COMP 4920 Senior Design Project II, Spring 2019
Proje Özeti Formu

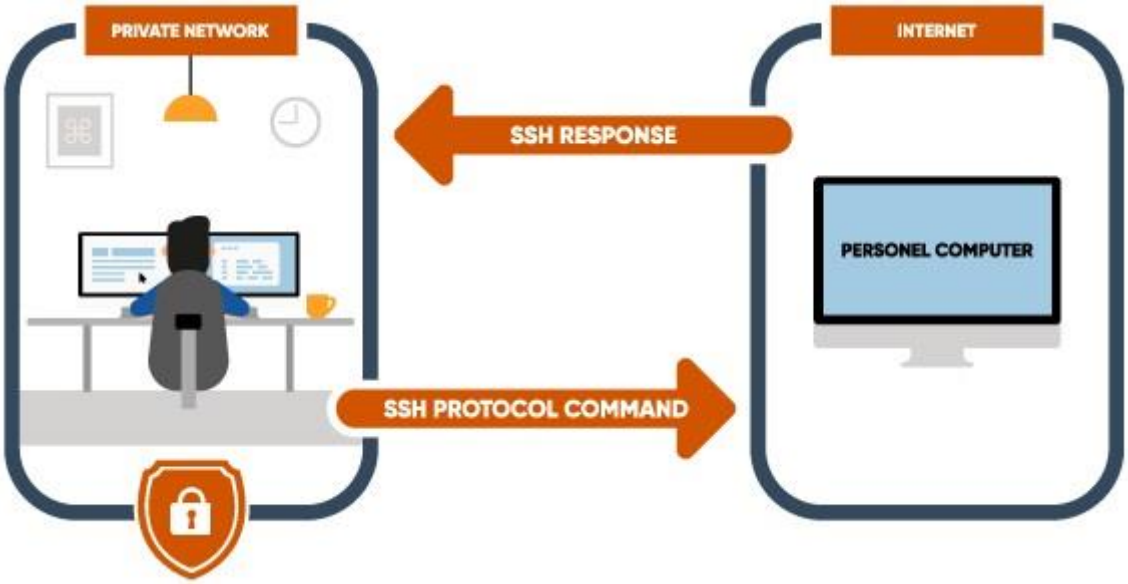
Proje Kodu ve Adı:	**METIN** – **M**essage **E**ncapsulation over **T**CP/**I**P **N**etwork
Öğrenciler: Ad, Soyad ve E-posta:	Mert ÖZGEN – ozgenmert96@gmail.com Mert PEHLİVANCIK – m.pehlivancik@gmail.com Taylan AKKUŞ – taylan.akkus@outlook.com
Proje Danışmanı Adı, Soyadı:	Hüseyin HIŞIL
Proje Çıktıları:	Requirements Specifications Document Design Specifications Document Product Manual
Proje Web Adresi (varsa):	https://github.com/metin35
Proje Özeti:	

Giriş

Günümüzde birçok kuruluş kendi bünyesinde kullandıkları ağ trafiğini, güvenlik duvarlarıyla korumaktadır. Kurum içerisinden veya dışarısından gelebilecek zararlı ağ trafiğine karşın özelleştirilmiş olan güvenlik duvarları; sistem yöneticileri tarafından kurum için gerekli görülmeyen portları engellerler. Ayrıca, açık olan portlardan gelen ağ trafiğini denetlemek de güvenlik duvarlarının işlevselliği arasındadır.

Bu durumda, kurum içerisindeki ağ kullanıcıları, bazı yazılım ve protokolleri kullanamazlar. Secure Shell (SSH) protokolüyle kurum dışındaki kişisel bilgisayarının terminaline erişmek isteyen kullanıcıların ağ paketleri güvenlik duvarı tarafından engellenir ve kullanıcılar ihtiyaç halinde, kişisel bilgisayarında bulunan dosyalara erişemezler.

Bu doküman üstte bahsedilen sorunu çözmek için geliştirilmiş Message Encapsulation Over TCP/IP Network (METIN) projesini özetlemek için hazırlanmıştır.



