Studio di buffer overflow in ambiente Win32 e realizzazione di un tool per la ricerca automatica di programmi vulnerabili a questo tipo di attacco

Università degli Studi di Milano
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Triennale in Informatica

Relatore: Ing. Mattia MONGA

Tesi di Laurea di:
Davide MARRONE
Matricola 654241

Sommario

Sommario

Introduzione

Definizione del problema

Con il termine *buffer overflow* si indica la scrittura in un buffer (o array) di una quantità di dati superiore alla sua capacità

Introduzione

Definizione del problema

Con il termine *buffer overflow* si indica la scrittura in un buffer (o array) di una quantità di dati superiore alla sua capacità

- Alcuni linguaggi di programmazione non controllano che la scrittura avvenga esclusivamente nei limiti del buffer

Introduzione

Definizione del problema

Con il termine *buffer overflow* si indica la scrittura in un buffer (o array) di una quantità di dati superiore alla sua capacità

- Alcuni linguaggi di programmazione non controllano che la scrittura avvenga esclusivamente nei limiti del buffer

La causa del problema

- Assenza nei linguaggi di programmazione di controlli automatici di integrità

Introduzione

Definizione del problema

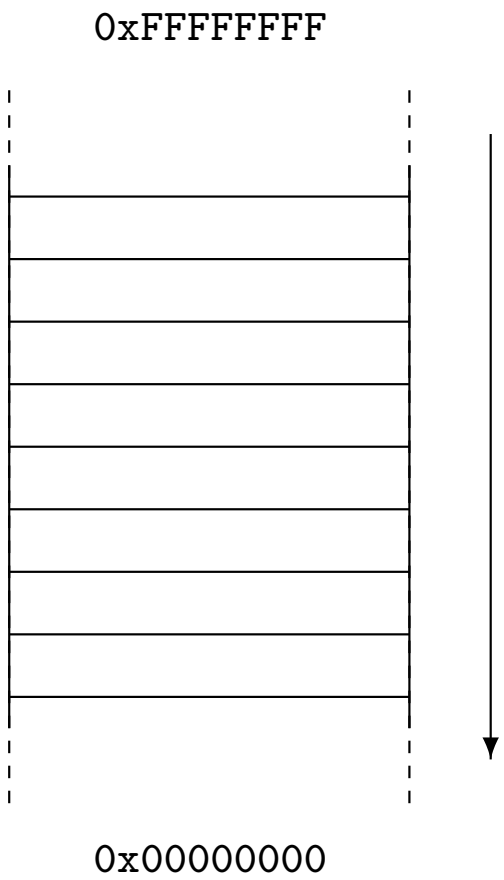
Con il termine *buffer overflow* si indica la scrittura in un buffer (o array) di una quantità di dati superiore alla sua capacità

- Alcuni linguaggi di programmazione non controllano che la scrittura avvenga esclusivamente nei limiti del buffer

La causa del problema

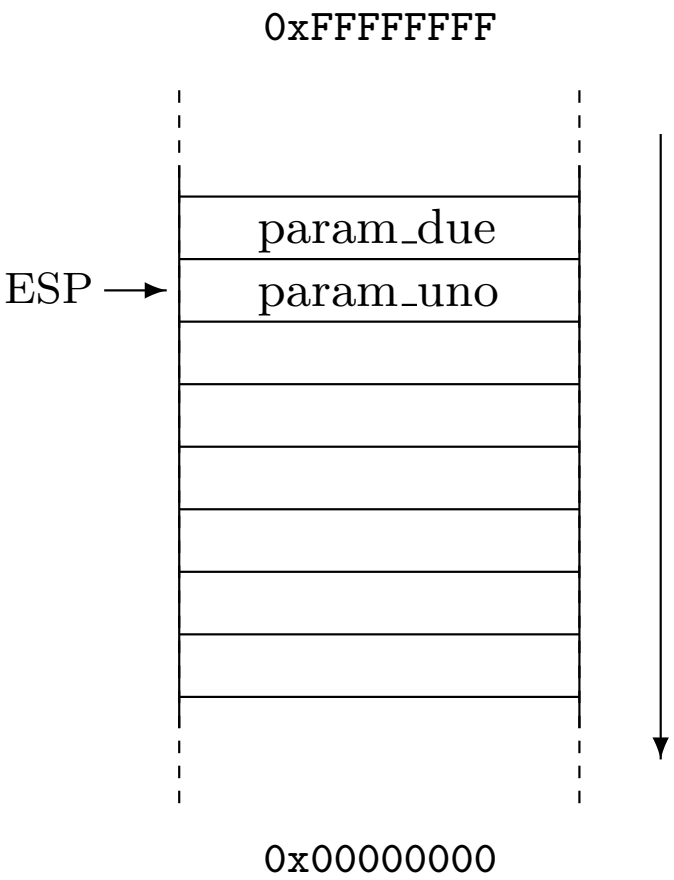
- Assenza nei linguaggi di programmazione di controlli automatici di integrità
- Assenza nei programmi di controlli accurati sull'input ricevuto

