UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et génie informatique

RAPPORT

Programmation sécurisée
GEI-771

Présenté à
Guy Lépine

Présenté par
Brian Compagnat – comb2301
Joel Perron-Langlois – perj2324

Sherbrooke – 19 septembre 2018

Table des matières

1 MODÉLISATION DE LA MENACE

1.1 IMPLIQUANTS DU LOGICIEL DE SONDAGE

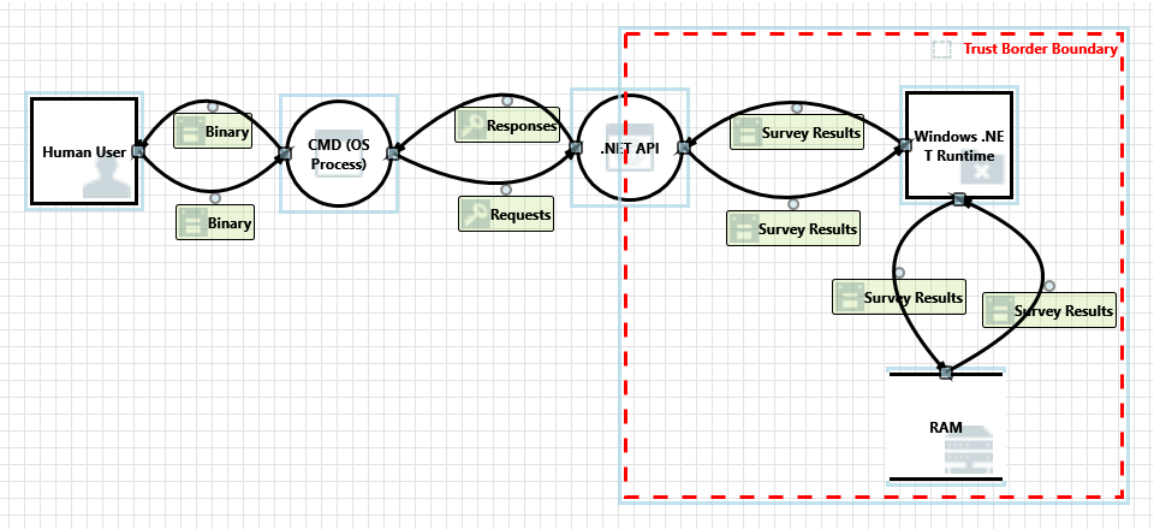


Figure 1 – Schéma de l'utilisation de l'application client utilisant un API pour communiquer un sondage au serveur

L'APP constitue en une situation où l'on doit faire le développement d'un logiciel de sondage. Afin de répondre au sondage, l'utilisateur doit utiliser une application console utilisant l'API du logiciel de sondage. Une clé d'API doit être utilisé à chaque appel vers le serveur, sans quoi la communication à ce dernier sera refusée. En plus de la clé d'API, il faut initialement s'authentifier au serveur avec un nom d'utilisateur et un mot de passe. Cette authentification ouvre une session de 30 minutes. À la suite de cette authentification, l'utilisateur choisi l'un des sondages auquel il veut répondre. Ensuite, il répond question par question au sondage par des caractères de A à D. Lorsque ce sondage est terminé, l'application lui demande s'il veut effectuer un autre sondage disponible.

De son côté, le serveur n'accepte seulement les connexions contenant une clé d'API valide et dont l'utilisateur s'est authentifié. Par ailleurs, seuls les communications HTTPS utilisant TLS sont acceptées et si on essaie d'utiliser HTTP, on est redirigé vers le protocole HTTPS. Le serveur ne peut être communiqué qu'avec trois (3) fonctions, soient connect(), GetAvailablePolls() et GetNext(). Tous d'entre eux utilise un mécanisme de sécurité, un nom d'utilisateur et un mot de

passe pour connect() et un identifiant d'utilisateur dont la session est ouverte pour GetAvailablePolls() et GetNext(). Les requêtes HTTP sont déjà sauvegardé par défaut, mais les *Exceptions* survenues dans le code sont aussi sauvegardées.

1.2 ANALYSE D'IMPACT DE LA SÉCURITÉ ET LES VECTEURS D'ATTAQUES

Bien que le logiciel de sondage ne contienne des renseignements confidentiels d'utilisateurs, leurs données restent personnelles et doivent être conservé en toute sécurité. Le rapport en annexe présente des vecteurs de sécurité que l'application pourrait contenir et dont des attaquants pourraient utiliser afin d'accéder à des zones interdites du logiciel.