Şekil 1-Metin Sistemine Genel Bakış

Yöntem

Sorunun çözüm yollarını üretebilmek için öncelikle kurumlarca kullanılan ağ mimarileri ve ağı korumak için kullanılan güvenlik duvarlarının nasıl çalıştığı araştırıldı. Yapılan inceleme ve araştırma neticesinde, çoğu kuruluşun ağ kullanıcılarının web sitelerine ulaşabilmelerini sağlamak için Hypertext Transfer protokolünü (HTTP) kullanımı amaçlı, güvenlik duvarlarınca 80 portuna izin verildiği anlaşılmıştır.

Edindiğimiz bilgi doğrultusunda geliştirilen çözüm fikri; 80 portu üzerinden HTTP ağ paketlerinin içeresine SSH protokolü paketlerini yerleştirmek ve bu HTTP paketlerini çözümleyerek içerisinden SSH komutlarını alıp, doğru adrese yönlendirdikten sonra gelen cevabı geriye HTTP cevabı olarak döndürebilecek bir sistem olmuştur.

Kullanıcı kurum içerisindeki bilgisayarında bulunan kullanıcı ara yüzüne, uzaktan erişmek istediği bilgisayarın IP adresini, kullanıcı adı ve şifresini girer. Girdiği bilgiler internette bulunan web sunucusuna HTTP isteği olarak gönderildikten sonra sunucu isteğin içindeki giriş bilgileriyle hedef bilgisayarla SSH oturumu başlatır. Başarılı giriş işleminden sonra kullanıcı SSH komutlarını gönderebildiği ilgili ara yüzden

göndermeye başlar. Tüm giden komutlar HTTP isteklerinin içerisine yerleştirildikten sonra hedef bilgisayara aktarılır. SSH serverdan gönderilen cevaplar web sunucu tarafından alınır ve HTTP cevabı olarak kullanıcıya geri gönderilir. Kullanıcı programı gelen cevap içerisindeki cevapları ara yüze yansıtır. Bu sayede, normal şartlarda erişmesi mümkün olmadığı iç ağ dışındaki bilgisayarına erişmiş olur.

Veri güvenliği için gönderilen bilgiler ve gelen cevaplar şifrelenerek ağ trafiğine gönderilmeli gelen şifreler istemci ve sunucu tarafından deşifre edildikten sonra işleme sokulmalıdır.

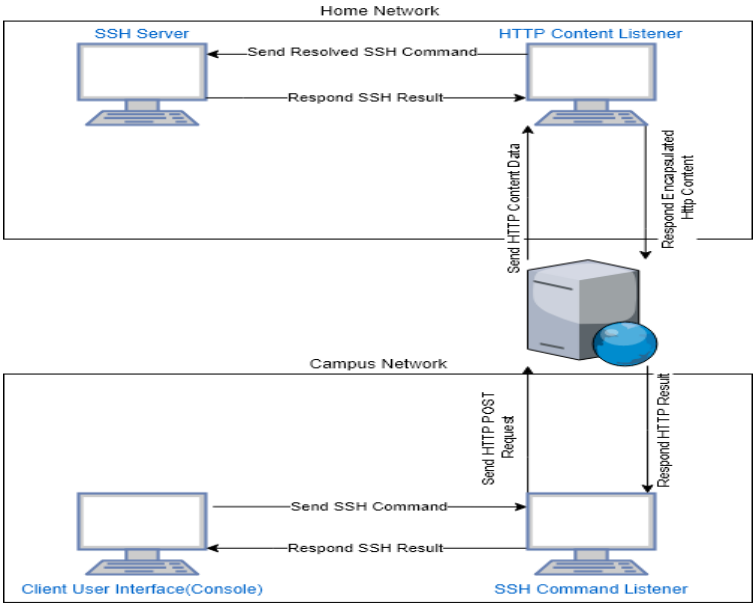
Sistem Simülasyon Ortamı

Sistemin çalışacağı koşulları denemek için incelenen kuruluşların ağ yapılarına benzer bir yapı kurulması gerekmektedir. Bunun için Oracle Virtualbox sanallaştırma yazılımını kullanarak; istemci, güvenlik duvarı ve web sunucu görevlerini üstlenecek, Linux dağıtımı olan Ubuntu 16.04 işletim sistemli üç ayrı sanal makine oluşturulmuştur.

