



Secure Code Review: dalla teoria alla pratica

Antonio Parata
<http://www.emaze.net>

Antonio.parata@emaze.net

OWASP-Day III

Centro di Competenza ICT-Puglia - Dipartimento di Informatica
Università degli Studi di Bari

23rd February 2009 - Bari (Italy)

Copyright © 2009 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License.

The OWASP Foundation
<http://www.owasp.org>

Agenda



Introduzione



Come condurre un Code Review



Tool di analisi statica



Code Review in pratica



Conclusioni



Chi sono

- Security Consultant in Emaze Networks
- Collaboro con il gruppo di ricerca indipendente USH
- Co-autore dell'owasp testing guide 3.0 e 2.0
- Application Security Enthusiast



Code Review – Chi, Come, Dove, Quando e Perché

- **Chi:** tipicamente svolto da un team di sviluppatori ed esperti di sicurezza (auditors)
- **Come:** gli auditor analizzano il codice sorgente alla ricerca di vulnerabilità
- **Dove:** in sessioni di circa 4 ore con una o due pause



Code Review – Chi, Come, Dove, Quando e Perché

● **Quando:** durante la fase di implementazione del software (o post rilascio)

● **Perché:**

- Permette di identificare un maggior numero di vulnerabilità
- Permette di applicare delle soluzioni ottimali già dalle prime fasi del ciclo di sviluppo
- Verifica che il codice sia scritto seguendo best practices e coding standards



Stato della sicurezza delle applicazioni web

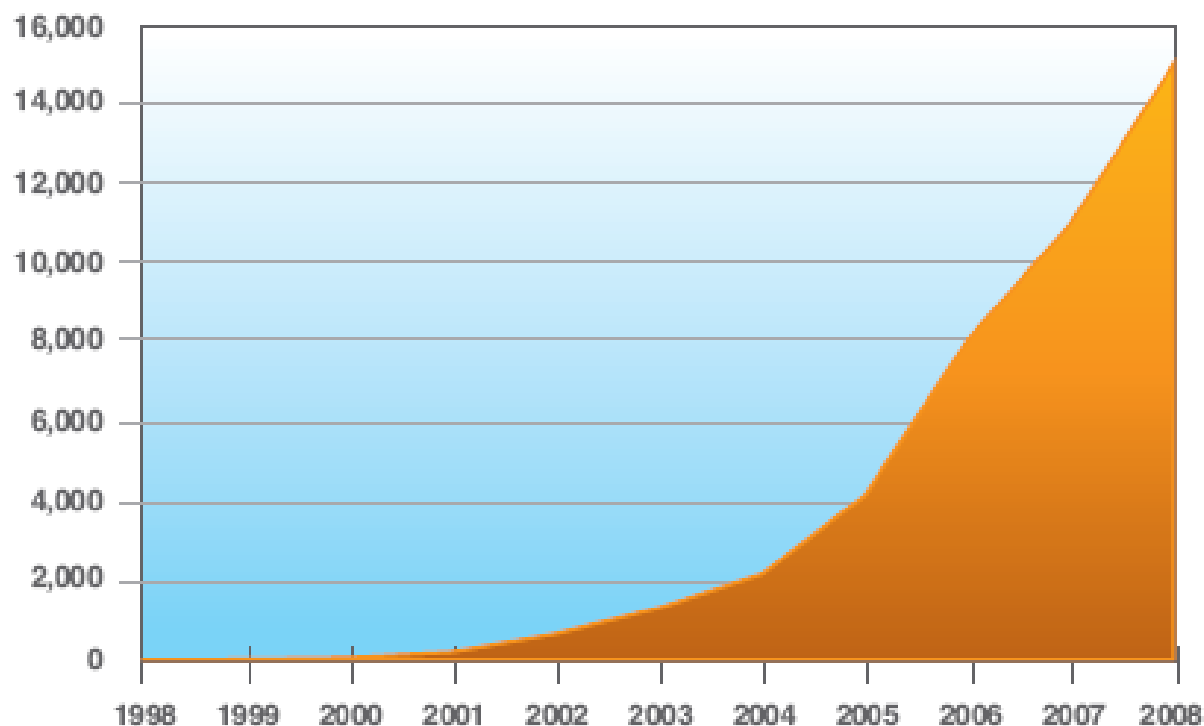
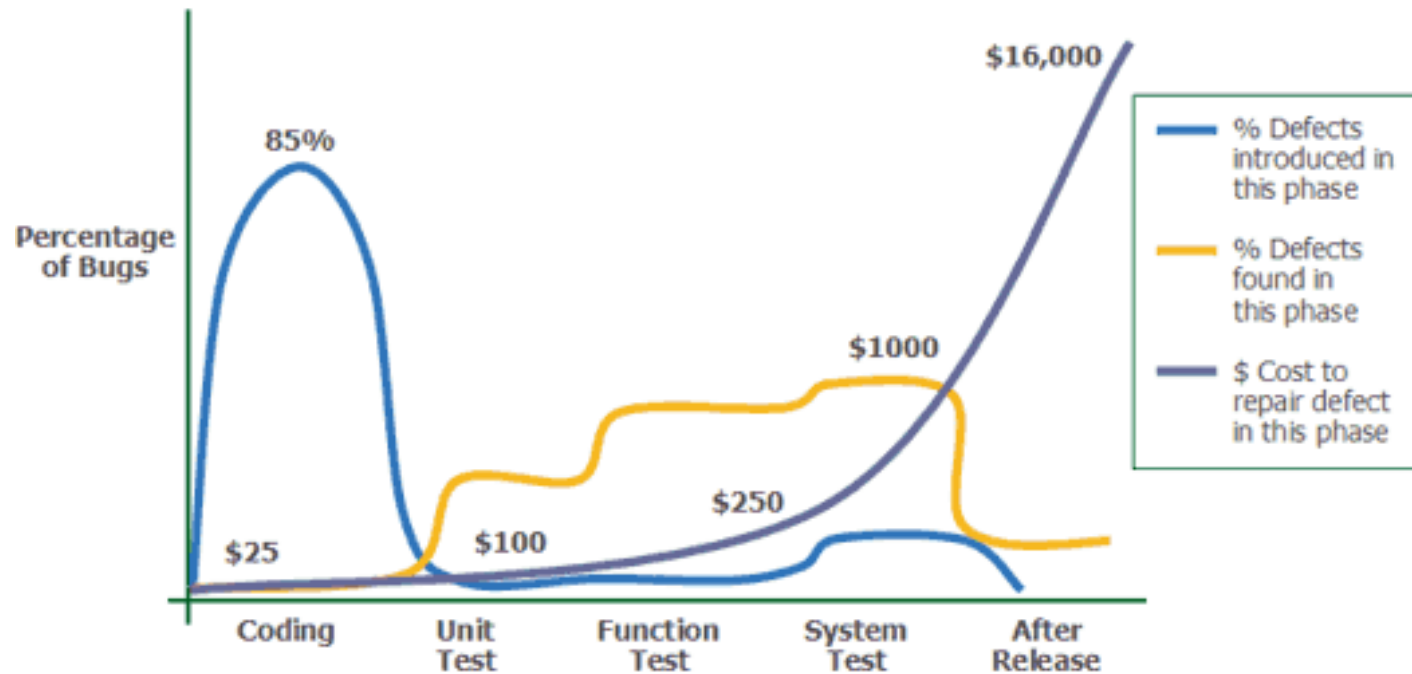