Sovvertire il normale funzionamento di un programma



Chiamare una funzione

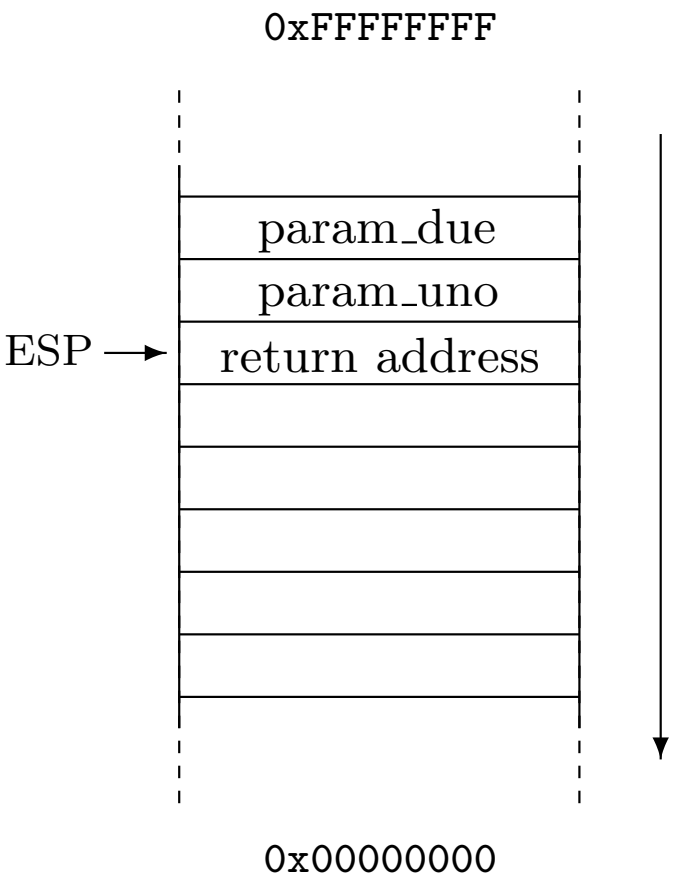
Sovvertire il normale funzionamento di un programma



Chiamare una funzione

1. I parametri sono passati sullo stack in ordine inverso

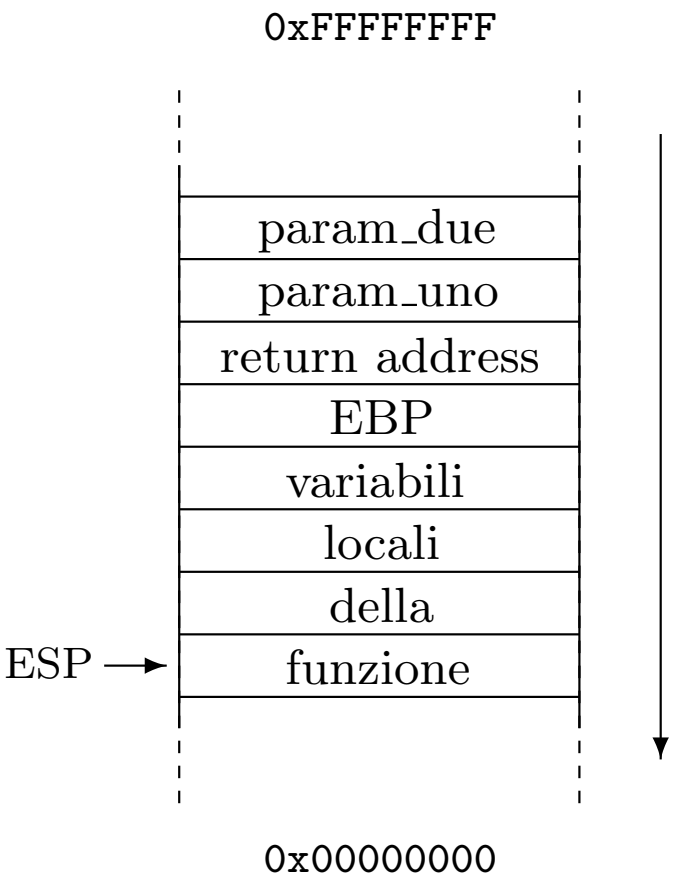
Sovvertire il normale funzionamento di un programma



