this could have made the attackers ability to push updates more difficult.
- **MFA:** Require multi-factor authentication for performing releases and to access web servers.
- **Have a patching process in place for Servers and application dependencies alike:** utilizing an effective patching process is a vital part of maintaining security for applications and servers alike.
- **Add alerts to the logging and incidence response capabilities:** Logging was evident on some of these servers in retrospective analysis by Talos. However, these logs were not triggering any alerts and as such didn't provide the level of protection they could have otherwise provided. This could be collected by SIEM or some other mechanism so they can trigger an alert. These alerts could be addressed accordingly by an internal or external incidence response team.
- **Network segmentation:** Applying network segmentation makes it much more difficult for attackers to move laterally when they have compromised a subnet. This can decrease the likelihood of
- **Periodic application and network penetration testing:** this can give a simulated effort of what a hacker would do. It would have highlighted the gaping holes in the servers at the very least and would have no doubt provided more useful insight to Linkos group who are lacking in security insight and practices.

- **IDS & IPS systems -** Apply IDS and IPS systems to help detect and prevent attackers looking to infiltrate their networks

Identify and critically assess the consequences of a similar breach in terms of

a. The end users of an application (12.5 marks)

Within 45 seconds it brought down the entire network of a Ukrainian bank as well as a transit hub being fully infected in 16 seconds.[8] There was a recklessness to this attack that resulted in 64 countries overall ending up being affected by this. While it's clear that Ukraine was the primary target with infections being 3 times as high as its nearest other country, the US, it caused havoc across the globe [9]. In Ukraine, 10 percent of all computers in the entire country were wiped completely. It even demolished computers that were active at the Chernobyl cleanup site. [8] NotPetya managed to knock out 6 power companies, 4 hospitals, 22 banks, two airports as well as many government agencies. Throughout Kiev ATMs completely stopped working and people were not able to make any withdrawals. Businesses such as the pharma company Merck lost upwards of $870,000,000, the delivery company FedEx $400,000,000 to name a few. Overall, the damages allegedly arrived at $10 billion.

The attackers cleverly and ruthlessly performed the attack the day before Constitution Day in Ukraine. This resulted in less people available to actually monitor systems and many people being on holidays at that very moment. [30]

We're able to see the importance of due diligence in the selection of software vendors that we're willing to have in our organisations and home systems. The effect of not doing so as shown with Linkos group who were entirely negligent in their security controls and practices can have dire downstream consequences. Some questions worth considering: Do these software vendors have adequate security programs, protocols, practices and controls in place? Do they address security issues when they're pointed out or do they try to deny them to save face? Neglecting to perform this thorough and ongoing investigation can have horrendous consequences as demonstrated here. As mentioned previously these types of Supply Chain Software attacks are on the rise and the necessity of taking their consideration seriously has never been greater.

The danger of being able to bypass most security controls by exploiting the trust in place is huge. A multitude of even more additional devastating attacks could have occurred through the exploitation of this supply chain attack vector.

Attack types that did occur and others that were possible:

- **Backdoors:** The attackers could have looked to deploy multiple means of persistence on the update servers. From that they could look to exploit Linko Group again and again. They could further use their update process to send out more backdoors and gain remote access to a whole host of different companies' networks. Again, this would cause dire reputational damage for Linkos Group and undoubtedly lose them further customers and revenue.
- **Denial of service**: directly caused as evidenced with Maersk & Merck and caused huge loss of money. The business of Maersk was completely stifled with hundreds of trucks literally lined up outside their docking areas and unable to enter. Merck were equally unable to produce some vital pharmaceutical drugs due to this breach. Attackers were able to either brick their systems or wipe them making operations next to impossible.

- **information stealing:** stealing of intellectual property & other sensitive data. Subsequent extortion attempts where potentially embarrassing information was acquired could also follow. Attackers would be inside the networks of customers and unless proper security measures were in place they could laterally move through the network and take their time and find as much valuable information that they could and exfiltrate it to likely other compromised servers to protect their anonymity.
- **Lateral movement to database servers & acquiring user information from databases:** This could result in GDPR breaches and related fines to follow. If the databases are not properly secure and the data was not encrypted, attackers could look to compromise the database servers through using known exploits either for the kind of database running or the server itself. By gaining access they could then extract and spread that information online to embarrass the user or sell it on the dark web.
- **Planting other malware on their system** such as keyloggers, or spyware obtaining sensitive information from companies and other users. This again gives the attacker increased persistence on their systems and gains them information they should not have access to.
- **Ransomware:** NotPetya posed as ransomware when it was actually a wiper type of malware. Actual ransomware could also have been deployed through these packages instead. This could encrypt their hard drives as well as boot loaders until a ransom is paid. Here the attackers may or may not give the users their systems and data back & the advice in these circumstances is not to pay the ransom and ensure that you have backups in place and disaster recovery practices and policies.
- **Remote code execution:** attackers could perform arbitrary code execution with having remote access. With free reign of a system, the attacker would only be limited by their imagination and expertise.