Voir **5 Annexe – Rapport des vecteurs d'attaques**

2 PLAN DE TESTS

Le tableau suivant présente le plan de tests de l'application client et du serveur. Tous les tests possibles non pas tous été inscrits dans le tableau.

Table 1 – Plan de test des applications développées

	Client		**Serveur**	
Description du test	**Réponse attendue**	**Réponse obtenue**	**Réponse attendue**	**Réponse obtenue**
Communication HTTPS	Oui	(whireshark windows)	Oui	(whireshark windows)
Communication HTTP	Non	Non	Redirection HTTPS	Redirection HTTPS
Connect avec bon creds	-	-	Retourne userid	Retourne userid
Connect avec mauvais username	Please enter username	Please enter username	Unauthorized (http)	Unauthorized (http)
Connect avec mauvais password	Please enter password	Please enter password	Unauthorized (http)	Unauthorized (http)
Connect mauvais creds (apres 3 fois)	Exit	Exit	Unauthorized (http)	Unauthorized (http)

Connect avec objet creds null	-	-	BadRequest (http)	BadRequest (http)
Connect avec rien	-	-	BadRequest (http)	BadRequest (http)
Fait requete apres 30 min de connect()	-	-	Deconnection, Unauthorized	Deconnection, Unauthorized
Connect sans API key	-	-	Unauthorized (http)	Unauthorized (http)
GetAvailablePolls avec bon UserId	(gerer dans le code)	(gerer dans le code)	Retourne la liste de sondages	Retourne la liste de sondages
GetAvailablePolls avec mauvais UserId	(gerer dans le code)	(gerer dans le code)	BadRequest (http)	BadRequest (http)
GetAvailablePolls sans UserId	(gerer dans le code)	(gerer dans le code)	BadRequest (http)	BadRequest (http)
Connect sans API key	-	-	Unauthorized (http)	Unauthorized (http)
GetNext avec bon userId et answer	Fait la requete	Fait la requete	Retourne un PollQuestion	Retourne un PollQuestion
GetNext mauvais userId et bon answer	-	-	BadRequest (http)	BadRequest (http)
GetNext bon userId et bon mauvais answer	Redemande la question	Redemande la question	BadRequest (http)	BadRequest (http)

3 PROCESSUS D'IDENTIFICATION ET DE PUBLICATION DES BOGUES DE SÉCURITÉ

Certes, des vulnérabilités non-prévues et de nouvelles découvertes feront en sorte que notre application contient des failles de sécurité. Afin de réagir efficacement face à ces failles il est nécessaire d'avoir un processus mis en place pour gérer celles-ci. La gestion d'un tel incident fait partie du fameux « Security Development Lifecycle » (SDL).

Afin d'identifier ces bogues de sécurité, il serait bon d'avoir un adresse courriel dédié à recevoir ces bogues de sécurité qui provient des utilisateurs. Par ailleurs, l'utilisation d'un pipeline CI/CD (ex : Jenkins) est très intéressant puisqu'il permettrait de tester le bon fonctionnement et l'exemption de bogues évidents du logiciel de sondage à chaque nouvelle itération. L'identification de bogues peut aussi être fait en suivi les médias de cybersécurité et étant à jour

sur les nouvelles vagues d'attaques. Tous ces chemins d'identification de bogues de sécurité permettent de prévenir d'attaques futures.

Chaque menace n'a pas le même niveau d'importance et ces menaces ne sont pas obligatoirement être gérer dans l'immédiat. Des organisations bien reconnus, tel que le NIST, proposent des plans de réponse. Celui qui sera présenté s'inspire de celui publié par le NIST.