Chiamare una funzione

1. I parametri sono passati sullo stack in ordine inverso
2. L'istruzione call salva sullo stack il return address

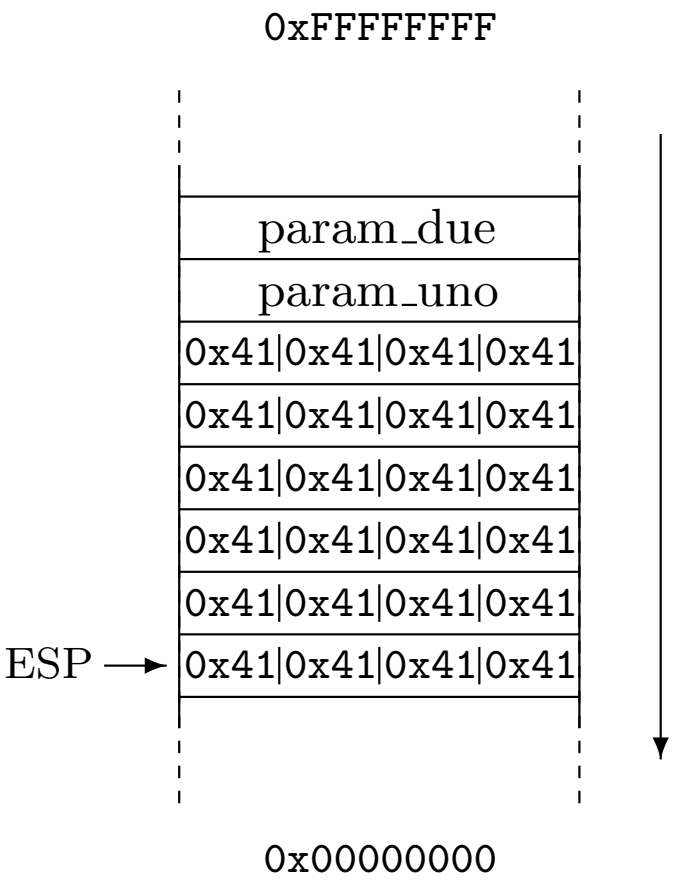
Sovvertire il normale funzionamento di un programma



Chiamare una funzione

1. I parametri sono passati sullo stack in ordine inverso
2. L'istruzione call salva sullo stack il return address
3. Il prologo della funzione alloca lo spazio per le variabili locali

Sovvertire il normale funzionamento di un programma



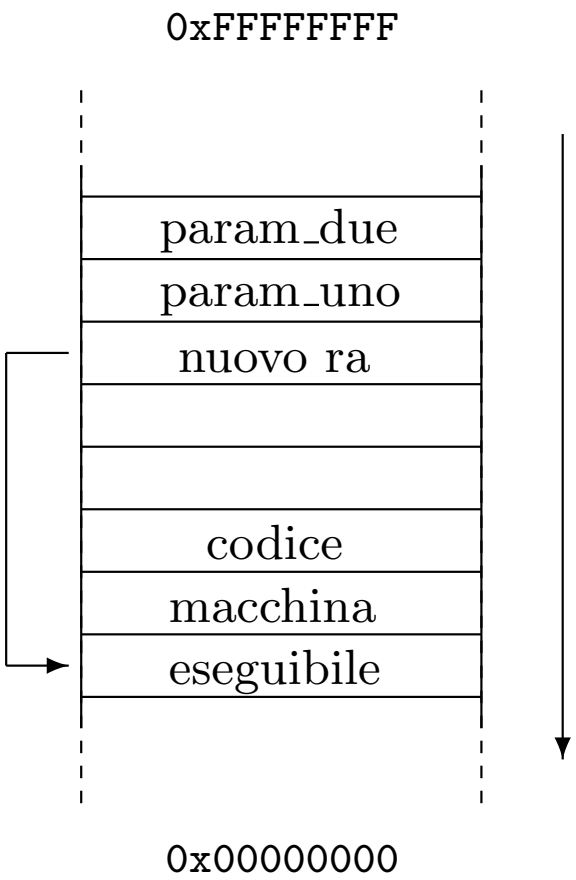
Chiamare una funzione

1. I parametri sono passati sullo stack in ordine inverso
2. L'istruzione call salva sullo stack il return address
3. Il prologo della funzione alloca lo spazio per le variabili locali