The absolute necessity of having an effective patch management strategy in place in businesses, schools, healthcare providers, government agencies can not be understated. A number of the exploits used by NotPetya are completely patchable and rendered ineffective. At the same time, features of Windows are still exploitable today with tools like Mimikatz which Windows has not completely neutralized as of yet. The benefits of a deeper analysis reveal some IOCs that can help to detect and prevent malware throughout the cyber kill chain. Beyond that regular backups are very important to protect against wipers and ransomware alike.

Additional Preventative measures for users:

- **disable SMBv1 & employ least privilege:** SMB has been at the centre of a number of potent malware attacks from notpetya to wannacray. Disabling that and other unnecessary ports and services when they are not needed can help mitigate this attack vector.
- **adding a rule on your router or firewall to block incoming SMB traffic on port 445** -> unless there is a business justification for having SMB traffic coming in on a given network, it ought to be blocked.
- **patching** -> keeping up to date on the latest updates for software and system makes attackers job much more difficult and may help to fend off those looking for an easy entrance through known exploits.
- **maintaining awareness of current high profile breaches which could have downstream impacts on their systems.** Ongoing vigilance by security

engineers and security focals of what breaches and latest could have negative effects on their company allows you to act fast and potentially protect their organization from a downstream impact.

- **have a means of revoking software access to systems in the event of Software Supply Chain attacks**: being able to add a given software to a blacklist of software or a means of stopping it's use could help to protect against software supply chain attacks when that software has been shown to be no longer secure.

b. The owner/users of an application (12.5 Marks)

Interestingly Linkos Group or Intellect Services as they were known as at the time, refused to take any responsibility for the breach, asserting:
"We studied and analysed our product for signs of hacking - it is not infected with a virus and everything is fine, it is safe," said Olesya, managing partner at Intellect Service. "The update package, which was sent out long before the virus was spread, we checked it 100 times and everything is fine."

Through this we're able to see the sheer reluctance of some organisations to truly take security seriously and their responsibility towards it and their users. Even in the face of investigations from the Ukrainian police and all indicators pointing to them they refused their culpability. Ukrainian police were forced to seize the company's servers as additional attacks were underway and even still they didn't admit their role in this. [27] We can see their naivety in assessing the situation and inability (or unwillingness) to identify signs of compromise in their software process and systems as illustrated earlier. This is highlighted in Talos investigation and other investigations afterwards. Intellect Services poor security practices and poor choices in how they responded to the breach led to one of the most devastating cyber attacks in history.

Through this we're able to see the sheer reluctance of some organisations to truly take security seriously and their responsibility towards it and their users. It's fairly common in SME businesses and no doubt a big contributing factor in their selection for exploit. Even in the face of investigations from the Ukrainian police and all indicators pointing to them they refused their culpability. Ukrainian police were forced to seize the company's servers as additional attacks were underway and even still they didn't admit their role in this. [27]

According to an insurance bee survey, 83% of SME's do not recover from breach and have to close up permanently. [28] Intellect Services was permanently scarred as a company as a result of this with their name attached to this breach. The reputation damage was so pronounced that the company changed their name from Intellect Services to Linkos Group. Interestingly, they did not look to change the product name of M.E.Doc. [32] The exact revenue figure that this breach cost the company does not appear to be publicly available. It's safe to speculate that this would have lost them a vast amount of customers, revenue and there were probably some people who lost their jobs within Intellect Services as well. Hence why there ends up being so many closures to SME's after a breach as denoted earlier.

Since these attacks were carried out and no-one was able to recognize them. The attackers had a lot of freedom to be able to carry out further attacks had they wanted

to. Many of which would be similar to what was mentioned in the previous section including:

- ransomware
- intellectual property theft
- denial of service by wiping their servers
- impersonation