Figure 16: Cumulative Count of Web Application Vulnerabilities, 1998 – 2008

*fonte: **IBM Internet Security Systems X-Force® 2008 Trend & Risk Report**



Code Review nel Secure Development Lifecycle



*fonte: **Applied Software Measurement**, Capers Jones, 1986



Metodologie per la rimozione dei difetti a confronto

	Requirements Defects	Design Defects	Code Defects	Document Defects	Performance Defects
Reviews/ Inspections	Fair	Excellent	Excellent	Good	Fair
Prototypes	Good	Fair	Fair	Not Applicable	Good
Testing (all forms)	Poor	Poor	Good	Fair	Excellent
Correctness Proofs	Poor	Poor	Good	Fair	Poor

Figure 5-3 Defect removal methods

fonte: **Applied Software Measurement, Capers Jones, 2008*



Agenda

- Introduzione

- Come condurre un Code Review

- Tool di analisi statica



- Code Review in pratica

- Conclusioni





Attività di Code Review

Code Walkthroughs

-  Simulazione del codice “by hand”
-  Durante la simulazione si cerca di identificare eventuali errori di qualsiasi tipo

Code Inspection

-  Si guarda il codice da una prospettiva diversa, ovvero si cerca di identificare specifiche tipologie di errori
-  È bene specificare o avere in mente che tipologie di errore si intende identificare



Code walkthroughs

- Gruppi da massimo 3/5 persone
- I partecipanti sono in possesso (da prima di inizio della review) di un documento scritto che esponde l'architettura dell'applicazione
- I meeting hanno una durata prefissata (alcune ore)
- Il focus è l'identificazione degli errori e non la loro risoluzione
- I ruoli sono: designer, moderatore, auditor e esperti di sicurezza del codice
- Mutua cooperazione; non bisogna valutare l'operato dei programmatori
 - L'esperienza mostra che la maggior parte degli errori viene individuata dal designer all'atto della spiegazione del design dell'applicazione