Buffer overflow

- È possibile prendere il controllo del return address

Sovvertire il normale funzionamento di un programma



Chiamare una funzione

1. I parametri sono passati sullo stack in ordine inverso
2. L'istruzione call salva sullo stack il return address
3. Il prologo della funzione alloca lo spazio per le variabili locali

Buffer overflow

- È possibile prendere il controllo del return address
- È possibile far eseguire al processo del codice arbitrario

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice
- Modifiche ai compilatori per aggiungere del codice nel prologo e nell'epilogo delle funzioni (tecnica canary)

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice
- Modifiche ai compilatori per aggiungere del codice nel prologo e nell'epilogo delle funzioni (tecnica canary)

Proposta del CERT–IT

Il CERT–IT per l'ambiente unix ha sviluppato OBOE (Object–code Buffer Overflow Evaluator)

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice
- Modifiche ai compilatori per aggiungere del codice nel prologo e nell'epilogo delle funzioni (tecnica canary)

Proposta del CERT–IT

Il CERT–IT per l'ambiente unix ha sviluppato OBOE (Object–code Buffer Overflow Evaluator)

- Si parte dal presupposto che il codice sorgente del target non sia a disposizione

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice
- Modifiche ai compilatori per aggiungere del codice nel prologo e nell'epilogo delle funzioni (tecnica canary)

Proposta del CERT–IT

Il CERT–IT per l'ambiente unix ha sviluppato OBOE (Object–code Buffer Overflow Evaluator)

- Si parte dal presupposto che il codice sorgente del target non sia a disposizione
- Si effettua del fault–injection sui programmi

Come limitare il problema

Soluzioni esistenti

- Ispezione automatica del codice
- Modifiche ai compilatori per aggiungere del codice nel prologo e nell'epilogo delle funzioni (tecnica canary)

Proposta del CERT–IT

Il CERT–IT per l'ambiente unix ha sviluppato OBOE (Object–code Buffer Overflow Evaluator)

- Si parte dal presupposto che il codice sorgente del target non sia a disposizione
- Si effettua del fault–injection sui programmi

Scopo di questo lavoro

Realizzare OBOE per l'ambiente Win32 (Win–OBOE)

OBOE per l'ambiente Win32

Obbiettivo

Individuare buffer overflow causati dall'input che viene passato al programma

OBOE per l'ambiente Win32

Obbiettivo

Individuare buffer overflow causati dall'input che viene passato al programma

Input considerato

Tutti i dati di input, se gestiti in maniera scorretta, possono causare un buffer overflow, le due fonti analizzate sono:

OBOE per l'ambiente Win32

Obbiettivo

Individuare buffer overflow causati dall'input che viene passato al programma

Input considerato

Tutti i dati di input, se gestiti in maniera scorretta, possono causare un buffer overflow, le due fonti analizzate sono:

1. Parametri passati a linea di comando

OBOE per l'ambiente Win32

Obbiettivo

Individuare buffer overflow causati dall'input che viene passato al programma

Input considerato

Tutti i dati di input, se gestiti in maniera scorretta, possono causare un buffer overflow, le due fonti analizzate sono:

1. Parametri passati a linea di comando
2. Chiavi di registro

Composizione dell'input per la ricerca di buffer overflow

Stringhe utilizzate

- Stringhe composte da caratteri fissi

Composizione dell'input per la ricerca di buffer overflow

Stringhe utilizzate

- Stringhe composte da caratteri fissi
  $\longrightarrow$ formano degli indirizzi che fanno riferimento a zone di memoria del *kernel*

Composizione dell'input per la ricerca di buffer overflow

Stringhe utilizzate

- Stringhe composte da caratteri fissi
  $\longrightarrow$ formano degli indirizzi che fanno riferimento a zone di memoria del *kernel*
- Stringhe composte da caratteri pseudocasuali

Composizione dell'input per la ricerca di buffer overflow

Stringhe utilizzate

- Stringhe composte da caratteri fissi
  $\longrightarrow$ formano degli indirizzi che fanno riferimento a zone di memoria del *kernel*
- Stringhe composte da caratteri pseudocasuali
  $\longrightarrow$ sono utilizzate per diversificare i possibili flussi di esecuzione dei programmi