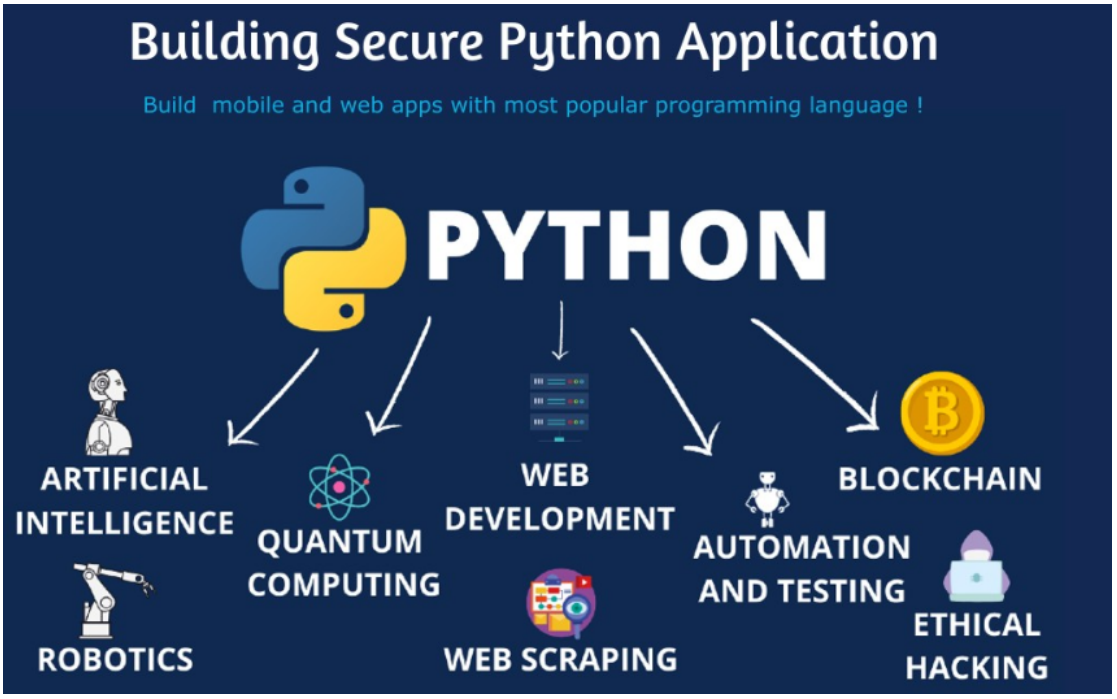
Ransomware: After exploiting the update server for a means to launch their software supply chain attack, the attackers could then have looked to extort money from Linkos Group through ransomware. They already had access to their network and thus it may not have been that difficult to target further systems on the network with ransomware. They could have let loose a strain of something like wannacry or other ransomware that could spread across the networks on port 445. Thus could have brought their operations to grinding halt and resulted in more revenue loss. It's unlikely that they had backups in places as well if the rest of their security practices were anything to go by so this could have severely damaged the companies ability to function. They likely would have had to rebuild networks and devices from scratch.

Stealing Intellectual property & compromising customer data: intellectual property serves to be a competitive advantage for companies and can be sold on the dark web by hackers or released online to cause some reputational damage. Due to the fact that servers were not being updated and ports were left wide open, it would not be that hard for hackers to move laterally on the network. Attackers would have been able to locate & exfiltrate intellectual property at whatever speed they would have liked since Linkos group did not have any alerting or tools such as an IDS or IPS to detect their presence. If databases were as poorly secured as the rest of the infrastructure, hackers could have obtained customer data, manipulated or corrupted that data or again sold it on the dark web. This kind of exposure would have led to further GDPR fines and even greater reputational damage.

Denial of service: The attackers had full access to their network and one of their primary servers, they could have performed the same actions that NotPetya performed on other systems and completely bricked their devices. They already demonstrated that they have malware capable of this and could have leveraged that to directly damage Linkos Group themselves and not just their customers.

Impersonation: The attackers could have also looked to leverage one of M.E.Docs features to send out messages and used that to impersonate the company themselves and write messages to customers. Those messages could be designed to hurt the reputation of Linkos Group further and result in loss of customers and further revenue. This may also be possible if Linkos Group had inhouse email servers and the attackers could have posed as the CEO of Linkos Group and brought further damage to the company.

Section 2 - Part 1



[1]

Python is a programming language that was first released in 1991 and is widely used in application development. In 2022, it even surpassed Java as the most popular programming language according to the TIOBE index. [2] Python has become a particular staple also for use by cybersecurity professionals. The fact that it's easy to use, easy to read data structures, it offers powerful libraries and a very clean programming language makes it an excellent choice for the cyber industry. It has a whole range of libraries tailored specifically for the security industry. It can be used to perform penetration testing, analzye malware, perform scanning and evaluate different cyber threats.

Due to the fact that Python is open source, there is a big community of people who look to address security concerns as they arise. This is one of the benefits of having an open source language where many developers are scrutinizing its code on a regular basis.

One of the big changes in Python occurred with the upgrade from Python 2 to Python 3. There was a big shift syntactically and required a lot of developers to rewrite a lot of code. Moving to Python 3 did bring, however, some very valuable additions on the security front. It brought about the introduction of OpenSSL as one of the core runtime libraries. A library which provides secure communications over the internet protecting against eavesdropping with both TLS and SSL encryption possible. It also brought in default unicode encoding of Strings as well as changes to error handling and the semantics surrounding error handling. It also saw changes to eval and input functions providing greater improvements on functions for sanitizing input for numeric values. This can play an important role in protecting against malicious and malformed input as well as code injection. [3] Earlier versions of Python had some

security vulnerabilities including remote code execution vulnerabilities in version 2.7. [4] This particular version is no longer supported since 2020 and it's recommended to use at least version 3 or ideally the latest version to ensure you're up to date with the latest features and security updates. Python's current version is 3.11 as of October 2022.

Packages from Python can be installed via the command line tool Pip and those packages are stored on Pypi. This particular package distribution has had its own issues and highlights with a study highlighting 46% of those libraries having a security vulnerability in them. [5] Packages on Pypi do not go through any form of security review before being published and this is something that very much puts the onus on the developers to put in their due diligence when selecting a library for use from Pypi.