Code inspection

- Tecnica di lettura del codice basata sull'**identificazione di errori** attraverso delle **Checklist**
- Checklist basate su
 - Uso di funzioni considerate "*non safe*"
 - Uso di variabili non inizializzate
 - Array indexes out of bounds
 - Confronti tra variabili di tipo signed e unsigned
- Esistono svariate Checklist già pronte all'uso
 - <http://www.sans.org/score/checklists/WebApplicationChecklist.pdf>



Condurre un Code Review

- Modalità di navigazione del codice
 - Control-flow sensitive
 - Data-flow sensitive
- Strategia di Code Auditing
 - Trace Malicious Input (taint propagation)
 - Candidate Point Strategies (sink point)



Control-flow sensitive VS Data-flow sensitive

```
1. int bob(int c) {  
2.     if (c == 4)  
3.         fred(c);  
4.     if (c == 72)  
5.         jim();  
6.     for (; c; c)  
7.         updateglobalstate();  
8. }
```

➊ Control-flow sensitive → 123|45|67|8

➋ Data-flow sensitive (var c) → 123|46|8



Trace Malicious Input + Control-flow sensitive

```
1. $myvar = $_GET['name'];
2. if ($counter > 0)
3.     sayhello("Ben tornato ".$myvar);}
4. else
5.     sayhello("Benvenuto");}
6. function sayhello($msg)
7. {
8.     print_header();
9.     print_menu();
10. print $msg;
11.     print_footer();
12. }
```

The diagram illustrates the flow of control from the input variable `$myvar` to the output of the `sayhello` function. Blue arrows trace the path from the assignment of `$myvar` in line 1, through the conditional logic in lines 2-5, and into the `sayhello` function in line 6. The arrows continue through the function's body (lines 8-11) and point towards the word **vulnerabilità** (vulnerability), indicating that the output is directly controlled by the input without any sanitization or validation.



Candidate Point Strategies + Data-flow sensitive

```
1. $U = $_POST['username'];
2. $P = $_POST['password'];
3. if (!isset($U) || !isset($P)) {
4.     echo "Password non valida";
5. }
6. Else {
7.     $resOk = check_login($U, $P);
8.     if ($resOk) {
9.         print "Accesso consentito";
10.        doAdminStuff();
11.    }
12.    else {
13.        print "Password e/o Username errati";
14.    }
15. }
16. function check_login($username, $password) {
17.    $sql = "SELECT count(*) FROM Utenti WHERE
18.        Username='$username' AND Password=MD5('$password')";
19.    $res = execute_query($sql);
20.    return $res > 0;
21. }
```

vulnerabilità



Agenda

- Introduzione
- Come condurre un Code Review
- Tool di analisi statica
- Code Review in pratica
- Conclusioni

