HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Analiza Shatter napada

CCERT-PUBDOC-2005-12-143

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Sustavi upravljani događajima (eng. *event-driven system*), kao što je *Windows* grafičko korisničko sučelje, svoj rad temelje na jedinici informacije koja se naziva događaj (eng. *event*). Svaka korisnička interakcija s operacijskim sustavom (unos znakova s tipkovnice, korištenje miša i sl.), kao i zahtjevi samog operacijskog sustava (iscrtavanje prozora, gašenje aplikacije i sl.), predstavlja jedan događaj. Kod *Windows* obitelji operacijskih sustava, događaji se nazivaju porukama (eng. *window messages*).
Prozor pojedine aplikacije asociran je s odgovarajućom „window" procedurom, zaduženom za primanje i obradu poruka. Operacijski sustav većinu takvih procedura unaprijed definira, najčešće u obliku DLL (*Dynamic Link Library*) datoteka.
Ovaj dokument opisuje rad *Windows* sustava poruka**[1]**, analizira izvorni *Shatter* napad**[2]** te ocjenjuje trenutnu situaciju glede opisane ranjivosti.

2. Windows sustav poruka

Korisničko grafičko sučelje omogućava interakciju korisnika i aplikacija. Komunikacija se izvodi putem jednog ili više prozora koji pripadaju odgovarajućoj aplikaciji, a nalaze se na korisničkom grafičkom sučelju. Operacijski sustav prozorima šalje informacije u obliku poruka koje generira sami sustav (npr. interakcijom korisnika), aplikacija kojoj prozor pripada ili druga aplikacija. Poruka se šalje odgovarajućoj „window" procedura, a sadrži četiri parametra: identifikator prozora kojemu je poruka namijenjena, identifikator poruke koji određuje njenu namjenu te dva posebna parametra koji određuju podatke ili lokaciju podataka koji se koriste prilikom obrade poruke. Vrijednost i značenje tih parametara ovisi o vrsti poruke, tako da mogu sadržavati cjelobrojnu vrijednost, pokazivače na strukture i sl. U slučaju kada se parametar ne koristi, on ima `NULL` vrijednost. Općenito, poruke se prema vrsti mogu podijeliti u dvije glavne kategorije: one koje su definirane na razini operacijskog sustava i pojedinačne aplikacije.

2.1. Poruke definirane na razini operacijskog sustava

Poruke definirane na razini operacijskog sustava koriste se za komunikaciju između operacijskog sustava i aplikacije, pri čemu valja napomenuti kako i aplikacije mogu generirati sistemski definirane poruke. Kategorija kojoj određena poruka pripada definirana je simboličkom konstantom, pri čemu prefiks znakovnog niza određuje kojoj je vrsti prozora poruku namijenjena. Za analizu „Shatter" napada od posebne su važnosti prefiksi „EM" („Edit control") te „WM" („General window" kategorija), koji pokrivaju velik broj različitih poruka, između ostalog poruke o korisničkim interakcijama, podacima unesenim putem izbornika, stvaranju i upravljanju prozorima te DDE (*Dynamic Data Exchange*) poruke.

2.2. Poruke definirane na razini aplikacije

Aplikacije mogu stvarati poruke namijenjene komunikaciji s vlastitim prozorima ili s prozorima koji pripadaju drugim procesima (aplikacijama). Identifikatore poruka iz intervala `0x0000` do `0x03FF` operacijski sustav rezervira za sistemski definirane poruke, pa te vrijednosti aplikacije ne mogu koristiti za privatne poruke (poruke namijenjene vlastitim prozorima). Za privatne poruke koristi se interval od `0x0400` (identifikator `WM_USER` poruke) do `0x7FFF`. Ako aplikacija želi stvoriti novu poruku, pozvat će `RegisterWindowMessage` funkciju za registriranje poruke. Operacijski sustav će generirati jedinstveni identifikator poruke iz intervala `0xC00` do `0xFFFF`, koji osigurava da različite aplikacije ne koriste iste identifikatore poruka za različite svrhe.

2.3. Sustav usmjeravanja poruka

Prilikom usmjeravanja poruka Windows operacijski sustav koristi dvije metode. Prva se metoda temelji na slanju poruka u poseban red, koji se naziva red poruka (eng. *Message Queue*), a predstavlja memorijski objekt koji privremeno sprema poruke. Druga se metoda temelji na slanju poruka izravno odgovarajućim „window" procedurama.