This leads to one of the common tips for using python to use it in a virtualized environment in case one of the packages is vulnerable and thus it won't be able to have an effect on other projects or your main system. Also when installing packages be sure that it is the legitimate package you want to install and updated as well. For example the package "00Seven" is a very different package from the package "000Seven" despite the very similar names. It's important to check for known issues in any of the packages and ensure that the provider of the package actively addresses security issues when they have identified in a timely manner.

A good example of this is the cryptographic library pycrpto. This particular project has had security issues in it for some time and has not been addressed as the project is no longer being updated. An alternative here could be to use pycryptodome instead:
`pip install pycryptodome`
It offers a whole host of functionality such as accelerated AES, Eliptic Curve Cryptography, SHA-3 & bcrypt hashing algorithms as well as RSA & DSA Key generation to name a few of it's capabilities. [6]

Python offers three different ways of importing packages as well. One that is absolute, another relative and finally the last is implicit. Implicit paths can operate with not requiring the location to be specific. This can result in running a malicious package or a trojan horse pretending to be the real thing or has been expanded out from something real to contain other malicious code. Selecting to use absolute paths instead can protect against this and ensure you're not taking any risks.

Another security practice to follow with python is to set debug to false. Frameworks such as Django have been to set debug to true by default. This leads to errors being output that could give attackers information that could be useful for them in exploiting the application. [7] String formatting can also open up security issues in Python. Some of the ways of formatting strings introduced in Python 3 have been shown to leak sensitive data. This includes f-strings and str.format(). A more effective alternative is to use the Template class that's part of the String class. This can be good for generated data and handling user input. [8]

Another part of python programming that requires security is Http requests. A prominent library in this space is Requests. Here, what version of this libraries you're using becomes a security consideration since for example, earlier versions of Request library has its own vulnerabilities that can well be exploited otherwise. Another library that requires best practices is urllib as this can be susceptible to request

smuggling. [8] Deserialization ought to be performed only from trusted sources to prevent the unintended execution of malicious code. This could look to corrupt user data, perform remote execution of arbitrary malicious code, denial of service or abuse of logic. [9]

In order to protect against this, make use of packages that ensure safety of the data in a given sandbox before deserializing it. One package to avoid in this process is the Pickle package provided in the framework Flask as it's notorious for being insecure. Additional protections include using a digital signature to corroborate the data during your input process. Making use of language agnostic data formats like JSON can also help. Deserializing the data otherwise in a low privilege environment as well can be another mitigation. [10]

Input validation is a vital aspect of being able to protect against a variety of injection attacks from SQL injection to XSS. One Python library that can help in the process of input validation is re. RE stands for regular expression and allows you to check if a particular string matches the expected and allowed input for our given feature. [11] A Python library that can be helpful for validating file details is os, it can help you to obtain information such as the type of file, it's size and so on. This library can help protect against file upload vulnerabilities. [12] Another useful tool is Bandit. It's a static analysis code scanning tool which can help to find vulnerabilities in your code and yaml files as well. It also points directly to the line where the vulnerability may be and ranks the vulnerability from high to low.

Veracode's State of Software Security Volume 11 provides an examination of flaws in applications by programming language and their corresponding frequency. [13][14] The languages examined were Python, Java, Php, Javascript & C#. This research had scans performed across 130,000 different applications. It highlights that 74% of applications have at least one security vulnerability in them. It also has a lot of content specifically around Python.