Les étapes à suivre sont celles-ci :

A. Compréhension de l'incident

1. Reçoit et analyse de l'incident de cybersécurité
2. Déterminer le niveau et l'impact de l'incident
    a. Faible : N'empêche pas le bon fonctionnement de l'application
    b. Modéré : Peut avoir un impact sur le bon fonctionnement de l'application
    c. Élevé : Démontre un impact sur l'application et sur les données
    d. Critique : Impacte aussi d'autres applications/services
    e. Urgence : Empêche la compagnie d'opéré
3. Formuler une approche et des actions à prendre pour régler cet incident
4. Regarder les logs de l'application et des outils de sécurité en place
5. Comprendre la motivation de l'attaquant
6. Identifier les traces de l'attaquants

B. Réaction face à l'incident

1. Selon le risque énuméré en A.2 déterminé si on doit mettre dans le backlog la réparation de l'incident, dans le prochain sprint, mobiliser les troupes immédiatement ou vivre avec le risque et si on doit interrompre des services
2. Réparer l'incident
3. Valider que l'incident ne puisse se reproduire

C. Retour sur l'incident

1. Utiliser les leçons de l'évènement pour améliorer le plan de réponse

4 RECOMMANDATION DU SYSTÈME D'EXPLOITATION

L'un des points intéressants de l'utilisation d'une application en .Net Core est le fait qu'elle peut être roulé sur différents OS (Windows, Linux, macOS). Il est difficile de croire qu'il y a un grand

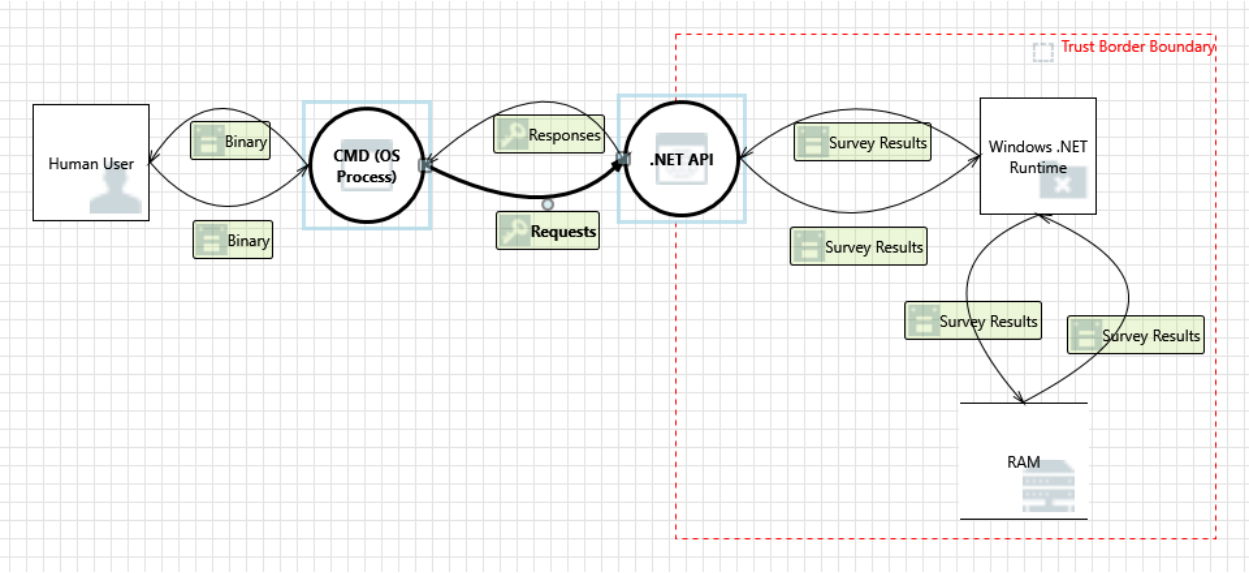
vainqueur au niveau de la sécurité des serveurs. On recommande d'utiliser le système d'exploitation dont l'administrateur est le plus à l'aise puisqu'il sera en mesure de mettre en place les configurations nécessaires, de mettre et garder à jour le serveur ainsi que d'avoir un meilleur contrôle d'accès aux données contenues dans le serveur. Bref, le choix du système d'exploitation n'a pas vraiment d'importance en autant que le serveur soit bien configuré et à jour.