Poruke koje se šalju u red nazivaju se poredanim porukama (eng. *queued messages*), a najčešće sadrže informaciju o interakcijama korisnika putem miša ili tipkovnice (npr. `WM_MOUSEMOVE`, `WM_LBUTTONDOWN`, `WM_KEYDOWN`, `WM_CHAR`). Operacijski sustav održava jedan jedinstveni red poruka te po jedan red poruka za svaku dretvu (eng. *thread*) kojoj odgovara jedan prozor grafičkog korisničkog sučelja. Tijekom interakcije korisnika, upravljački program (eng. *driver*) neke ulazno-izlazne jedinice generira poruke te ih smješta u globalni red poruka. Operacijski sustav potom uzima jednu po jednu poruku iz reda, provjerava odredište poruke te smješta istu u red poruka povezan uz odgovarajuću dretvu. Dretva potom skida poruku iz vlastitog reda te upućuje operacijski sustav na „window" proceduru zaduženu za njenu obradu. Aplikacija dohvaća poruke pozivom funkcije `GetMessage` dok se pregledavanje poruke bez skidanja iz reda izvodi `PeekMessage` funkcijom. Nakon skidanja poruke iz reda, aplikacija poziva funkciju `DispatchMessage` koja upućuje operacijski sustav na „window" proceduru zaduženu za obradu poruke.

Poruke koje se izravno šalju odgovarajućoj „window" proceduri nazivaju se neporedane poruke (eng. *nonqueued messages*) i važnije su za analizu ovog napada. Takve poruke zaobilaze globalni red poruka i red poruka pojedine dretve. Prilikom pokretanja aplikacije, operacijski sustav joj šalje

nekolicinu takvih poruka (`WM_ACTIVATE`, `WM_SETFOCUS`, `WM_SETCURSOR`) koje aplikaciju obavještavaju da ju je pokrenuo korisnik. Takve se poruke šalju i tijekom rada aplikacije, primjerice u slučaju da aplikacija koristi funkciju `SetWindowPos` za pomicanje prozora.