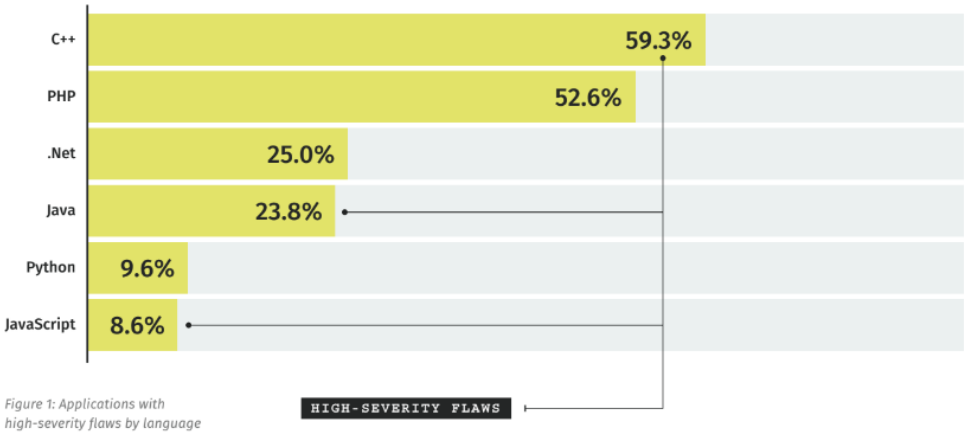
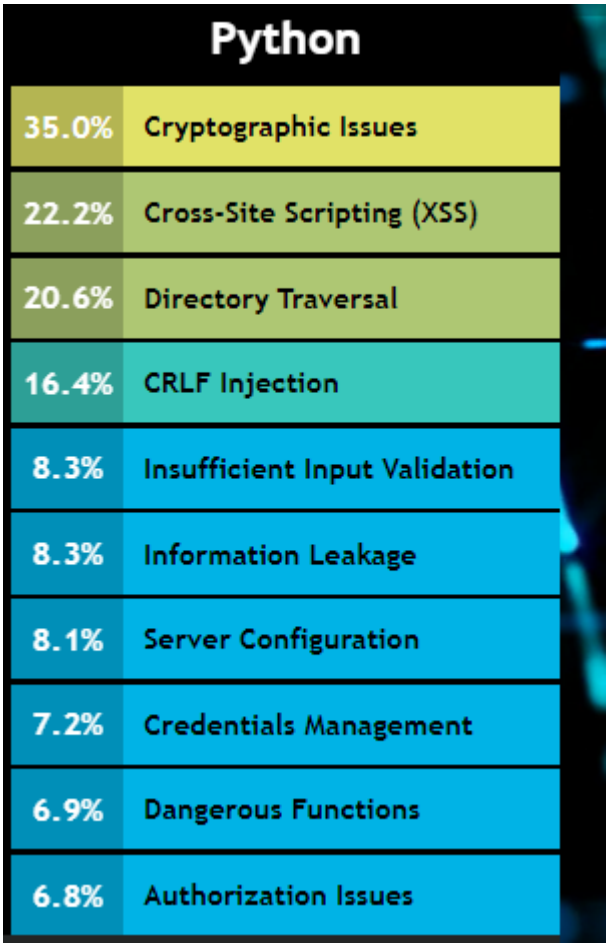


Figure 1: Applications with high-severity flaws by language

59 percent of C++ applications have high (and very high) severity flaws, compared to just 9 percent of JavaScript applications. It's flipped with Java, as only 24 percent had critical flaws this time around.

Python was shown to be having a lot less high severity flaws with only coming out with 9.6% which was drastically lower than languages like C++ at 57.3% and Php at 52.6%. It was only bettered by Javascript on this metric.



[15]
Knowing the kind of security flaws that tend to be more prevalent with a selected language can help to aim more focus in those areas for developers. For Python, that would be flaws around Cryptographic issues (present in 35% of applications scanned), Cross site scripting (22.2%) & Directory traversal (20.6%) in particular. Those were the top three identified.

Those **cryptographic issues** can include not properly validating certificates, as well as using broken cryptographic algorithms, lacking in encryption strength & storing sensitive information in cleartext. These can be remediated by properly adhering to secure cryptographic practice. By understanding the capabilities of different cryptographic algorithms, their strengths and weaknesses and where they are best applied we can help to protect against this most frequent issue in Python applications.

Cross-Site Scripting is a form of injection attack whereby the attacker leverages client scripts that are malicious in nature to exploit a web application. [19] It routinely features in the OWASP Top 10, currently at number 3 as part of Injection attacks. There are also a number of CWE references for XSS (CWE-79, CWE-352 & CWE-113) and it's included in the
SANS Top 25 Most Dangerous Programming Errors. [16] [17] [18] There are three types: Reflected XSS attacks, Stored/Persisted XSS attacks & Dom based XSS attacks.

Dangers of XSS:
- could steal session data and then perform actions illegitimately as that user such as transfer money to the attackers.

- could manipulate html elements on the web page defacing the website
- automatically download some malware to the user's system such as a keylogger. [20]
- control the browser remotely

Preventing XSS:
- Server side validation of the input using whitelisting of values/characters.
- Making use of tried and tested HTML encoding functions or url encoding for the http response.
- Set both the secure and httpOnly cookie flags. HttpOnly cookie flag prevents client side scripts from being able to change cookie information and the secure cookie flag forces the cookies to be only transferred https.
- Apply same origin policy to limit executable scripts to the origin.
- Don't use vulnerable functions for untrusted input such as the html function eval(), element.innerHTML, element.outerHTML, document.write(), document.writeln(). Use safer functions such as textContent() or innerText() instead.

Directory traversal is where attackers are able to gain access to restricted directories and files. They were found in almost half (47.8%) of applications in the Veracode report. By applying secure principles such as deny by default this can help here. Given users ought to be able to only access paths based on their privileges delegated to them and nothing more to adhere to least privilege as well. It can also help to block certain user input for paths such as ../ for example and also keeping dependencies up to date and running frequent static analysis scans which are able to identify these vulnerabilities.

In conclusion, this report highlights the importance of considering programming languages as part of secure application development. Some languages have a higher prevalence of flaws and this ought to be taken into account when choosing a language for a new application. Python does not feature as highly in terms of security vulnerabilities despite being a very popular language. It does however require its own considerations for the types of vulnerabilities that more regularly occur when using the language. Making use of some of the best practices mentioned can go some way towards helping your python applications to be more secure.

Section 2 - Part 2

1. Choose any standalone application and discuss, if you were to rewrite it in either C#, C++ or python, how would you apply the elements researched in part one of section two to make the application secure. Provide examples of code to support your answer and clearly list and document the features you are securing (6 features). (30 Marks)

Q 2.