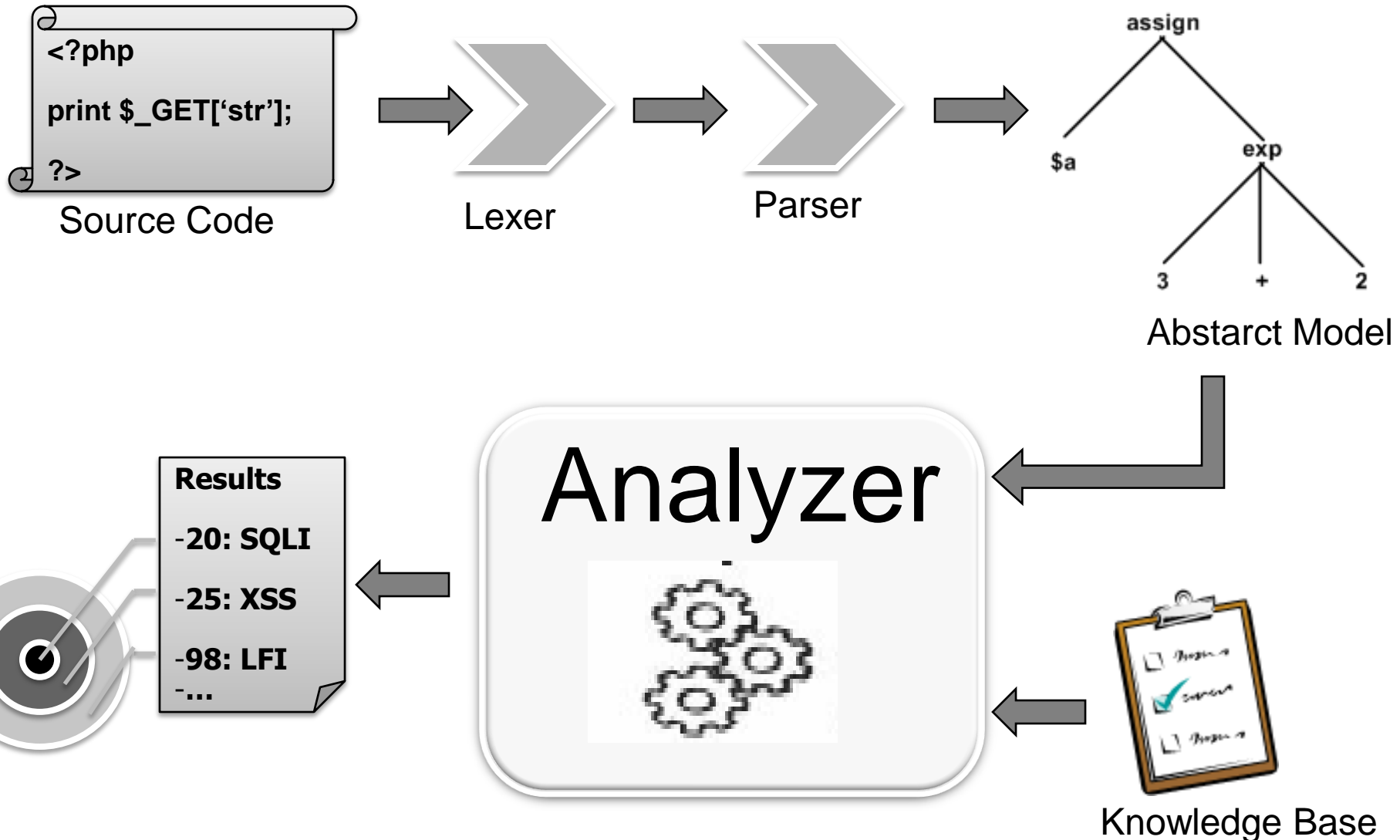


Analisi statica

- Analizza il codice senza eseguirlo (simulazione)
- Considera più fattori e path di esecuzione (anche dette tracce di esecuzione)
- Non é a conoscenza di cosa fa il codice (cosa buona)
- Permette di ripetere l'analisi periodicamente in modo automatico (valutazione dell'andamento della sicurezza)