Sanal ağ kartları, ağ trafiğinin güvenlik duvarı üzerinden olacak şekilde ayarlandı. İlk olarak, sistem hiçbir ağ duvarı kuralı olmadan çalıştırılıp ağ paketlerinin doğru şekilde istemci-güvenlik duvarı-sunucu ve tersi yönde gidebildiği gözlemlenip doğrulandı. Akabinde, güvenlik duvarı makinesine iptables kuralları ağ trafiğini sadece HTTP trafiğine izin verecek şekilde ayarlandı. Bunun doğrultusunda 80 portu dışında diğer bütün portlara gelen paketlerin engellenip HTTP paketlerinin simülasyon ortamında düzgün şekilde iletildiği gözlemlendi.

Yazılım Sistem Mimarisi

Yazılım sistem mimarisi üç ayrı bilgisayar üzerinde çalışan üç ayrı sürecin iletişim kurmasıyla gerçekleşir. Bunlardan birisi hazırda bulunan OpenSSH-Server yazılımıdır. Proje kapsamında sıfırdan yapılacak olan Metin-Client ve Metin-Server yazılımları da sistemde yer alacak diğer iki süreçtir.
Şekil 2. De sistemin genel tasarımı gösterilmektedir. Metin-Client güvenlik duvarı aktif olan sadece 80 portunun izinli olduğu kurum ağında yer alır. Kullanıcıdan aldığı girdileri HTTP isteği olarak Metin-Server sunucusuna yollar ve sunucudan gelen HTTP cevaplarını kullanıcıya uygun bir şekilde gösterir. Metin-Server ise gelen isteklerdeki SSH komutlarını çözümledikten sonra OPENSSH-Server yazılımına gönderip cevap bekler. Gelen cevap HTTP cevap paketlerinin içerisine veri olarak yerleştirilir ve isteğine cevap beklemekte olan Metin-Client'a gönderilir.



Şekil 2-Metin Sistem Mimarisi

METIN-Server

Metin-Server, C programlama dili ile geliştirilmiştir. Metin-Server'ın içinde HTTP Web Server ve SSH Client eş zamanlı olarak çalışmaktadır. SSH-Client, libssh kütüphanesi kullanılarak yazılmıştır. Akışın önemli kısımları güvenliği arttırmak amacıyla, AES-256 kullanılarak şifrelenmiştir. Şimdi kodun önemli bölümleri ve programın çalışma mantığı kod üzerindeki önemli fonksiyonlar incelenerek adım adım anlatılacaktır.
Aşağıda örneği verilen ilk kod bloğu "libssh.c" isimli dosya içine kaydedilmiştir.

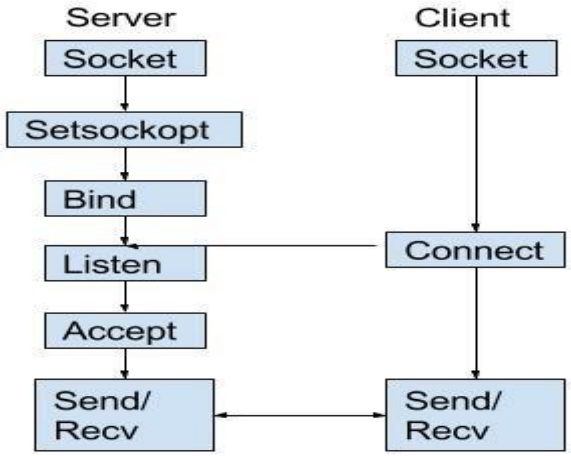
```
1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <libssh/libssh.h>
4. #include <unistd.h>
5. #include <sys/types.h>
6. #include <sys/socket.h>
7. #include <netinet/in.h>
8. #include <netdb.h>
9. #include <arpa/inet.h>
10.  #include <string.h>
11.  #include <sys/stat.h>
12.  #include <fcntl.h>
13.  #include "free_channel.h"
14.  #include "free_session.h"
15.  #include "error.h"
16.  #include "shell_session.h"
17.  #include "new_session.h"
18.  #include "execute_command.h"
19.  #include "interactive_shell_session.h"
20.  #include "setHTTPHeader.h"
21.  #include "report.h"
22.  #include "server.h"
23.  #include "ConnectServerAndSSHSession.h"
24.  #include "recursive.h"
25.  #define SIZE 1024
26.
27.  int main(int argc, char** argv) {
28.
29.                int server_port = 0;
30.                char IP[15];
31.                memset(IP, 0, 15);
32.                printf("Server IP Address: ");
33.                scanf("%[^\n]%*c", &IP);
34.                printf("Server Port: ");
35.                scanf("%d", &server_port);
36.                recursive(server_port, IP);
37.                 return 0;
```