The standalone application selected to rewrite is Skype. The six features which will be re-designed from a security perspective are: authentication/login, session management, searching for users, auditing and logging, profile image upload, data in transit security & ensuring a valid certificate on the backend.

1. Authentication/Login

Here we'll look to protect against brute force with account lockout, store passwords as hash (with strong hashing algorithm), password complexity requirements, mask password.

Password Complexity Check

```
1. import re

2. def validatePassword(password):

3. if(bool(re.match('^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])
   (?=.*[!@#&()–[{}]:;',?/*~$^+=<>]).{10,20}$',
   password)) ==True):
            return true
 4.  else:
            return false
```

Line 1: imports the regular expression library which will be used to compare the password to which will ensure it meets password complexity requirements.
Line 2: start of the method validatePassword which takes in one parameter, the string password.
Line 3: perform password complexity check: Password must contain at least one uppercase character, lower case character, special character, and be between 10 to 20 characters long.

Requiring passwords to be of length 11 or greater provides great protection against brute force attacks.

Hashing Password

```
1. #!/usr/bin/python
```

```
2. import bcrypt

3. def generatePassword(plaintextPassword):

4. salt = bcrypt.gensalt()
5. hashedPassword = bcrypt.hashpw(plaintextPassword,
   salt)
```

Line 2 is importing the bcrypt library. This is a hashing algorithm library.
Line 3 plaintext password is set as parameter for method generatePassword.
Line 4 is where a salt is generated.
Line 5 is where the hashed password is generated from the plaintext password and salt.
Utilizing bcrypt for password hashing is due to it's strength relative to other options. It's stronger than PBKDF2, as well as argon2 and scrypt for memory requirements that are less than 4MB which is the case here.

Account Lockout:

```
            currentNumAttempts = attemptsEntity.getAttempts();

            if (attemptsEntity.getAttempts() > 3):

                  isAccountLockedUpdated = true

                  userEntity.setAccountLocked(true)

                  userService.updateUser(userEntity,
isAccountLockedUpdated);

                  logger.info("Exception occurred: " +

                  "Login attempts have exceeded limit. Account is now
blocked.");

                  raise LoginAttemptsLimitReachedException()
```

Here we're obtaining the current number of attempts of a given user trying to login. We check to see if it's gone beyond three attempts. if it has, we set the user to have their account locked. This is saved to the database, the logger outputs the error message and an exception is thrown.

2. Session Management

```
tools.sessions.httponly = True
tools.sessions.secure = True
tools.sessions.timeout =30
```

Here we're using cherryPy library to ensure our sessions are protected with both the httpOnly and secure flags set. This ensures that the cookie can't be modified on the client side and forces it to be sent over https ensuring that it's encrypted. It also ensure

the session has a timeout at 30 minutes forcing users to re-authenticate. These would be added to a configuration file.
[21]

These security controls help protect against cross site scripting attacks as well as session hijacking attacks.

```
cherrypy.session.regenerate() cherrypy.session['sessionId']
= cherrypy.request.generate_id()
```

This code snippet is where we're creating a session. This would be used when the user reaches the login page. It would also be called after the user has been successfully authenticated or created. This protects against session fixation by creating a new session.

```
session = cherrypy.session
session['sessionId'] = null
```

This line would be placed in the logout method and would delete the session when the user is logging out.

3. Searching for users
This will require a database call to obtain users that are searched for. This will need some SQL injection protection & also XSS protection:

```
          import MySQLdb as mdb
          import re as regex
          if(bool(regex.match('^(?=.*[A-Za-z-',.
          ]).{1,20}$', username) == false):
               raise  InvalidUsernameCharacters()

          con = mdb.connect('localhost', 'testuser',
          'test623', 'testdb')
def getUser(username):

     with con:

          cur = con.cursor()

          users = cur.execute("Select * from Users where
          name= %s",(username))
     return users
```

Line 1: Here we're importing a library MySQLdb to connect to a MySQL database and perform our search query.
Line 2: Import regex library for allowlist of characters we will permit.
Line 3: Validate characters input by users to ensure only allowed characters are entered to protect against both SQL injection and XSS attacks.
Line 4: The connection is established.
Line 5: A search query is performed with a prepared statement. This separates the execution of the query from the input of the user thus protecting against SQL injection.

4. Auditing and logging

```
import logging
import logging.config
from random import randint
from flask import Flask
from flask_log_request_id import RequestID,
RequestIDLogFilter

logname = "service.log"
handler = logging.FileHandler(logname) # here we set the
log name

handler.setFormatter(
     logging.Formatter("%(asctime)s:%(levelname)s:%(reque
     st_id)s - %(message)s")
) # i make the format more compact
handler.addFilter(RequestIDLogFilter())  #  Add request
id contextual filter
logging.getLogger().addHandler(handler)


def myGenericMethod():
     logging.info('Some message here’)
```

This code is going to make use of both the logging library and also flasks capabilities to produce a request id. It’s setting up the log to contain the request id along with other time based information. Then there is a sample of that logger being called in the method myGenericMethod.