Analizzatori statici – Internals



Tool esistenti a confronto

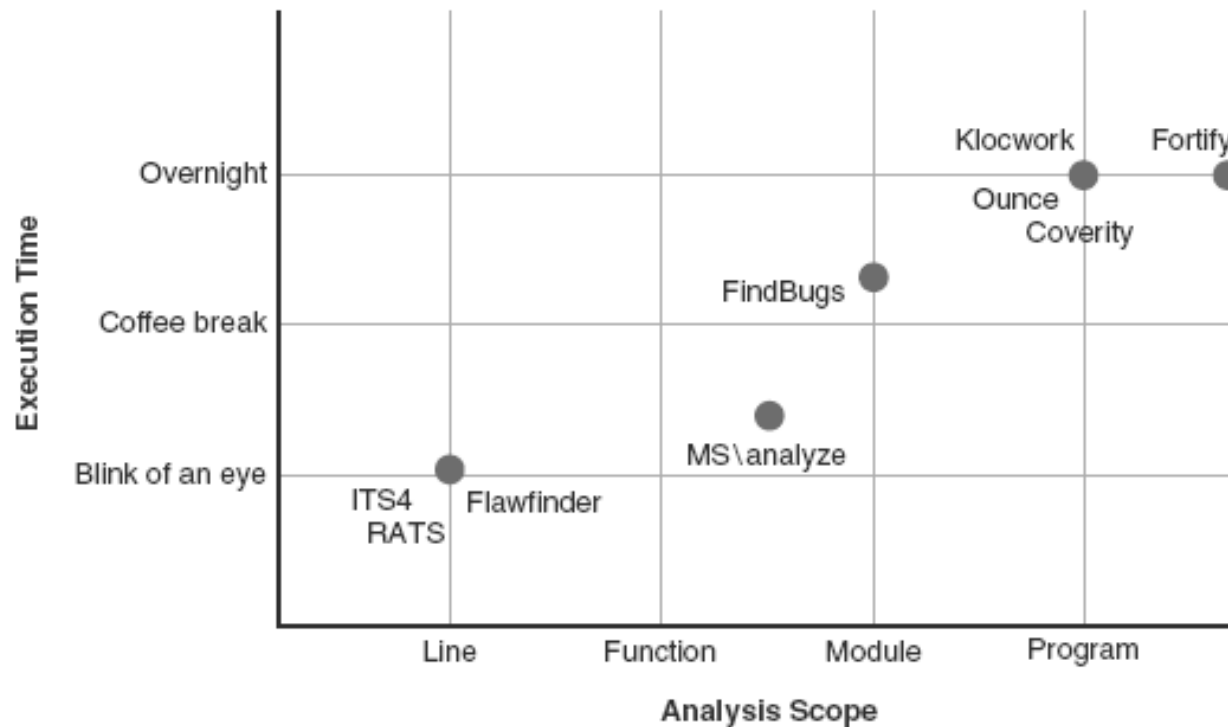



Figure 2.2 Analysis scope vs. execution time for the bug finding and security tools discussed in Section 2.1.

**fonte: Secure Programming with Static Analysis*




Tool di analisi web based

Owasp Orizon

-  "This project born in 2006 in order to provide a framework to all Owasp projects developing code review services. The project is in a quite stable stage and it is usable for Java static code review and some dynamic tests against XSS. Owasp Orizon includes also APIs for code crawling, usable for code crawling tools."

Project Leader: Paolo Perego






Pixy

-  "Pixy is a Java program that performs automatic scans of PHP 4 source code, aimed at the detection of XSS and SQL injection vulnerabilities. Pixy takes a PHP program as input, and creates a report that lists possible vulnerable points in the program, together with additional information for understanding the vulnerability."



Tool di analisi web based

Codeminer

-  Tool di analisi statica per PHP
-  Identifica vulnerabilità di vario genere (non è legato a nessuna tipologia in particolare)
-  Simulazione delle funzioni più comuni di PHP (maggiore efficienza e minor numero di falsi positivi)
-  Estendibile attraverso plugin
-  ... ancora in fase di sviluppo 😊



Agenda

- Introduzione
- Come condurre un Code Review
- Tool di analisi statica
- Code Review in pratica
- Conclusioni



Spot the Bug – Candidate Point Strategies

```
1. function _holiday_cmp($a,$b) {  
2.   if (($year_diff = ($a['occurrence'] <= 0 ?  
3.     0 : $a['occurrence']) - ($b['occurrence'] <= 0 ?  
4.     0 : $b['occurrence'])))  
5.   {  
6.     return $year_diff;  
7.   }  
8.   return $a['month'] - $b['month'] ?  
9.     $a['month'] - $b['month'] : $a['day'] - $b['day'];  
10.}
```

```
11.$send_back_to=str_replace('&locale='.$_POST['locale'],",$send_back_to);  
12.$file = './holidays.'$_POST['locale'].'.csv';
```

```
13.if(!file_exists($file) || filesize($file) < 300) {  
14.  if (count($_POST['name'])) {  
15.    $fp = fopen($file,'w');  
16.    if ($_POST['charset'])  
17.      fwrite($fp,"charset\t".$_POST['charset']."\n");  
18.    $holidays = array();
```



Spot the Bug – Candidate Point Strategies

```
1. function _holiday_cmp($a,$b) {
2.   if (($year_diff = ($a['occurence'] <= 0 ?
3.     0 : $a['occurence']) - ($b['occurence'] <= 0 ?
4.     0 : $b['occurence'])))
5.   {
6.     return $year_diff;
7.   }
8.   return $a['month'] - $b['month'] ?
9.     $a['month'] - $b['month'] : $a['day'] - $b['day'];
10.}
```



```
11.$send_back_to=str_replace('&locale='.$_POST['locale'],",$send_back_to);
12.$file = './holidays.'$_POST['locale'].'.csv';
```

```
13.if(!file_exists($file) || filesize($file) < 300)
14.  if (count($_POST['name'])) {
15.    $fp = fopen($file,'w');
16.    if ($_POST['charset'])
17.      fwrite($fp,"charset\t".$_POST['charset']."\n");
18.    $holidays = array();
```

vulnerabilità



Agenda




- Introduzione
- Come condurre un Code Review
- Tool di analisi statica
- Code Review in pratica

● Conclusioni



Conclusioni

Code Review

-  Identifica le vulnerabilità già durante la fase di implementazione (diminuiscono i costi di *patching*)
-  Molto efficace nell'identificazione di errori
-  Attività Brain Intensive (utilizzare tool di analisi statica)

Analisi statica e Secure Code Review vanno di pari passo

-  Secure Code Review è un servizio “artigianale” e non può essere simulato da un programma



Domande?

Antonio Parata – Security Consultant

<http://www.emaze.net>

Antonio.parata@emaze.net

