שם המגיש: עומר גרייף

ת"ז : 2082000154

תאריך הגשה: 04/07/23

שאלה 1 – IPSEC :

1. התרחיש שתואר אפשרי במידה וb מתפקד כgetway tunnel_mode בנוסף להיותו מחשב קצה. כלומר הפקטה המקורית נעטפת בשכבה של IPsec וIP לsocket של הipb עם gwA. כלומר הפקטה שתצא מb תתעטף אצלו ותעבור דרך gwB לgwA שם תבדק נכונות הIP ומשם הפקטה תעבור לa המבוקש
השורה המתאימה:

rule	direction	Src. adrs	Dst. adrs	protocol	Src. port	Dst. port	ack	action	Additional Parameters
Forwarding out secure b	out	IPb	IPgwA	IPSEC	ANY	ANY	ANY	forward	

2. תכני ה SAD אינם תקינים מכיוון שהפרוטוקולים המצוינים בטבלת SAD של כל אחד מהGW אינם מתאימים ועל כן הGW יזרוק את החבילות ולא יוכל לאמת ולוודא את תוכנן. כלומר, בעת שליחת פקטה כאשר היא תעבור בGWB היא תעבור הצפנה וחתימה כחלק מפרוטוקול ESP אבל כאשר היא תקלט בGWA היא תיזרק עקב אי תאימות לפרוטוקול שמצפה לקבל AH.

3. כעת תכני הSAD תקינים זאת מכיוון שהפקטה שנשלחת מקיימת את כל התנאים הבאים: ראשית הSPI היוצא אכן מתאים לSPI הנכנס, הפרוטוקולים זהים במוד תעלה ESP עם מפתח k1 זהה עבור שניהם ועל כן הפקטה תצליח לעבור את אימות הGW.
עם זאת מבחינת הצפנה הפרוטוקול אינו בטוח מכיוון שמופיע הערך null_siper שמהווה אינדיקציה לכך.
4. ראשית נבחין כי הטבלאות תקינות, תואמות ובעלי מפתחות הצפנה כך שהפרוטוקול יכול לפעול בהצלחה. עם זאת ישנו חסרון משמעותי בערכים הנתונים מכיוון שהמפתחות הצפנה והפענוח למפתחות נכנסים ויוצאים זהים. דבר שתוקף יכול לנצל למשל על ידי התקפת, תוקף יכול לתפוס פקטה בתעבורה מB לA למשל ולשנות לה את כתובת הIP החיצונית שאינה מוצפנת שתהא מGWA לGWB . הפקטה בסיכוי סביר תאושר מכיוון שהיא תענה על כל הקריטריונים (בהנחה ש sequnce_number מתקיים) ועל ידי כך לחדור את GWB עם פקטה שלא הייתה אמורה לחדור.
5. הכללים אינם עונים על הדרישות ראשית נבחין כי החבילות שמגיעות, מגיעות מוצפנות ESP מGWA בTRANSPOT MODE ועל כן לא ניתן לדעת מה הפורטים המקוריים כדי לתקן זאת, אם נשנה את פרוטוקול הכניסה לtunnel נקבל שהפקטה תנותח לפי המידע הפנימי ולא המעטפת ונוכל לאמת את הפורטים,IP והנתונים הפנימיים . בנוסף הפורט

צריך להיות גדול מ1023 ולא קטן. עבור פקטות שיוצאות בגלל שעוברים קודם בSPD וזה עוד לפני שינויים כלשהם במעטפת הפקטה הכל קשורה.

rule	direction	Src. adrs	Dst. adrs	protocol	Src. port	Dst. port	ack	action	Additional Parameters
http out	out	b	a	AH	<1023	80	ANY	secure	Tunnel Mode to GWA, ESP
http in	in	a	b	AH	80	<1023	ANY	secure	Tunnel Mode to GWA, ESP

<u>שאלה 2 - SSL</u>

1. הפרוטוקול מקיים את תכונת PFS. כלומר חשיפה של מפתח מסוים לא תוביל לגילו מפתח שהיה בשימוש. בפרוטוקול הנ"ל קיימת סיסמה ארוכת טווח שיא pass שמוסכמת על ידי המשתמש ובעזרתה הלקוח מעביר את החלק שלו במפתח DH להתקשרות כך שלאחר ה session של יצירת מפתחות DH אין שימוש בסיסמה pass בשיח הנוכחי והערכים שנקבעו לו לא מושפעים מערכי הpass. בנוסף המפתח DH שייווצר יאפשר שליחה של pre_master_secret שממנו נגזר המפתח של הsession. בסיום תהליך היצירה נהוג להשמיד את מפתחות DH כדי למנוע כל יכולת שחזור של מפתחות אלו ועל כן הם יהוו סיסמא טווח קצר ומתוקף כל אי התליות אכן מתקיים עקרון הPFS.

2. הפרוטוקול בטוח, מכיוון שהשרת והלקוח מסכימים על master_key שנשאר רלוונטי רק לsession הרלוונטי ומועבר באופן מוצפן בהתבסס על DH וpass. בנוסף גם סיסמא ארוכת הטווח מתבטאת בהצפנה מפתח DH של הלקוח לשרת, כלומר לא עוברת באופן גלוי ברשת כך שניסיונות התקפה אקטיביים לא יוכלו לקבל את התוכן או ליצור התחזות כלשהי.

