7.1.4 Resource Injection

Come riconoscerla

L'applicazione apre un socket di rete, per l'ascolto delle connessioni in entrata, utilizzando dati non attendibili per configurarlo, consentendo a un eventuale malintenzionato di controllarlo.

Il malintenzionato potrebbe perciò essere in grado di aprire una backdoor che gli consenta di connettersi direttamente al server delle applicazioni, acquisendo il controllo del server o esponendolo ad altri attacchi indiretti. In particolare, modificando il numero di porta del socket, un utente malintenzionato può essere in grado di aggirare controlli di rete deboli, mascherando l'attacco da parte di altri dispositivi di rete.
Una resource injection può essere sfruttata anche per bypassare i firewall o altri meccanismi di controllo degli accessi. Si può anche utilizzare l'applicazione come proxy per la scansione delle porte delle reti interne e per l'accesso diretto ai sistemi locali; oppure indurre un utente a inviare informazioni riservate a un server fasullo.

Come difendersi

Non consentire a un utente di definire i parametri relativi ai sockets di rete. Il principio della white list può essere adottato per scegliere un valore tra quelli ammissibili, codificandoli – ad esempio - in una switch.

Esempio:

```
int main( int argc, char* argv[] )
{
	int sockfd, portno;
	struct sockaddr_in serv_addr = {};
	struct hostent *server;

	if ( argc != 3 )
		errorAndExit();

	server = gethostbyname(argv[1]);
	if (server == NULL)
		errorAndExit();

	portno = atoi(argv[2]);

	serv_addr.sin_family = AF_INET;
	memcpy(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
	serv_addr.sin_port = htons(portno);

	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		errorAndExit();

	if (connect(sockfd, &serv_addr, sizeof(serv_addr)) < 0)
		errorAndExit();

	sendAndProcessMessage(sockfd);

	close(sockfd);
}
```

In questo esempio la configurazione del socket viene realizzata con l'input non verificato proveniente dall'utente. Qui di seguito la scelta è ristretta a una white list:

```
int main( int argc, char* argv[] )
{
	int sockfd, portno;
	struct sockaddr_in serv_addr = {};
	char* portname;

	if ( argc != 1 )
		errorAndExit();
```