5 ANNEXE – RAPPORT DES VECTEURS D'ATTAQUES

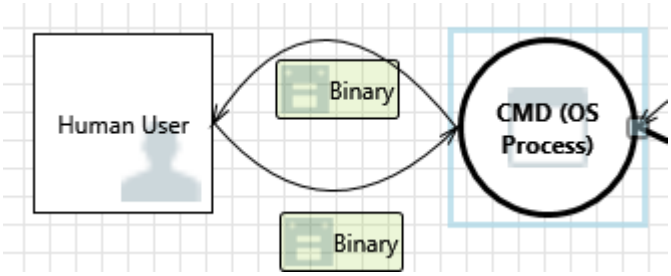
Threat Model Summary:

Not Started	0
Not Applicable	2
Needs Investigation	0
Mitigation Implemented	13
Total	15
Total Migrated	0

Diagram: Diagram 1



Interaction: Binary



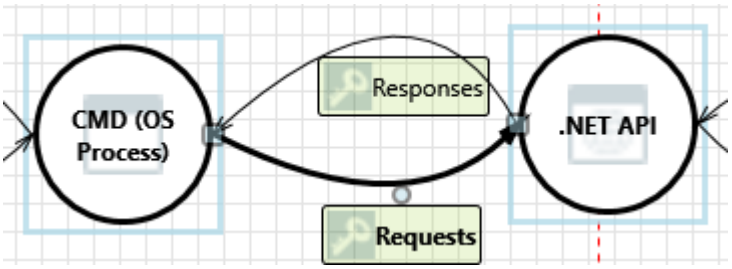
1. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: CMD (OS Process) may be able to impersonate the context of Human User in order to gain additional privilege.

Justification: La sécurité de la console du client ne compromet pas le côté du serveur, où sont stocké les données. Aucune donnée importante n'est contenue du côté client. On ne compromet pas le client, bien que l'obfuscation de l'application cliente rend la vie plus dur aux attaquants.

Interaction: Requests



2. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege

Description: .NET API may be able to impersonate the context of CMD (OS Process) in order to gain additional privilege.

Justification: Un système authentification par clé d'API et par mot de passe est en place. Ainsi, si l'API .NET essaie d'imiter la console du client, elle doit tout de même avoir possession de la clé d'API et d'avoir le userID afin de changer les réponses du client associé à cet userID.

3. JavaScript Object Notation Processing [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: If a dataflow contains JSON, JSON processing and hijacking threats may be exploited.

Justification: Les objets JSONs sont vérifiés du côté serveur ainsi que tous les attributs du JSONs sont validé afin qu'ils aient un sens à la réponse attendue.

4. Replay Attacks [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.

Justification: Cette faille est gérée avec TLS1.2

5. Collision Attacks [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.

Justification: Utilisation de TLS1.2

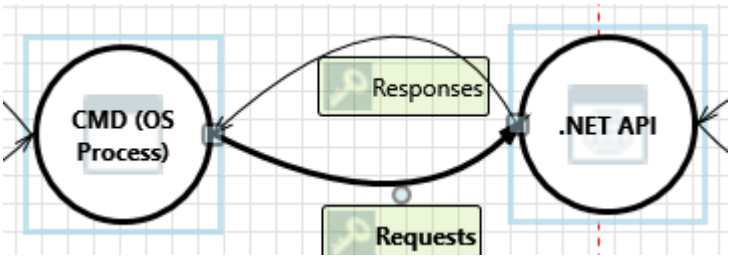
6. Weak Authentication Scheme [State: Mitigation Implemented (Need better password encryption)] [Priority: High]

Category: Information Disclosure

Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.

Justification: Nous avons implémenté un système de clé API, ainsi que d'authentification par nom d'utilisateur - mot de passe. Nous acceptions seulement TLS1.2 et l'authentification par dégradation n'est pas possible. Cependant, dans le cas présent le mot de passe est hasher en Base64 dans le système de stockage. Il est évident que ce dernier devra être encrypter avec un algorithme beaucoup plus sécuritaire dans le futur (SHA-512). Cependant, puisque la communication est HTTPS, un attaquant ne verra pas le mot de passe en clair dans les paquets.

Interaction: Responses



7. Collision Attacks [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Attackers who can send a series of packets or messages may be able to overlap data. For example, packet 1 may be 100 bytes starting at offset 0. Packet 2 may be 100 bytes starting at offset 25. Packet 2 will overwrite 75 bytes of packet 1. Ensure you reassemble data before filtering it, and ensure you explicitly handle these sorts of cases.

Justification: Utilisation du protocole TLS1.2

8. Replay Attacks [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Packets or messages without sequence numbers or timestamps can be captured and replayed in a wide variety of ways. Implement or utilize an existing communication protocol that supports anti-replay techniques (investigate sequence numbers before timers) and strong integrity.