Sample of how this would look like:

```
2021-01-19 04:43:52,482:INFO:bcc58751-fd8e-4de7-93c5-0a9bed68ba30 -
```

It contains the date and time of the request followed by a unique request id. This would be followed by whatever message is added by the developers afterwards.

The trace id is particularly important for both security concerns and development support. Every request that goes through will have one unique trace id and from that we're able to see all of the services and actions that occur on that given request. This could be particularly useful in the event of a security incident or alert. It will allow security personnel to be able to trace what happened throughout a given request to the application. The logger also contains a timestamp with the time and date included which again can be important information in a security alert and to add to this, the IP address of the client making the request is also logged at the beginning of every request. So if a given attacker is performing malicious actions over many different requests, we'll be able to combine these pieces of information from the IP, timestamp, trace id and accompanying actions to create a picture of what the attacker has been doing

5. Profile image upload

Filetype security.

```
import os

MAX_FILE_SIZE = 1024 * 2048
1. if (userIsLoggedIn()):
     validateFileDetails(filename)

def validateFileDetails(filename):
     FILETYPE_ALLOWLIST= [“.jpg”, “.png”, “.gif”]

2.    if (filename != ‘’&& filename.count(‘.’) > 1
&& len(filename) < 20):
          file_extension =
     os.path.splittext(filename)[1]

3.        if (file_extension not in FILETYPE_ALLOWLIST):
               return “File type is not acceptable.”

          fileSize = os.path.getsize("/path/to/" +
     filename)

4.        if (fileSize > MAX_FILE_SIZE):
               return “File size is not acceptable.”
          return “File details validated”
     else:
          return “Filename is not acceptable.”
```

Line marked as 1: Ensure the user is logged in before validating any file details.
Line marked as 2: Ensure the filename has characters, does not contain more than 1 full stop. Ensure the length is no greater than 20 characters.
Line marked as 3: Ensure the file extension is only either .jpeg, .gif or .png. Reject all other file types.
Line marked as 4: Ensure the file size is not exceeded. Size is obtained in bytes, limit is just over 2MB.

We’re looking to ensure to limit the filetype inputs to an allowlist of acceptable types to protect against malware of different varieties being uploaded. Rejecting more than 1 full stop helps to protect against double barrel file extensions obscuring the real file types. This can be a tactic used by attackers. Ensuring the user is logged in also restricts this functionality to those who are authenticated.

6. Securing data in transit & ensuring valid certificate on the backend

```
import requests

response=requests.get('https://myBackendServer.com/')
```

Here we can use the requests library to perform a request to the backend. It will only return a 200 response if the website, in this case, our backend server has a valid SSL certificate. Otherwise it will throw an SSLError like the following:

```
requests.exceptions.SSLError:
HTTPSConnectionPool(host='www. expired.badssl.com',
port=443): Max retries exceeded with url : / (Caused by
SSLError (SSLCertVerificationError(1,
'[SSL:CERTIFICATE_VERIFY_FAILED] certificate verify
failed: unable to get local issuer certificate
(_ssl.c:1131)’)))
```

This helps ensure that we are dealing at least with a website that has a valid certificate.

Generate a certificate through the relevant certificate authority being used by the organisation. This helps to provide trust in communications with the server backend of application. A sample self signed certificate can be generated with:

```
openssl req -new -x509 -keyout .ssh/key.pem -out
.ssh/certtificate.pem -days 365 -nodes
```

Then using the newly created we’re able to establish TLS connection with the following code:

```
import http.server
import ssl

httpd = http.server.HTTPServer(('localhost', 443),
http.server.SimpleHTTPRequestHandler)
httpd.socket = ssl.wrap_socket (httpd.socket,
certfile='./certificate.pem', server_side=True,
ssl_version=ssl.PROTOCOL_TLS
httpd.serve_forever()
```

This simply starts up our HttpServer at localhost on port 443. It makes use of our self-signed certificate and also TLS. Thus, ensuring the data sent from the desktop application to the backend application is encrypted over TLS. As mentioned previously, this certificate would be swapped out for a CA signed one for any production system.

References/Bibliography

[1] Digiprama, How is Python The Most Popular & Secure Programming Language for Software Application Development, March 2022, Online [Available]: https://www.digiprima.com/blogs/how-python-popular-secure-programming-language-for-software-application-development

[2] Statistics Times, Top Computer Languages, June 2022, Online [Available]: https://statisticstimes.com/tech/top-computer-languages.php

[3] B.Cipot, Synopsys, Six Python Security Best Practices, November 2021, Online [Available]: https://www.synopsys.com/blogs/software-security/python-security-best-practices/

[4] CVE Details, Online [Available]: https://www.cvedetails.com/vulnerability-list/vendor_id-10210/product_id-18230/Python-Python.html

[5] J.Ruohonen, A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI, July 2021, Online [Available]: https://arxiv.org/abs/2107.12699