3. הפרוטוקול עמיד בפני תקיפת מילון מתוקף פסיבי מכיוון שהצירופים היחידים שיכולים להיות משמעותיים עבור התוקף הם הצירופים המוצפנים של מפתח DH ושל pre_master_key . עם זאת הוא לא יכול לקבל אתpre_master_key מכיוון שהוא מוצפן על ידי DH שחלקו בעצמו מוצפן על ידי הpass. כל ניסיון פסיבי (על ידי האזנה בלבד) לפענוח חלק זה לא יישא פרי שכן באף שלב החלק המוצפן של מפתח DH לא עובר בצורה גלויה ועל כן לא ניתן לבדוק את הפענוח.

4. הפרוטוקול לא עמיד בפני תקיפת מילון אקטיבית. למשל התוקף יכול להתחזות לשרת ולשלוח לו ערכים מתאימים ובפרט עם חלק ממפתח DH שלו כך שהלקוח יחזיר לו את הודעות 3 ו4 שמכילות את המפתח המוצפן של החלק של הלקוח בDH וpre_master_secret מוצפן DH. כעת התוקף יכול לבצע התקפת מילון של ערכים ידועים עבור הסיסמא pass ועל ידי כך למצוא את g^ymodp שממנו ייגזר k שממנו יגזר pre_master_secret ושממנו ייגזר finish_client. כלומר מכיוון שאין לנו את g^ymodp התקפת מילון סטנדרטית לא תעזור כי אין לנו למי להשוות את הניסיונות אבל בהינתן הודעת finish שניתן לבדוק האם הערך שאנו מנחשים מגיע להתאמה עם ערך זה ועל כן לאפשר התחזות וגילו K.

5. כעת הפרוטוקול בטוח מכיוון שכפי שהסברתי בסעיף הקודם הסמכנו על ידיעת הfinish של הלקוח על מנת לאפשר בדיקת ניסיונות של התקפת המילון. כעת מכיוון שעלינו להעביר finish מהשרת בטרם קיבלנו הודעת finish מהלקוח לא נוכל להשוות את הנתונים ולהצליח להתחזות לשרת. בנוסף הכיוון ההפוך של התחזות ללקוח גם אינה אפשרית מכיוון שאין ברשותנו את הpass ועל כן אנו אמורים להעביר מפתח dh בתחילת ההתקשרות מוצפן בהתאם לpass.

<u>שאלה 3 – Wireless Security</u>

1. EAP-Request/Challenge:

AS to router

Challenge (מוצפן*)	APP – מוצפן ומאומת על ידי שכבה IPSEC
UDP Next Protocol -> Radius over EAP	TRANSPOT - מוצפן ומאומת על ידי שכבה IPSEC
TYPE: ESP Next protocol -> UDP	IPSEC
IP_{AS} to IP_{router} Next protocol -> IPSEC	IP
MAC_{AS} to MAC_{router}	PHYS+MAC

router to AP

Challenge(מוצפן*)	APP – מוצפן ומאומת על ידי שכבה IPSEC
UDP Next Protocol -> Radius over EAP	TRANSPOT - מוצפן ומאומת על ידי שכבה IPSEC
TYPE: ESP Next protocol -> UDP	IPSEC
IP_{AS} to IP_{AP} Next protocol -> IPSEC	IP
MAC_{router} to MAC_{AP}	PHYS+MAC

AP to laptop

Challenge	APP
UDP	TRANSPOT
IP_{AS} to IP_{AP} Next protocol -> UDP	IP
MAC_{AP} to MAC_{laptop}	PHYS+MAC

2. הודעת PMK :

AS to router

PMK (מוצפן)	APP – מוצפן ומאומת על ידי שכבה IPSEC
UDP Next Protocol -> Radius over EAP	TRANSPOT - מוצפן ומאומת על ידי שכבה IPSEC
TYPE: ESP Next protocol -> UDP	IPSEC
IP_{AS} to IP_{router} Next protocol -> IPSEC	IP
MAC_{AS} to MAC_{router}	PHYS+MAC

Router to AP

PMK (מוצפן)	APP – מוצפן ומאומת על ידי שכבה IPSEC
UDP Next Protocol -> Radius over EAP	TRANSPOT - מוצפן ומאומת על ידי שכבה IPSEC
TYPE: ESP Next protocol -> UDP	IPSEC
IP_{AS} to IP_{AP} Next protocol -> IPSEC	IP
MAC_{router} to MAC_{AP}	PHYS+MAC

3. בקשת HTTP שנשלחת כמתואר:

Laptop to AP

APP	TYPE: HTTP data
TRANSPOT	TYPE: TCP dest_port: 80 orig_port:<1024 Next protocol-> http
IP	IP_{laptop} to IP_{google} ack:yes Next protocol -> TCP
PHYS+MAC	MAC_{laptop} to MAC_{AP}

AP to router

APP	TYPE: HTTP data
TRANSPOT	TYPE: TCP dest_port: 80 orig_port:<1024 Next protocol-> http ack:yes
IP	IP_{laptop} to IP_{google} Next protocol -> TCP
PHYS+MAC	MAC_{AP} to MAC_{router}

Router to … to google

APP	TYPE: HTTP data
TRANSPOT	TYPE: TCP dest_port: 80 orig_port:<1024 Next protocol-> http ack:yes
IP	IP_{router} to IP_{google} Next protocol -> TCP
PHYS+MAC	MAC_{router} to MAC_{google} (change many MACS before get google)