Justification: Utilisation du protocole TLS1.2

9. JavaScript Object Notation Processing [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: If a dataflow contains JSON, JSON processing and hijacking threats may be exploited.

Justification: Les objets JSONs sont traités par le serveur avant d'être envoyés et les informations sont limités.

10. Elevation Using Impersonation [State: Not Applicable] [Priority: High]

Category: Elevation Of Privilege

Description: CMD (OS Process) may be able to impersonate the context of .NET API in order to gain additional privilege.

Justification: La sécurité de la console du client n'impacte pas directement le serveur, où sont stocké les données. On ne compromet pas le client, bien que l'obfuscation de l'application cliente rend la vie plus dur aux attaquants.

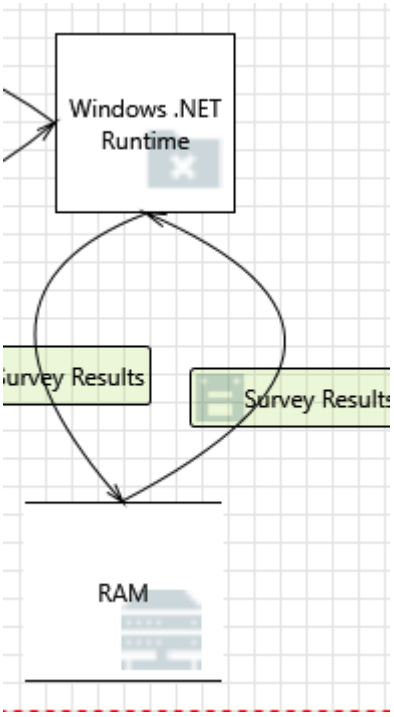
11. Weak Authentication Scheme [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Custom authentication schemes are susceptible to common weaknesses such as weak credential change management, credential equivalence, easily guessable credentials, null credentials, downgrade authentication or a weak credential change management system. Consider the impact and potential mitigations for your custom authentication scheme.

Justification: Nous avons implémenté un système de clé d'API ainsi que de mot de passe. Nous devons sécuriser le stockage de ses données sensibles.

Interaction: Survey Results



12. Weak Access Control for a Resource [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of RAM can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: Cette partie se trouve dans notre barrière de confiance, mais en effet, si notre serveur est compromis, les données de la RAM le seront aussi. L'implémentation d'une base de donnée communiquant uniquement avec le serveur de façon sécurisé ajouterait une barrière de sécurité et un étape additionnel pour les attaquants. Cependant, sous Windows pour accéder à la RAM il faut un accès au niveau du Kernel, donc si simplement un service est compromis, les données de la RAM ne sont pas forcément compromises.

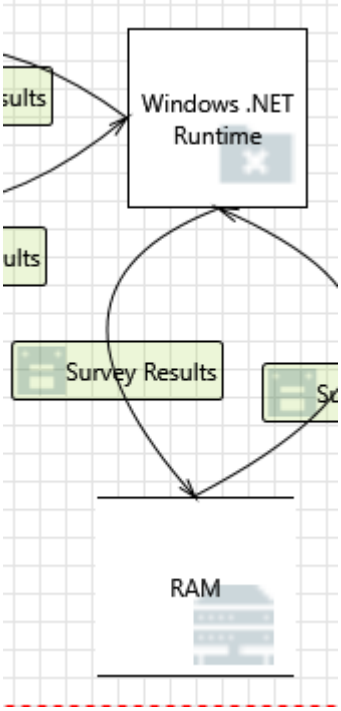
13. Spoofing of Source Data Store RAM [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: RAM may be spoofed by an attacker and this may lead to incorrect data delivered to Windows .NET Runtime. Consider using a standard authentication mechanism to identify the source data store.

Justification: Cette partie se trouve dans notre zone de confiance et il est dur d'imaginer quelqu'un qui serait en mesure de spoofer la RAM. Si elle a accès au serveur avec des privilèges élevés, les données seront compromises.

Interaction: Survey Results



14. Spoofing of Destination Data Store RAM [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: RAM may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of RAM. Consider using a standard authentication mechanism to identify the destination data store.

Justification: Cette partie se trouve dans notre zone de confiance.

Interaction: Survey Results

15. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege

Description: .NET API may be able to impersonate the context of Windows .NET Runtime in order to gain additional privilege.

Justification: Cette partie se trouve dans notre zone de confiance.