Come individuare un buffer overflow

Stategia utilizzata

Per capire se sia presente un buffer overflow:

1. Si crea il target passando una stringa con la lunghezza massima per il tipo di input considerato

Come individuare un buffer overflow

Stategia utilizzata

Per capire se sia presente un buffer overflow:

1. Si crea il target passando una stringa con la lunghezza massima per il tipo di input considerato
2. Si analizza il comportamento del processo in funzione dell'input

Come individuare un buffer overflow

Stategia utilizzata

Per capire se sia presente un buffer overflow:

1. Si crea il target passando una stringa con la lunghezza massima per il tipo di input considerato
2. Si analizza il comportamento del processo in funzione dell'input
3. Se il processo genera un'eccezione si cerca di determinare la posizione del return address

Come individuare un buffer overflow

Stategia utilizzata

Per capire se sia presente un buffer overflow:

1. Si crea il target passando una stringa con la lunghezza massima per il tipo di input considerato
2. Si analizza il comportamento del processo in funzione dell'input
3. Se il processo genera un'eccezione si cerca di determinare la posizione del return address
4. Si diminuisce la stringa fino a sovrascrivere esattamente i quattro byte del return address

Come individuare un buffer overflow

Stategia utilizzata

Per capire se sia presente un buffer overflow:

1. Si crea il target passando una stringa con la lunghezza massima per il tipo di input considerato
2. Si analizza il comportamento del processo in funzione dell'input
3. Se il processo genera un'eccezione si cerca di determinare la posizione del return address
4. Si diminuisce la stringa fino a sovrascrivere esattamente i quattro byte del return address
5. Quando viene trovata la posizione del return address si tenta di far eseguire al processo del codice arbitrario

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

- In 8 programmi sono state rilevate diverse eccezioni

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

- In 8 programmi sono state rilevate diverse eccezioni
- In un caso (mysql.exe) Win–OBOE è riuscito in automatico a far eseguire del codice arbitrario al target

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

- In 8 programmi sono state rilevate diverse eccezioni
- In un caso (mysql.exe) Win–OBOE è riuscito in automatico a far eseguire del codice arbitrario al target

Chiavi di registro

Sono stati testati 60 programmi che utilizzano il registro

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

- In 8 programmi sono state rilevate diverse eccezioni
- In un caso (mysql.exe) Win–OBOE è riuscito in automatico a far eseguire del codice arbitrario al target

Chiavi di registro

Sono stati testati 60 programmi che utilizzano il registro

- Non è stata rilevata nessuna eccezione

Prove effettuate

Parametri passati a linea di comando

Sono stati testati 40 programmi che accettano parametri da linea di comando

- In 8 programmi sono state rilevate diverse eccezioni
- In un caso (mysql.exe) Win–OBOE è riuscito in automatico a far eseguire del codice arbitrario al target

Chiavi di registro

Sono stati testati 60 programmi che utilizzano il registro

- Non è stata rilevata nessuna eccezione
  $\rightarrow$ Risultato dovuto al funzionamento della API che legge le stringhe dal registro

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile
- In ambiente open–source sono disponibili delle patch per il kernel del sistema operativo (PaX)

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile
- In ambiente open–source sono disponibili delle patch per il kernel del sistema operativo (PaX)
- Win–OBOE può essere uno strumento valido soprattutto per altri tipi di input

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile
- In ambiente open–source sono disponibili delle patch per il kernel del sistema operativo (PaX)
- Win–OBOE può essere uno strumento valido soprattutto per altri tipi di input
- Si può estendere la versione sviluppata per analizzare altre fonti di input

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile
- In ambiente open–source sono disponibili delle patch per il kernel del sistema operativo (PaX)
- Win–OBOE può essere uno strumento valido soprattutto per altri tipi di input
- Si può estendere la versione sviluppata per analizzare altre fonti di input
  $\longrightarrow$ Socket

Conclusioni e Sviluppi Futuri

Conclusioni e Sviluppi Futuri

- I buffer overflow ancora oggi sono un problema, soprattutto per i programmi in cui il sorgente non è disponibile
- In ambiente open–source sono disponibili delle patch per il kernel del sistema operativo (PaX)
- Win–OBOE può essere uno strumento valido soprattutto per altri tipi di input
- Si può estendere la versione sviluppata per analizzare altre fonti di input
  - $\longrightarrow$ Socket
  - $\longrightarrow$ File