[6] Pypi, January 2023, Online [Available]: https://pypi.org/project/pycryptodome/

[7] D.Roche, Sqreen, Top 10 Python Security Best Practices, January 2020, Online [Available]: https://blog.sqreen.com/top-10-python-security-best-practices/

[8] M.Hollander, May 2020, Online [Available]: https://www.securecoding.com/blog/python-security-practices-you-should-maintain/

[9] D.Mata, OWASP Insecure Deserialization with Python, June 2020, Online [Available]: https://davidmatablog.wordpress.com/2020/06/07/owasp-insecure-deserialization-with-python/

[10] B.Yildrim, Kondukto, Insecure Deserialization, June 2022, Online [Available]: https://kondukto.io/blog/insecure-deserialization

[11] Python Docs, re, Online [Available]: https://docs.python.org/3/library/re.html
[12] Python Docs, os, Online [Available]: https://docs.python.org/3/library/os.html

[13] Veracode, State of Software Security Volume 11 Infosheet, 2023, Online [Available]: https://info.veracode.com/state-of-software-security-volume-11-flaw-frequency-by-language-infosheet-resource.html

[14] Veracode, State of Software Security Volume 11, 2023, Online [Available]: https://info.veracode.com/rs/790-ZKW-291/images/Veracode_State_of_Software_Security_2023.pdf

[15] Veracode, Security Flaw Heat Map, Online [Available]: https://www.veracode.com/sites/default/files/pdf/resources/ipapers/security-flaw-heatmap/index.html

[16] CWE-79, Mitre, [Online] Available: https://cwe.mitre.org/data/definitions/79.html#:~:text=CWE%20Glossary%20Definition-,CWE%2D79%3A%20Improper%20Neutralization%20of%20Input%20During%20Web%20Page%20Generation,('Cross%2Dsite%20Scripting')&text=The%20software%20does%20not%20neutralize,is%20served%20to%20other%20users.

[17] CWE-352, Mitre, [Online] Available: https://cwe.mitre.org/data/definitions/352.html

[18] CWE-113, Mitre, [Online] Available: https://cwe.mitre.org/data/definitions/113.html

[19] Kirsten S, OWASP, [Online] Available:

https://owasp.org/www-community/attacks/xss/#:~:text=Cross%2DSite%20Scripting%20(XSS),to%20a%20different%20end%20user.

[20] Web Security Store, July 2nd 2021, [Online] Available:
https://websitesecuritystore.com/blog/real-world-cross-site-scripting-examples/

[21] CherryPy, CherryPy Docs[Online] Available:
https://docs.cherrypy.dev/en/latest/pkg/cherrypy.lib.sessions.html#cherrypy.lib.sessions.Session.generate_id

[22] A. Greenberg, The Untold Story of NotPetya, the Most Devastating Cyberattack in History, Wired, Aug 2018, [Online] Available:
https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-codecrashed-the-world/

[23] S. Fayi, What Petya/NotPetya Ransomware Is and What Its Remediations Are, Jan 2018, [Online] Available:
https://www.researchgate.net/publication/324489505_What_PetyaNotPetya_Ransomware_Is_and_What_Its_Remidiations_Are

[24] D. Maynor, The MeDoc Connection, July 2017, [Online] Available:
https://blog.talosintelligence.com/the-medoc-connection/

[25] Digitz, Proftp Vulnerability could Allow an Attacker to Gain a Shell in Your Server,[Online] Available:
https://digitz.org/blog/proftp-vulnerability-could-allow-an-attacker-to-gain-a-shell-in-your-server/

[26] A.Cherapanov, WeLiveSecurity, Analysis of Telebots' cunning backdoor, July 2017, [Online] Available:
https://www.welivesecurity.com/2017/07/04/analysis-of-telebots-cunning-backdoor/

[27] C.Cimpanu, M.E.Doc Software Was Backdoored 3 Times, Servers Left Without Updates Since 2013, July 2017, [Online] Available:
https://www.bleepingcomputer.com/news/security/m-e-doc-software-was-backdoored-3-times-servers-left-without-updates-since-2013/

[28] L. Irwin, 20 Cyber Security Statistics for 2022, February 2022, [Online] Available:
https://www.itgovernance.eu/blog/en/20-cyber-security-statistics-for-2022#:~:text=1)%2043%25%20of%20cyber%20attacks,data%20breaches%20occur%20at%20SMEs

[29] V. Davies, Cyber., Software supply chain attacks tripled in 2021 says Argon, January 2022, [Online] Available:
https://cybermagazine.com/cyber-security/software-supply-chain-attacks-tripled-2021-says-argon

[30] G. Schram, Cybrary, June 2021, [Online] Available:
https://www.cybrary.it/blog/notpetya-its-consequences/

[31] N. Goud, Cybersecurity Insiders, [Online] Available:
https://www.cybersecurity-insiders.com/ukraines-accounting-software-firm-refuses-to-take-cyber-attack-blame/

[32] Linkos Group, [Online] Available: https://www.linkos.ua/