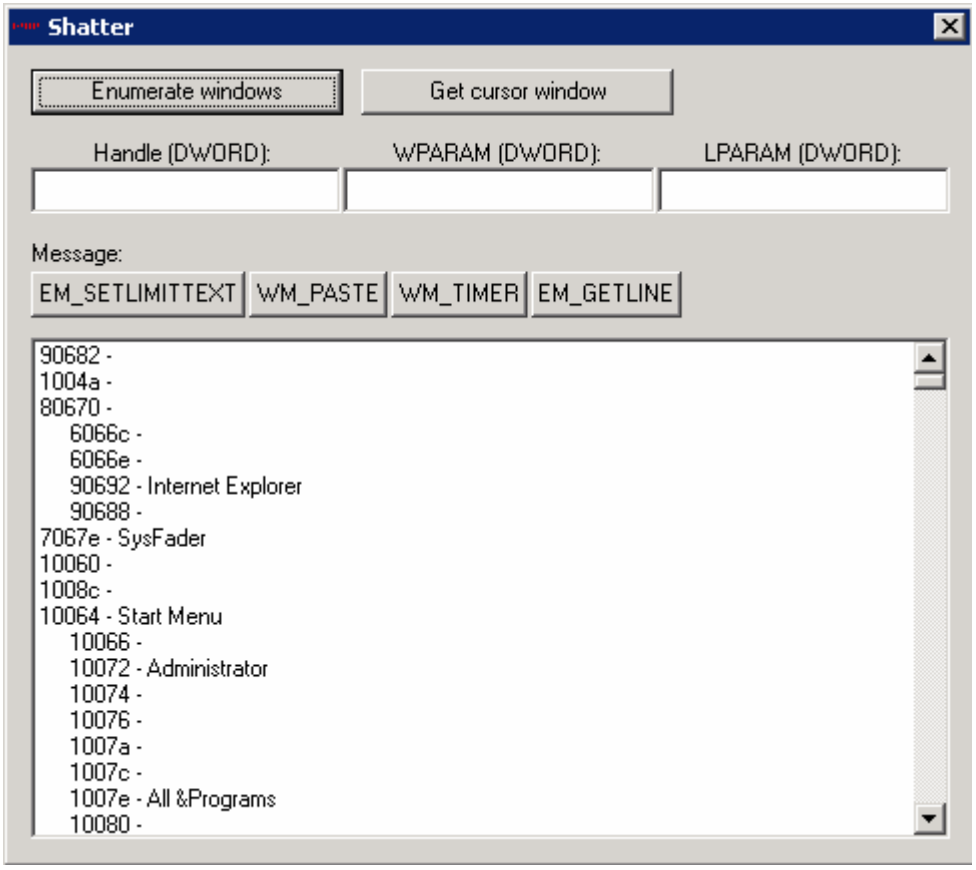
2.4. Nedostaci Windows sustava poruka

Windows sustav poruka razvijen je u vremenu kada svijest o računalnoj sigurnosti nije bila na današnjoj razini. Sama činjenica da bilo koji prozor u korisničkom grafičkom sučelju može poslati proizvoljnu poruku bilo kojem drugom prozoru predstavlja sigurnosni nedostatak. Nedostatak je još ozbiljniji u slučaju da na istom grafičkom sučelju postoje aplikacije koje su na različitim razinama ovlasti, uzevši u obzir da je njihova međusobna komunikacija omogućena. Iako je korisnik na sustav prijavljen s ograničenim ovlastima, postoji velika vjerojatnost da na istom korisničkom grafičkom sučelju postoji aplikacija koja je pokrenuta s najvišim mogućim ovlastima (primjerice antivirusni alat, koji mora pristupati i najosjetljivijim dijelovima operacijskog sustava odnosno memorijskog prostora).

3. Shatter napad

Shatter napad se zasniva na činjenici navedenoj u prethodnom poglavlju – mogućnosti komunikacije s bilo kojom aplikacijom koja se nalazi na korisničkom grafičkom sučelju. Izvorni se napad generalno može podijeliti u četiri osnovne faze: odabir odgovarajućeg prozora, uklanjanje određenih restrikcija, ubacivanje proizvoljnog programskog kôda te njegovo izvođenje. Prilikom analize napada, pretpostaviti će se da je napadač prijavljen na operacijski sustav kao *guest* korisnik te da na njegovom grafičkom korisničkom sučelju postoji aplikacija koja je pokrenuta uz *LocalSystem* ovlasti i posjeduje prozor koji sadrži *edit box* element za unos znakova. Napad je omguće izvesti i s udaljene lokacije, u slučaju da ranjivi Windows operacijski sustav koristi „Remote Desktop“ funkcionalnost te ima otvoreni *guest* korisnički račun putem kojeg se napadač prijavljuje na sustav.

U prvoj fazi napada, napadač odabire odgovarajući prozor koji sadrži kontrolu u koju je moguće upisati neke vrijednosti (bilo koja vrsta *edit* kontrole). Poznavanje naziva prozora koji u sebi sadrži navedenu kontrolu dovoljno je da se od Windows operacijskog sustava zatraži referenca na istu (eng. *handle*). Postupak je moguće obaviti koristeći alat (*Shatter*) koji je Chris Paget razvio i priložio uz izvorni dokument o napadu (Slika 1).



Slika 1: *Shatter* alat

Nakon što napadač posjeduje *handle* na prozor, komunikacija s odgovarajućom aplikacijom može započeti. Slanjem `EM_SETLIMITTEXT` poruke napadač postavlja proizvoljnu vrijednost maksimalne dopuštene duljine znakovnog niza koji se smije upisati u danu kontrolu. Na taj način, slanjem spomenute poruke koja kao prvi parametar sadrži heksadekadsku vrijednost `0xFFFFFFFF`, napadaču je otvorena mogućnost ubacivanja 4 GiB (teoretski) programskog kôda, što je preduvjet za njegovo izvršavanje.

U trećoj fazi, napadač koristi `WM_PASTE` poruku ne bi li u kontrolu ubacio proizvoljni binarni programski kôd ljuske (eng. *shellcode*). Budući da će u sljedećoj fazi biti potrebno saznati memorijsku adresu lokacije na kojoj je izvršni kôd spremljen, najjednostavnije je ubaciti veliki broj `NOP` (*No Operation*) instrukcija, s time da se na početak umetne neka prepoznatljiva sekvenca znakova.