Yukarıdaki kod bloğu main kod bloğu olarak adlandırılır ve program main kod bloğundan başlar. Gerekli olan kütüphaneler ve başlık dosyaları genellikle burada programa eklenir. Main kod bloğunda "server_port" ve "IP" olmak üzere iki adet değişken tanımlanmıştır. Bu değişkenlerin içindeki hafıza tarafından atanan değerler, ön değer atama yapılarak hafızadan temizlenir. Bu işlem programın çalışması için büyük önem arz etmektedir. Çünkü hafızada kalan değerler sunucunun istenilen şekilde çalışmasını engeller. Örneğin; sunucu "192.168.1.85" yerel IP adresinde başlatılmak isteniyor fakat hafızada kalan hafıza tarafından değişkene ilk olarak atanan değerler, istenen IP adresin dışında bir adreste sunucuyu başlatmaya zorlayabilir. Bu sebeple değişkenlerin ön değer atama yoluyla temizlenmesi gereklidir. IP ve Port değerleri kullanıcıdan alınır ve alınan bu değerler HTTP Web Server başlatılırken kullanılır. Bu sayede kullanıcı isteği IP de ve portta sunucuyu başlatabilir.

Aşağıdaki kod bloğu “recursive.h” isimli dosyaya kaydedilmiştir.

```
1. #ifndef RECURSIVE_H_
2. #define RECURSIVE_H_
3. #include "shell_session.h"
4. int recursive(int server_port, char* IP) {
5.                 server(server_port, IP);
6.                 ConnectServerAndSSHSession();
7.                 int rv = 0;
8.                 rv = new_session(username, password, hostname, server_port, IP);
9.
10.                 if (shell_session(session, server_port, IP) != SSH_OK) {
11.                                 error(session, server_port, IP);
12.                                 free_channel(channel);
13.                                 return rc;
14.                 }
15.                 interactive_shell_session(session, channel, server_port, IP);
16.  }
17.
18.
19.  #endif /* RECURSIVE_H_ */
```

Bu başlık dosyası “recursive()” isimli bir fonksiyona sahiptir. Bu fonksiyon içinde ilk olarak “server.h” başlık dosyasının içinde bulunan “server()” fonksiyonu çağırılır. Sunucunun çalışma prensibi aşağıdaki durum diyagramında gösterilmiştir:



Şekil 3-Sunucu ve İstemci için Durum Diyagramı

İlk olarak durum diyagramında da görüldüğü üzere TCP soket oluşturulur ve bind() fonksiyonu ile soket sunucu adresine bağlanır. Sunucu “listen()” fonksiyonu ile istemciden gelecek olan bağlantı isteklerini karşılar. Eğer sunucu ve istemci arasında bağlantı kurulursa, onlar arasında “accept()” fonksiyonu kullanılarak veri aktarımı gerçekleşebilir. Bu aşamaların hepsi gerçekleştikten sonra HTTP Server istemci ile bağlantı kurmak için hazır hale gelir. “server()” fonksiyonunun sona ermesiyle beraber “recursive()” fonksiyonun içindeki “ConnectServerAndSSHSession()” fonksiyonu başlar. Burada istemciden sunucuya gelen istek, istemci tarafından güvenliği arttırmak amacıyla AES-256 ile şifrelendiği için, gelen isteğin şifresinin çözülmesi gerekmektedir. Önceden paylaşımı yapılmış anahtar yoluyla şifreli isteğin şifresi çözülür ve fonksiyonun dışında “ConnectServerAndSSHSession.h” ‘ın içinde tanımlanmış olan “username”, “password” ve “hostname” adlı genel değişkenlere atanır. Bu değişkenlerin genel değişken olarak tanımlanmasıyla beraber diğer başlık dosyalarından da erişimleri sağlanır. Bu fonksiyonun bitmesiyle beraber “new_session.h” içinde bulunan “new_session()” fonksiyonu çağırılır. Bu fonksiyonda SSH bağlantısı için gerekli olan SSH oturumu açılır ve fonksiyon bittikten sonra “shell_session.h” ‘ın içinde bulunan “shell_session()” fonksiyonu çağırılır. Bu fonksiyon açılan oturumu kullanarak SSH bağlantısı için kanal açar. “recursive()” fonksiyonunda çağırılan son fonksiyon “interactive_shell_session()” ‘dır. Bu fonksiyon ile önceden oluşturulmuş olan kanal yardımıyla SSH-Server ile veri alışverişi gerçekleştirilir. SSH-Client’a SSH-Server’dan aktarılan veriler HTTP-Server

üzerinden HTTP-Client'a gönderilir. Kullanıcı HTTP-Client yardımıyla SSH-Server'a göndermek istediği komutu istek olarak HTTP-Servar'a gönderir. Sonrasında HTTP-Server komutu alıp SSH-Client'a gönderir. SSH-Client'da komutu SSH-Server üzerinde çalıştırıp çıktısını aynı adımları takip ederek geriye yollar.

METIN-Client

Metin-Client, HTTP Client'ın özelleştirilmiş halidir. METIN-Client'ın amacı; METIN-Server ile bağlantı kurduktan sonra, METIN-Server'a SSH-Server'ın üzerinde çalışacak olan komutları gönderip, bu komutların çıktılarını METIN-Server'dan alıp programa çıktı olarak basmaktır. METIN-Server ile bağlantının sağlanabilmesi için kullanıcıdan bağlanmak istedikleri sunucuya ait IP ve sunucunun dinleme yaptığı port numarası istenir. Kullanıcıdan alınan değerler "HTTPClient.c" içindeki "ConnectHTTPServer()" isimli fonksiyona iletilir ve bu fonksiyon vasıtasıyla METIN-Server'ın içinde çalışan HTTP-Server ile bağlantı sağlanır. "HTTPClient.c" içerisinde bulunan "CreateSSHSession()" isimli fonksiyon çağırılır ve kullanıcıdan METIN-Server'ın SSH-Server'a bağlantı yapabilmesi için gerekli olan kullanıcı adı, şifre ve SSH-Server'ın IP adresi alındıktan sonra güvenlik amacıyla AES-256 ile şifrelenir. Şifrelenmiş olan veri istek olarak METIN-Server'a "SendHTTPRequest()" fonksiyonuyla gönderilirken, METIN-Server'dan alınan veriler "ReadResponse()" fonksiyonuyla alınır.

Sonuç

Yapılan araştırma, inceleme ve çalışmalar sonucunda; sadece HTTP ağ trafiğinin izin verildiği güvenlik duvarı kullanan bir kurumun ağından normal koşullarda erişilemeyecek bir bilgisayara SSH komutları gönderilip geriye beklenen cevaplar alınmıştır. Çözümlenmesi hedeflenen sorun, Message Encapsulation over TCP/IP Networks (METIN) projesiyle çözümlenmiş oldu. METIN Projesi sonucunda ortaya çıkmış olan yazılımı kullanan bir kullanıcı; kullanıcı adını, şifresini ve IP adresini bildiği bir bilgisayara SSH oturumu açabilir ve talep ettiği dosyalara erişebilir.