U posljednjoj fazi, napadač mora saznati memorijsku adresu lokacije na kojoj se nalazi izvršni kôd koji je umetnuo. To je moguće ostvariti korištenjem bilo kojeg *debugger* alata (primjerice besplatni *WinDgb*) pomoću kojeg se pretražuje memorijski prostor, i to tražeći specifičnu sekvencu koja označava početak umetnutog izvršnog kôda. Nakon što je saznao adresu željene memorijske lokacije, napadač šalje poruku `WM_TIMER`, čiji je drugi parametar postavljen na vrijednost različitu od nule. `WM_TIMER` poruka kao drugi parametar sadrži adresu funkcije koja se izvodi kada istekne vremenski period zadan prvim parametrom poruke. U slučaju primitka tako oblikovane poruke, aplikacija nastavlja izvođenje s adrese zadane drugim parametrom u poruci. Napadač kao adresu odabire vrijednost nekoliko stotina okteta veću od adrese pronađene *debugger* alatom. Iako su pomaci u memorijskom adresnom prostoru mogući, veliki broj umetnutih `NOP` instrukcija osigurava da će se izvođenje programskog kôda nastaviti od prve „smislene“ napadačeve instrukcije.

Navedeni primjer napada ne predstavlja jedinu mogućnost iskorištavanja nedostataka Windows sustava poruka. Modifikacijom napada moguće je izvesti prepisivanje proizvoljnih dijelova radne memorije koristeći `EM_GETLINE` poruku, izvesti DoS (eng. *Denial of Service*) napad rušenjem aplikacije (u slučaju da rad aplikacije ovisi o količini podataka dohvaćenih iz *edit* kontrole, čija je svojstva moguće promijeniti načinom prikazanim u izvornom napadu) odnosno izazvati prepisivanje spremnika koji se nalazi na programskoj gomili (eng. *heap*) ili stogu (eng. *stack*).

3.1. Microsoft-ova sigurnosna zakrpa

Microsoft je po objavi izvornog dokumenta o *Shatter* napadu potvrdio navedeni propust kod Microsoft Windows NT 4.0, Windows NT 4.0 Terminal Server Edition, Windows 2000 i Windows XP operacijskih sustava, objavio sigurnosno upozorenje**[3]** i odgovarajuće zakrpe. Nedostatak je s njihove strane klasificiran kao visoko rizičan pa razvijena zakrpa modificira svojstva `WM_TIMER` poruke na način da memorijska adresa koja se šalje porukom mora prethodno biti registrirana od strane aplikacije korištenjem `SetTimer()` API funkcije. Zakrpa je osim navedene, djelomično promijenila i funkcionalnost određenog broja Microsoft-ovih procesa koji su pokrenuti kao interaktivni servisi, a sastavni su dio operacijskog sustava.

4. Ranjivost „zakrpanih“ operacijskih sustava

Iako je Microsoft promptno potvrdio i reagirao na ranjivost, zakrpa koju je objavio uklanja isključivo jedan aspekt nedostatka – onaj uzrokovan `WM_TIMER` porukom. U periodu nakon objave službene Microsoft-ove zakrpe, u javnost je izašao velik broj novih primjera iskorištavanja propusta u funkcionalnosti *Windows* sustava poruka**[4]**. Neki od njih napadaču koji ima mogućnost prijavljivanja i rada na lokalnom korisničkom računu omogućavaju izvođenje programskog kôda po volji dok mu drugi omogućavaju prepisivanje proizvoljnih memorijskih lokacija. U nastavku slijedi prikaz i kratki opis nekih od njih.

Nekolicina *Windows* poruka kao parametar `Send_Message()` funkcije primaju povratnu adresu na funkciju. U tu skupinu spadaju poruke: `LVM_SORTITEMS`, `LVM_SORTITEMSEX` te `EM_SETWORDBREAKPROC`. Navedene je poruke moguće iskoristiti za izvođenje programskog kôda po volji te potencijalno neovlašteno stjecanje povišenih korisničkih ovlasti.

Poruka	`LVM_SORTITEMS`
Opis	Koristeći funkciju usporedbe koju definira pozivajuća aplikacija, izvodi sortiranje objekata *View* kontrole.

Način poziva	`Send_Message((HWND)hWndControl,`
	`(UINT)LVM_SORTITEMS,`
	`wParam = (WPARAM)(LPARAM)lParamSort,`
	`lParam = (LPARAM)(PFNLVCOMPARE)pfnCompare);`

Parametri	`lParamSort` – Vrijednost koju definira aplikacija, predaje se funkciji uspoređivanja. `pfnCompare` – Memorijska adresa funkcije uspoređivanja koju definira aplikacija. Funkcija se poziva prilikom operacije sortiranja.

Veliki broj *Windows* poruka kao jedan od parametara prima pokazivač (eng. *pointer*) na strukturu tipa `POINT` ili `RECT`, koji se potom koristi za dohvaćanje `GDI` (*Graphics Device Interface*) informacija o određenom prozoru. Na taj način primljeni pokazivači ne prolaze nikakav proces provjere. U tu skupinu, između ostalog, spadaju sljedeće poruke: `HDM_GETITEMRECT`, `HDM_GETORDERARRAY`, `LVM_GETITEM`, `LVM_GETITEMPOSITION`, `PBM_GETRANGE`, `TVM_GETITEM`.

Poruka	`HDM_GETITEMRECT`
Opis	Određuje pravokutnik koji omeđuje traženi objekt iz zaglavlja kontrole.

Način poziva	`Send_Message((HWND)hWndControl,`
	`(UINT) HDM_GETITEMRECT,`
	`(WPARAM)wParam,`
	`(LPARAM)lParam);`

Parametri	`wParam` – Indeks objekta za kojeg se traži informacija o omeđujućem pravokutniku. `lParam` – Pokazivač na `RECT` strukturu u koju se sprema informacija o omeđujućem pravokutniku.

Slanjem proizvoljne vrijednosti kao `lParam` parametar, proces koji prima poruku izvodi prepisivanje zadane memorijske lokacije, bez ikakve prethodne provjere.

Nekolicina poruka napadaču omogućava zapisivanje proizvoljnih vrijednosti u memorijski prostor gomile (eng. *heap*) namijenjene procesu koji prima poruku. U tu skupinu poruka spadaju: `WM_SETTEXT`, `SB_SETTEXT` te `SB_GETTEXTLENGTH`.

Poruka	`WM_SETTEXT`
Opis	Aplikacija šalje ovu poruku za postavljanje vrijednosti znakovnog niza koji predstavlja ime prozora.
Način poziva	`Send_Message((HWND)hWndControl, (UINT)WM_SETTEXT, wParam = 0, lParam = (LPARAM)(LPCTSTR)lpsz);`
Parametri	`Ipsz` – pokazivač na znakovni niz koji završava `NULL` znakom.

5. Zaključak

Iako se izvorni *Shatter* napad temelji na poruci `WM_TIMER`, ovaj dokument pokazuje da nije bilo dovoljno izdavanje zakrpe koja uvodi ograničenje na navedenu poruku te na taj način onemogućava izvorni napad. U javnost je u proteklih nekoliko godina izašao velik broj napada koji iskorištavaju arhitekturu *Windows* sustava poruka na isti ili sličan način. Kao posljedica, lokalnom se zlonamjernom korisniku otvara mogućnost izvođenja programskog kôda po volji u sigurnosnom kontekstu ranjive aplikacije, što spada u red najkritičnijih sigurnosnih propusta.

Ipak, *Windows* sustav poruka nije isključivi krivac za ranjivost operacijskog sustava na *Shatter* napade. Sama situacija da korisnik s ograničenim ovlastima ima pristup grafičkim sučeljima procesa koji su pokrenuti s najvećim mogućim ovlastima predstavlja sigurnosni propust. *Shatter* napad samo iskorištava navedenu činjenicu te činjenicu da *Windows* sustav poruka omogućava komunikaciju procesa s različitim razinama ovlasti. Upravo zbog toga, Microsoft kao proizvođač ranjivih operacijskih sustava nije jedini odgovoran za *Shatter* ranjivost – ostali proizvođači programskih paketa, pogotovo onih koji pristupaju osjetljivim resursima (poput antivirusnih alata), moraju jezgru funkcionalnosti odvojiti od komponente namijenjene interakciji s korisnikom. Iako je ta činjenica oduvijek poznata, *Shatter* napad uvelike je podigao stupanj svijesti o navedenom problemu tako da danas, nekoliko godina nakon prve pojave napada, većina takvih aplikacija na siguran način provodi komunikaciju s korisnikom.

6. Reference

[1] Microsoft MSDN,
http://msdn1.microsoft.com
[2] Shatter Attacks – How to break Windows, Chris Paget,
http://security.tombom.co.uk/shatter.html
[3] Microsoft Security Bulletin MS02-071,
http://www.microsoft.com/technet/security/bulletin/MS02-071.mspx
[4] Shattering By Example, Brett Moore,
http://www.security-assessment.com/Whitepapers/Shattering_By_Example-V1_03102003.pdf
[5] Win32 Message Vulnerabilities Redux, iDefense,
http://www.rootsecure.net/content/downloads/pdf/shatter_attack_redux.pdf